2011

Combinatorial optimization problems in geometric settings

Gaurav Nandkumar Kanade
University of Iowa

COMBINATORIAL OPTIMIZATION PROBLEMS

IN

GEOMETRIC SETTINGS

by

Gaurav Nandkumar Kanade

An Abstract

Of a thesis submitted in partial fulfillment of the
requirements for the Doctor of Philosophy
degree in Computer Science
in the Graduate College of
The University of Iowa

July 2011

Thesis Supervisor: Associate Professor Kasturi Varadarajan

ABSTRACT

We consider several *combinatorial optimization problems* in a *geometric setting*. The first problem we consider is the problem of *clustering to minimize the sum of radii*. Given a positive integer k and a set of points with interpoint distances that satisfy the definition of being a "metric", we define a ball centered at some input point and having some radius as the set of all input points that are at a distance smaller than the radius of the ball from its center. We want to cover all input points using at most k balls so that the sum of the radii of the balls chosen is minimized. We show that when the points lie in some Euclidean space and the distance measure is the standard Euclidean metric, we can find an exact solution in polynomial time under standard assumptions about the model of computation.

The second problem we consider is the *Network Spanner Topology Design* problem. In this problem, given a set of nodes in the network, represented by points in some geometric setting - either a plane or a 1.5-D terrain, we want to compute a height assignment function h that assigns a height to a tower at every node such that the set of pairs of nodes that can form a direct link with each other under this height function forms a connected spanner. A pair of nodes can form a direct link if they are within a bounded distance B of each other and the heights of towers at the two nodes are sufficient to achieve Line-of-Sight connectivity - i.e. the straight line connecting the top of the towers lies above any obstacles. In the planar setting where the obstacles are modeled as having a certain maximum height and minimum

clearance distance, we give a constant factor approximation algorithm. In the case where the points lie on a 1.5-D terrain we illustrate that it might be hard to use Computational Geometry to achieve efficient approximations.

The final problem we consider is the *Multiway Barrier Cut* problem. Here, given a set of points in the plane and a set of unit disk sensors also in the plane such that any path in the plane between any pair of input points hits at least one of the given sensor disks we consider the problem of finding the minimum size subset of these disks that still achieves this separation. We give a constant factor approximation algorithm for this problem.

Abstract Approved: _____

Thesis Supervisor

_____

Title and Department

_____

Date

COMBINATORIAL OPTIMIZATION PROBLEMS

IN

GEOMETRIC SETTINGS

by

Gaurav Nandkumar Kanade

A thesis submitted in partial fulfillment of the
requirements for the Doctor of Philosophy
degree in Computer Science
in the Graduate College of
The University of Iowa

July 2011

Thesis Supervisor: Associate Professor Kasturi Varadarajan

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

PH.D. THESIS

This is to certify that the Ph.D. thesis of

Gaurav Nandkumar Kanade

has been approved by the Examining Committee for the thesis requirement for the Doctor of Philosophy degree in Computer Science at the July 2011 graduation.

Thesis Committee: _____
Kasturi Varadarajan, Thesis Supervisor

_____
Sriram Pemmaraju

_____
Sukumar Ghosh

_____
Alberto Segre

_____
Samuel Burer

ACKNOWLEDGEMENTS

There have been many people who have made a major influence in my experience doing algorithms research. Firstly, I must thank Prof. Kasturi Varadarajan. His guidance has helped me to understand the importance and feasibility of open research problems, to be able to efficiently search for relevant work in a field of interest and attack a problem looking at all possible aspects of it. He is a coauthor on all of the work presented in this document as well as other papers I have coauthored.

I must also thank Matt Gibson, Erik Krohn and Imran Pirwani. For about two to three years we were actively invovled in the Computational Geometery research and readings group here. I have learnt so much about research from the experience of my interactions with them as each of them brought their unique and vital contributions to every problem we looked at. All three played a crucial part on the work done on geometric clustering as well as a related metric clustering problem not considered in this document. Recently Matt and Erik also played a crucial role in another paper we coauthored that is not documented here. Matt was also deeply involved with me on my last work on barrier coverage and besides his immense contribution in research, can always be relied upon for virtually anything we need when it comes to writing papers.

I would also like to thank Dr. Sachin Garg, my mentor and manager during my summer internships at Motorola Labs, India and Yahoo! Labs, India. It was working under him here that I got interested in working on the Network Design problem

considered in this document. Working with him also gave me valuable perspective that a domain expert and a person in the research industry can bring.

I would also like to thank Dr. Sriram Pemmaraju. I took several algorithms courses and learned an incredible amount from him. He was also always keen and eager to discuss any research problems and provide invaluable suggestions. I would also like to thank Dr. Alberto Segre for giving me the opportunity to work with the Computational Epidemiology group where I had a different and invaluable experience of using algorithmic knowledge in an applied setting.

I would also like to thank Dr. Sam Burer. I enjoyed my linear programming course with him. This class really made me appreciate the importance of details and small margins in research. I would also like to thank Dr. Ghosh who was always very helpful with his advice on academic matters as well as making me feel at home when I first came to the country.

I would also like to thank Chandra Chekuri for valuable suggestions for the geometric clustering problem and the network design problem, Alon Efrat for his suggestion that led to the formulation of optimization problems in barrier sensor networks and Sariel Har-Peled for discussions that led to the subsequent algorithm.

I finally like to thank all past and present members of the Algorithms Reading Group at the University of Iowa. This was a great group to do research with over the past five years and the varied skills, viewpoints and methods that the members of this group brought with them has been of immense benefit to me in my research.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

CHAPTER 1
INTRODUCTION

This document contains work on several combinatorial optimization problems in a geometric setting. The problems considered arise from motivations in network design. In the design of networks we encounter several issues. For instance, one might wish to provide complete or maximum coverage for a set of clients via sensor networks or placing cell-phone towers. However installing too many stations is costly. In another setting one might wish to provide internet connectivity to a group of remote rural areas. However these areas are typically low-paying and system cost is an important consideration. In yet another scenario, one might wish to design a network of barrier sensors to detect intruders trying to cross over borders etc. Here too, one wishes to employ as few sensors as possible.

Each of the above problems can be formulated as a combinatorial optimization problem. For instance, the coverage problems can be formulated as versions of set cover [29, 19, 40, 49] - like a clustering problem or a facility location problem. In the clustering problem we are given a set of points in some general space and we want to group together "similar" or "nearby" points into a specified number of clusters so as to optimize some objective function. In the facility location problem one is concerned with optimal placement of facilities to serve a given set of clients so as to minimize the cost of opening facilities in addition to the cost of connecting clients to their nearest open facility. The internet connectivity problem can be formulated as a a problem of finding a minimum cost connected spanner over a graph where the villages form

the vertex set and point to point links form the edge set. The problem of designing a network of barriers that detect motion of intruders has a flavor of a "cut" problem where we have to cut off all possible paths while minimizing the number of sensors used.

To begin, we describe the combinatorial problems considered in this document in their original context and then consider them in a geometric setting.

Set Cover. Let us start with the Set cover problem which has been an extremely well-studied combinatorial optimization problem [29, 19, 40, 49]. Also several important combinatorial problems can be derived from or are special cases of this problem. These include clustering, dominating set, facility location problems, cut problems etc. In the set cover problem, we are given a set of objects X and a set $\mathcal{F}$ of subsets of X. The goal is to pick some subset of $\mathcal{F}$ that satisfies some constraints and optimizes some objective value. For example, one might want to pick the smallest subset of $\mathcal{F}$ that contains (covers) all objects of X. Or perhaps there is a weight assigned to each element in $\mathcal{F}$ and the goal is to find a subset that covers everything in X such that the sum of the weights of the elements in the subset is minimized.

In 1974, Johnson showed that a greedy algorithm for the set cover gives a logarithmic approximation [40]. That is, the algorithm will return a solution which is at most an $O(\log n)$ factor worse than an optimal solution where n is the size of the input. In 1993, Lund and Yannakakis [50] and Bellare, Goldwasser, Lund and Russell [8] showed that there is a positive constant c such that the general set

cover problem cannot be approximated in polynomial time within a $c \log n$ factor unless $NP \subseteq DTIME(n^{O(\log \log n)})$. In 1997, Raz and Safra [58] showed that the problem cannot be approximated within a $c \log n$ factor for some constant unless $P = NP$. In 1998, Feige [29] showed that the problem cannot be approximated to within a $(1 - o(1)) \log n$ factor unless $NP \subseteq DTIME(n^{O(\log \log n)})$. Note that the lower bounds and the upper bounds are the same up to a constant factor, and thus the computational complexity of this problem is settled.

Clustering. Consider the clustering problem. Metric clustering to minimize the sizes of clusters has been a well-studied problem in general - see for instance [48, 11, 2, 17, 24]. Here we are given a positive integer k and a finite set of points with interpoint distances that satisfy the condition of being a "metric". A metric d is a function defined frpm $P \rightarrow P$ such that each of the following holds:

- $d(u, u) = 0$ for each $u \in P$
- $d(u, v) = d(v, u)$ for each $u, v \in P$
- $d(u, v) \leq d(u, z) + d(z, v)$ for each $u, v, z \in P$

Given a metric d defined on a set P of n points, we define the ball $B(v, r)$ centered at $v \in P$ and having radius $r \geq 0$ to be the set $\{q \in P | d(v, q) \leq r\}$. We want to cover all of the points in P using at most k balls, so that some given objective function of the sizes of the balls is optimized. For example one might want to minimize the sum of the radii of the k clusters. For simplicity, we call such a problem clustering

to minimize the sum of radii. In this work the cost of a set of balls is the sum of the radii of those balls.

Facility Location. In the Facility Location problem we are given a set of potential facility sites L where a facility can be opened and a set of demand points D that must be serviced. We are also given a distance function $d : L \times D \to \mathcal{R}_+$ that indicates the cost of connecting a facility to a client and a cost function $f : F \to \mathcal{R}_+$ that indicates the opening cost of the facility. The goal is to pick a subset $F \subseteq L$ of facilities to be opened and an assignment $C : D \to F$ of clients to facilities that minimizes the total cost $\sum_{i \in F} f(i) + \sum_{i \in D} d(i, C(i))$.

In 1982, Hochbaum showed that a greedy algorithm for Facility Location gives an $O(\log n)$ approximation where n is the the number of locations. This factor is tight via an approximation-preserving reduction from the set cover problem [29]. The Facility Location Problem is NP-complete and MAX-SNP hard [35]. In the Metric Facility Location problem the connection costs satisfy the condition of being a metric (as defined above). For this problem, a series of polynomial time algorithms that return a solution at most a constant factor times worse was provided. See [62, 35, 46, 18, 38, 16, 37, 51, 63, 13]. Several approaches such as greedy heuristics, primal-dual approach, linear programming with rounding and greedy local search have been tried towards improving the approximation factor for this problem.

Network Spanner Topology Design. In certain types of networks there is a requirement of line of sight connectivity between antennas communicating with each

other - i.e. the line connecting the two antennas should clear all obstacles in the path. The obstacles in general may be in the form of trees, buildings and terrain. For this purpose, antennas are placed on top of towers. The required height of the towers depends on the positions and the heights of the obstacles along the link. The cost of a tower depends upon its height.

Let V be the set of nodes in a network. A *height (assignment) function* h gives an assignment of tower heights to every node in V. A pair of nodes (i, j) can see each other under a height function h if tower heights $h(i)$ at i and $h(j)$ at j are sufficient to achieve Line-of-Sight between i and j. i.e. the line joining the antennas mounted on the towers should clear any obstacles along the path. A pair of nodes (i, j) can form a *direct link* if they can see each other.

Let $COVER(h)$ denote the set of pairs of nodes that can form direct links under the height function h. Given a tower height, a *cost function* c, gives the cost of building a tower of that height. The *total cost* of a height function h is $\sum_{j \in V} c(h(v))$. Among all height functions h such that $COVER(h)$ results in a connected spanner (graph) on the vertex set V, our goal is to find the height function with the minimum total cost.

We show that this problem can be reduced to minimum Node Weighted Steiner Tree problem. A Steiner Tree consists of "extra" vertices (and edges) introduced to decrease total cost of connection - these are called Steiner Vertices. The classical Steiner Tree Problem seeks a minimum-cost connected subgraph containing a specified set of vertices (called terminals). In the Node Weighted Steiner Tree Problem, the

cost function is defined on the vertices (in addition to the edges, possibly).

Set Cover can also be reduced to the Node Weighted Steiner Tree problem in general graphs [29, 58], so an approximation ratio better than $\log n$ is not achievable in polynomial time unless $P = NP$ where n is the number of vertices in the graph. Klein and Ravi [45] gave a polynomial time approximation algorithm with a performance ration of $O(\log n)$, so this ratio is within a constant factor of optimal.

Multiway Cut. In the multiway cut problem we are given an undirected graph $G = (V, E)$, a cost function c on the edges and a set of terminals $T \subseteq V$ with $|T| = k$ and we are required to find a set $E' \subseteq E$ such that the removal of E' from E disconnects each terminal from all others. The minimization function is the sum of the costs of the edges in E'.

The multiway cut problem is shown to be NP-hard when the number of points that need to be separated from each other is at least 3 [21] It was also shown in [21] that the problem is MAX-SNP hard and hence cannot have a polynomial time approximation scheme unless $P = NP$. Hence the best one can hope for is a constant factor approximation. A simple algorithm that achieved this was also given by the authors [21]. This factor was improved to $\frac{3}{2}$ by [20] and further to 1.3438 in [41].

Problems in a Geometric Setting. Most versions of all of the above problems are NP-hard in general. However, it is interesting to explore the corresponding problems when restricted to a geometric setting. We shall see that some of the problems admit approximation algorithms with better guarantees than in the general combinatorial

case in certain quite naturally occuring geometric settings. However for some other problems this is not the case and geometry does not really make the problem any easier. We study and evaluate the scenarios in which geometry can make a difference and try to understand why it cannot in the rest of the cases.

How well a geometric version of a combinatorial optimization problem can be approximated depends on the problem at hand. For certain problems, the best known approximation algorithm does no better than the one in the general case. However for several other problems we are able to do much better than in the general case.

Consider the set cover problem where the set X is a collection of points in some geometric space and the sets $\mathcal{F}$ are subsets of X induced by the intersection of X with some geometric object. For example, X might be a collection of points in the plane and each set in $\mathcal{F}$ might be the subset of points in X contained within some disk or triangle in the plane. For the case where the sets $\mathcal{F}$ are axis-aligned rectangles the best known approximation algorithm does not do better than in the general case - i.e. a guarantee of $\log n$. Some other problems admit approximation ratios better than $O(\log n)$ but are still superconstant. For example, Varadarajan [64] and Aronov, Ezra and Sharir [3] give an $O(\log \log \log n)$ approximation for set cover with "fat" triangles. Gibson, Kanade, Krohn and Varadarajan showed that the problem of guarding an x-monotone polygonal chain has a polynomial time approximation scheme (PTAS) [34]. That is, for any $\epsilon > 0$, their algorithm returns a solution in polynomial time whose size is at most a factor of $1+\epsilon$ worse than the size of an optimal solution. Very few hardness of approximation results are known. One recent hardness result due to

Har-Peled [36] is that there is no PTAS for set cover with fat triangles.

Also consider the facility location problem where the set of facility sites L and demand points D are points in some geometric space and the distance metric d corresponds to the Euclidean distance. For $d = 2$ and a positive real c, Arora, Raghavan and Rao [5] give an algorithm that achieves with a probability $1 - o(1)$ a solution of cost at most $1 + \frac{1}{c}$ times the optimal in time $n^{O(c+1)}$.

In the rest of this chapter, I discuss the problems named above in more detail constrained to the geometric setting.

1.1 Clustering to Minimize the Sum of Radii

Given metric d defined on a set V of points (a metric space), we define the ball $B(v, r)$ centered at $v \in V$ and having radius $r \geq 0$ to be the set $\{q \in V | d(v, q) \leq r\}$. Here we consider the problem of computing a minimum-cost k-cover for a given set $P \subseteq V$ of n points where $k > 0$ is some given integer which is also part of the input. For $\kappa \geq 0$, a κ-cover for subset $Q \subseteq P$ is a set of at most κ balls, each centered at a point in P whose union covers (contains) Q. The cost of a set $\mathcal{D}$ of balls, denoted $\text{cost}(\mathcal{D})$, is the sum of the radii of those balls.

In the metric version of the k-cover problem, we are given P and k and the distance d between every pair of points in P. In the Euclidean version, P is given as a set of points in some fixed dimensional Euclidean space R^l, and d is then the standard Euclidean distance.

1.1.1 Previous Work

Bilo et al [11] give polynomial time approximation schemes for generalizations of this problem where the cost of a set of balls is defined to be the sum of the α-th power of their radii, where $\alpha \geq 1$. They also show that the problem is NP-hard even in the plane for $\alpha \geq 2$. Alt et al. [2] show that the NP-hardness result can be extended to any $\alpha > 1$. They give fast constant factor approximation algorithms for this and related problems. We focus on the case where $\alpha = 1$.

Charikar and Panigrahy [17] give a poly-time algorithm based on the primal-dual method that gives a constant factor approximation for the same problem.

1.1.2 Our Contribution

We obtain an "exact" polynomial-time solution for the Euclidean k-clustering problem to minimize sum of radii under an assumption about the model of computation not unusual in such contexts. In the geometric min-cost k-cover problem, the optimal solution is a set of k disks, each of which is centered at some input point and each of which has a radius that is the distance between two input points. If the input points have integer co-ordinates, the cost of such a solution is the sum of square roots of integers. Our algorithm requires the ability to determine, given two such candidate solutions, the one that has lower cost. We make the assumption that this can be done in polynomial time.

Notice that for variants of the problem such as the one in which the underlying distance metric is the L_1 rather than the Euclidean metric, we can indeed compare the

costs of two candidate solutions in polynomial time. In the context of the Euclidean metric, however, it is a longstanding open problem as to whether two sums of square roots of integers can be compared in polynomial time [22, 55].

We therefore translate our exact algorithm for the Euclidean metric into an approximation algorithm that does not make the above assumption on the computational model. We obtain an approximation algorithm that for any parameter $0 < \epsilon < 1$ runs in polynomial time in the input size and $\log \frac{1}{\epsilon}$ and returns a solution whose cost is at most $(1 + \epsilon)$ times the optimal.

Theorem 1. *There is an algorithm that, given a set P of points in the plane, an integer $k \geq 1$ and a parameter $0 < \epsilon < 1$, runs in time polynomial in the input size and $\log \frac{1}{\epsilon}$, and returns a k-cover of P whose cost is at most $(1 + \epsilon)$ times the cost of the optimal k-cover.*

The result is enabled by a simple observation about the structure possessed by optimal solutions - they are eminently separable. To state this formally, we focus on the planar case. We find a "separator" that divides the region of interest (i.e. the region in which the points lie) roughly equally and intersects at most 12 disks in the optimal k-cover for P. Such a separator is easy to find because we can guess the disks that this separator intersects since there are only 12 of them. This decomposes the problem into two smaller subproblems. With some additional play dealing mainly with recursion and aspect ratio issues we are able to obtain a polynomial time algorithm.

In Chapter 2 we give an algorithm for the case where the points under con-

sideration lie in the plane. Thus we shall prove Theorem 1. This extends easily to higher dimensional spaces.

1.2 Network Spanner Topology Design

Let V be the set of villages or nodes in the network. Let $n = |V|$. A *height (assignment) function* h gives an assignment of tower heights to every node in V. A pair of nodes (i, j) can see each other under a height function h if tower heights $h(i)$ at i and $h(j)$ at j are sufficient to achieve Line-of-Sight between i and j. i.e. the line joining the antennas mounted on the towers should clear any obstacles along the path. The actual obstacle model is described later. Nodes i and j are said to be within transmission range of each other if they are close enough to be able to transmit signals while obeying constraints on maximum allowed transmit power i.e. $d(i, j) \leq B$ where B is the maximum transmission range, and $d(i, j)$ is the Euclidean distance between i and j. A pair of nodes (i, j) can form a *direct link* if they can see each other and are within transmission range of each other.

Let $COVER(h)$ denote the set of pairs of nodes that can form direct links under the height function h. Given a tower height, a *cost function* c, gives the cost of building a tower of that height. The *total cost* of a height function h is $\sum_{j \in V} c(h(v))$. Among all height functions h such that $COVER(h)$ results in a connected spanner (graph) on the vertex set V, our goal is to find the height function with the minimum total cost. In our problem the cost of each tower is its height - i.e. $\forall i \in V, c(h(i)) = h(i)$. (This is based on empirical studies and making useful assumptions which we

describe later.)

1.2.1 Previous Work

A version of this problem was considered as a subproblem in topology planning for rural wireless mesh networks by Sen and Raman [61]. They describe a heuristic for the problem where a certain number of spanning trees are considered, and for each spanning tree, the optimal height assignment to build the edges in the tree is determined by solving a linear program. However, this is an extremely expensive process because of the number of trees considered. Panigrahi et al. [54] show that in a graph theoretic formulation where the set of villages V is the vertex set of the graph and the set of edges is the set of all candidate point to point links the problem of finding a height assignment h to V where cover $COVER(h)$ is a minimum cost connecting spanner is NP-hard and a better than $O(\log n)$ factor approximation algorithm cannot be expected. They also go on to present a greedy $O(\log n)$ factor approximation algorithm for the problem.

1.2.2 Network Spanner Topology Design in the Plane

Consider the geometric setting where all the nodes (vertices) V are points in the plane. Also there are certain assumptions made about the placement and heights of obstacles. In particular for each pair of villages (i, j) we place obstacles of height L on segment $\bar{ij}$ at a distance of d from i and at a distance d from j. We assume that the distance between any 2 villages is at least $2d$. This setting is motivated from real life scenarios in remote rural areas.

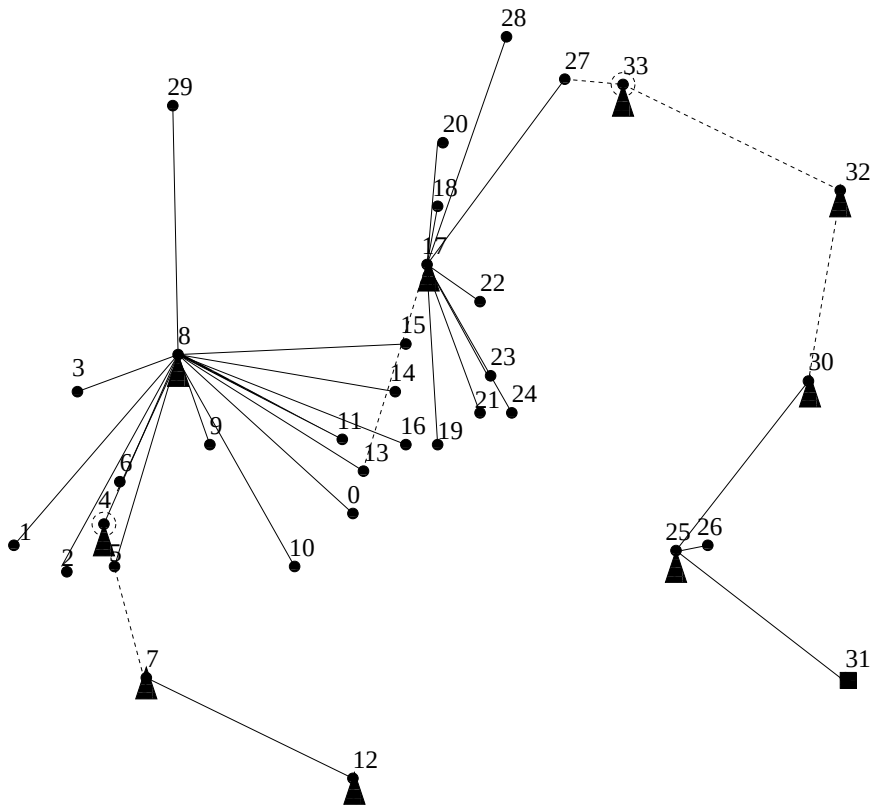


Figure 1.1: What a rural network topology may look like

1.2.2.1 An Example

First, consider an example for the plane setting. Figure 1.1 shows an example of a rural network setting. The dots are the nodes (villages), the triangles represent towers and the edges represent point-to-point links. Figure 1.2 describes the obstacle model for this setting.

1.2.2.2 Our Contribution.

We exploit the underlying geometry in the setting to obtain efficient approximation algorithms. We apply tools developed in the context of geometric set cover and facility location problems to this setting. We show that the geometric version of

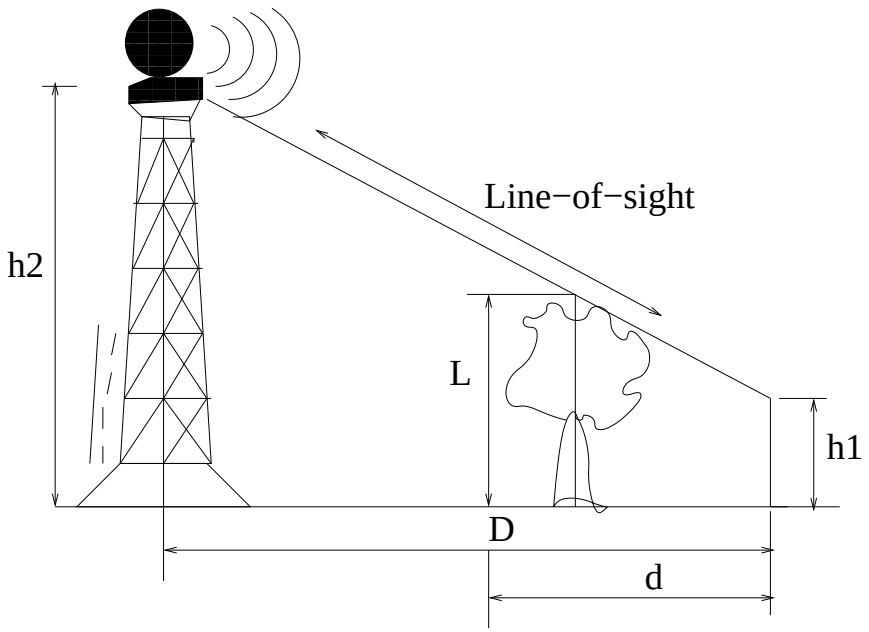


Figure 1.2: The Obstacle Model

this problem admits a constant factor approximation - i.e. we obtain in polynomial time a solution whose cost is at most a constant factor times the optimal. The result is in contrast to the general graph-theoretic formulation where we cannot hope to obtain a better than $O(\log n)$ approximation [54]. We transform the problem into a two stage problem - the first being a version of a facility location problem and in the second stage we ensure the connectivity of the solution. In a sense taller towers are facilities that serve villages with shorter towers in their surroundings and connect them to other similar cluster like structures in the topology. We use a linear programming based primal dual algorithm. In the second stage, we ensure that these clusters are connected. Our simulation results validate this two stage approach. This algorithm is efficient and amenable to practical implementation.

Theorem 2. *Given a set of nodes V in the plane, maximum transmission power range B, and an obstacle model as defined with parameters L (obstacle height) and d (obstacle clearance) we find a height assignment function h that assigns a non-negative height value to a tower at each node in V such that $COVER(h)$ results in a connected spanner on V and the total height $\sum_{j \in V} h(j)$ is within a constant factor of the total height in the optimal solution.*

1.2.3 Network Spanner Topology Design on the Terrain

A 1.5-D terrain is a polygonal chain in the plane that is x-monotone, that is, any vertical line intersects the chain at most once. A terrain T consists of a set of m vertices $\{v_1, v_2, \cdots, v_m\}$. The vertices are ordered in increasing order with respect to their x-coordinates. There is an edge connecting v_i with v_{i+i} for all $i = 1, 2, \cdots m-1$. For any 2 points a, b on or above the terrain T, we say that a *sees* b if the line segment $\bar{ab}$ lies entirely above or on the terrain.

The villages (nodes) V in our topology are a subset of the points on the terrain T. Thus we consider only discrete points on the terrain. Let $|V| = n$. A height assignment function h gives an assignment of tower heights to every node in V. A pair of nodes (i, j) in V can see each other under h if tower heights $h(i)$ and $h(j)$ are such that the tops of the towers see each other. This setting is interesting because of the general importance of the 1.5-D terrain in computational geometry.

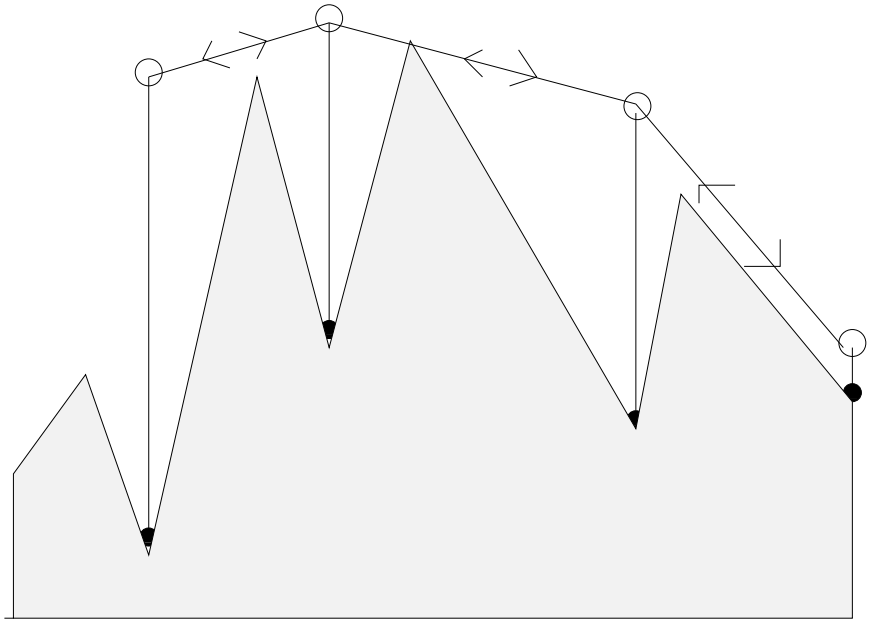


Figure 1.3: Example of Network on Terrain

1.2.3.1 Example.

Figure 1.3 shows an example of this. The dots are the nodes (villages) located in valleys here. The peaks are the obstacles. The straight lines with antennas on top are the towers constructed that enable line of sight visibility and hence this topology is connected.

1.2.3.2 Our Contribution

On the 1.5-dimensional terrain the reduction from the Node Weighted Steiner Tree problem immediately implies a logarithmic approximation. We explore various approaches to obtain a constant factor approximation and talk about why this problem might be hard. We describe different approaches used to tackle this problem - such as greedy algorithm, primal dual algorithm and dynamic programming and the

issues arising in each of these.

In Chapter 3, we will cover the problem of Network Design in the plane. We will explain how we formulate it as a geometric facility location problem, provide a primal dual approximation algorithm and also some empirical results. In Chapter 4 we will consider the problem of network design on a terrain.

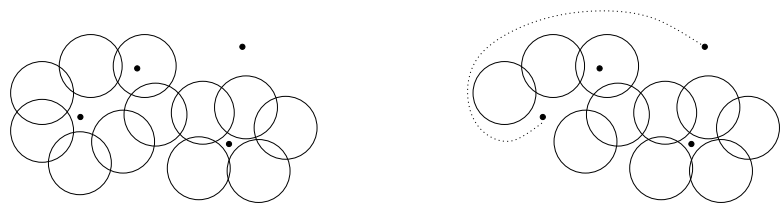
1.3 Multiway Barrier Cut

Let $P = \{p_1, p_2, \cdots p_k\}$ be a set of k points in the plane that need to be separated in some sense (perhaps there are intruders present at each of these spots and they need to be stopped from meeting each other). Let I denote a given set of sensors in the form of unit disks in the plane such that I separates P that is every path in the plane between any pair of points $p_i, p_j \in P$ intersects at least one of the disks in I. We consider the problem of finding a minimum cardinality subset of I that separates P. Refer figure 1.4 for an example.

Here we need to cut off all possible paths in the plane between any pair of points in P. Although there are infinitely many paths it is enough to consider a representative set of these that we need to cut. Due to this idea of cutting off all possible paths between every pair of points in P this problem has a flavor of the well-studied multiway cut problem.

1.3.1 Related Work

Until recently most work related to coverage in wireless sensor networks focussed on blanket coverage. Barrier coverage involves use of barriers as defense mech-



(a) (b)

Figure 1.4: Example of a multiway barrier cut. (a) This set of disks separates the points because every path connecting any two points must intersect a disk. (b) This set of disks does not separate the points.

anism against intruders [47] and trap coverage ensures that an object cannot travel too far before being detected [7, 60].

Sankararaman et. al [60] consider the related problem of "weak" coverage. In their problem, the region under consideration is completely covered by the given set of unit disk sensors and this set is to be partitioned into as many subsets as possible such that for each subset, the diameter of any cell of the region not covered is bounded by some k. Their work differs in flavor for ours mainly due to the assumption that the input set of disks completely covers the region under consideration.

1.3.2 Our Contributions

To the best of our knowledge we are the first to formulate this as an optimization problem. We provide a polynomial time constant factor approximation algorithm.

Theorem 3. *Let I be a set of n unit disks and P a set of k points such that I separates P. There is a polynomial time algorithm that takes as input such I and P,*

and returns a subset $O \subseteq I$ of disks that also separates P, with the guarantee that $|O|$ is within a multiplicative $O(1)$ of the smallest subset of I that separates P.

Our algorithm is simple and combinatorial and is in fact a greedy algorithm. We first present an $O(1)$-approximation algorithm for the following two-point separation problem: given a set of unit disks G, and two points s and t, and we wish to find the smallest subset $B \subseteq G$ so that B separates s and t.

Our greedy algorithm to the overall problem applies the two-point separation algorithm to find the cheapest subset B of I that separates some pair of points in P. Suppose that P is partitioned into sets $P_1, P_2, \ldots, P_\tau$ where each P_i is the subset of points in the same "face" with respect to B. The algorithm then recursively finds a separator for each of the P_i, and returns the union of these and B.

In Chapter 5, we describe the problem formulation, our greedy algorithm, and the analysis exploits the geometry of the setting to give a constant factor approximation. Our approach immediately generalizes if the input set of disks have arbitrary radii. This is because we do not make use of any specific geometric properties of unit disks, but only of disks in general.

CHAPTER 2
CLUSTERING TO MINIMIZE SUM OF RADII

Given a metric d defined on a set V of points (a metric space), we define the ball $\mathrm{B}(v, r)$ centered at $v \in V$ and having radius $r \geq 0$ to be the set $\{q \in V | d(v, q) \leq r\}$. In this work, we consider the problem of computing a minimum cost k-cover for a given set $P \subseteq V$ of n points, where $k > 0$ is some given integer which is also part of the input. For $\kappa \geq 0$, a κ-cover for subset $Q \subseteq P$ is a set of at most κ balls, each centered at a point in P, whose union covers (contains) Q. The cost of a set $\mathcal{D}$ of balls, denoted $\mathrm{cost}(\mathcal{D})$, is the sum of the radii of those balls.

Recall Theorem 1 which states that there is an algorithm that, given a set P of points in the plane, an integer $k \geq 1$ and a parameter $0 < \epsilon < 1$, runs in time polynomial in the input size and $\log \frac{1}{\epsilon}$, and returns a k-cover of P whose cost is at most $(1 + \epsilon)$ times the cost of the optimal k-cover. The result is enabled by a simple observation about the structure possessed by optimal solutions – they are eminently separable. To state this formally, we focus on the planar case and define the *length* of a rectangle R as the length of its longer side. If the rectangle is a square, the longer side is defined arbitrarily. The *width* of the rectangle is the length of its shorter side. A *separator* for R is any line s that is perpendicular to the longer side (length) of R, intersects R, and whose distance from each of the two shorter sides of R is at least a third of the length of R.

Lemma 4. *Consider an optimal κ-cover $\mathcal{D}$ for some set $Q \subseteq P$ of points contained*

in a rectangle R. The rectangle R has a separator that intersects at most 12 *disks in* $\mathcal{D}$.

Proof. Let l denote the length of R. Choose a separator uniformly at random from the set of separators for R. The probability that it intersects a disk $D \in \mathcal{D}$ is bounded by $\frac{2 \cdot \text{radius}(D)}{l/3}$. It follows that the expected number of disks in $\mathcal{D}$ intersected by a separator is at most $\frac{6}{l} \sum_{D \in \mathcal{D}} \text{radius}(D) = \frac{6 \cdot \text{cost}(\mathcal{D})}{l}$. Now $\text{cost}(\mathcal{D})$, the sum of the radii of disks in $\mathcal{D}$, is at most $2l$. This is because one disk of radius $2l$ centered at any point in Q covers Q. We conclude that the expected number of disks in $\mathcal{D}$ intersected by a random separator is at most 12.

□

This observation opens up the possibility of computing an optimal k-cover efficiently via dynamic programming, as exemplified by [4, 52]. Some additional play is however needed to achieve a polynomial time algorithm – we have to deal with the accumulation of the number of disks we need to guess in a recursive application of Lemma 4, and also with some complexity that arises because the aspect ratio of the input point set P is not assumed to be bounded by a polynomial in n.

In Section 2.1, we establish basic observations used subsequently. In Section 2.2, we present the polynomial time algorithm for the min-cost k-cover problem in the plane. The extensions to any fixed dimensional Euclidean space and to related problems such as MSRC are straightforward and are omitted here.

2.1 Preliminaries

The following lemma states a structural property of the optimal solution in the planar case. It is similar to observations made by Lev-Tov and Peleg [48] and its proof is sketched here for completeness.

Lemma 5. *Let OPT denote an optimal k-cover for $P \subseteq \Re^2$. Let R be a rectangle of length a. The number of disks in OPT of radius at least a that intersect R is bounded by a constant.*

Proof. Let $\mathrm{OPT}' \subseteq \mathrm{OPT}$ be the disks whose radius is at least $a > 0$. The rectangle R can be enclosed by a disk B of radius a. We will show that the number of disks in OPT' that intersect B is $O(1)$.

A basic property of the optimal solution OPT is its *admissibility*, the fact that no disk in OPT contains in its interior the center of another disk in OPT. This implies the the centers of any two disks in OPT' are at least a apart. It follows that there are $O(1)$ disks in OPT' whose centers lie within the disk B' concentric with B and with radius $10a$.

To bound the number of disks in OPT' with center outside B' that also intersect B, we partition the plane radially into 8 sectors with angle $\pi/4$ and centered at the center of B. (See Figure 2.1.) It is an easy consequence of admissibility that for each sector there can be at most one disk in OPT (and thus also OPT') that is centered in the sector outside B' and that intersects B.

□

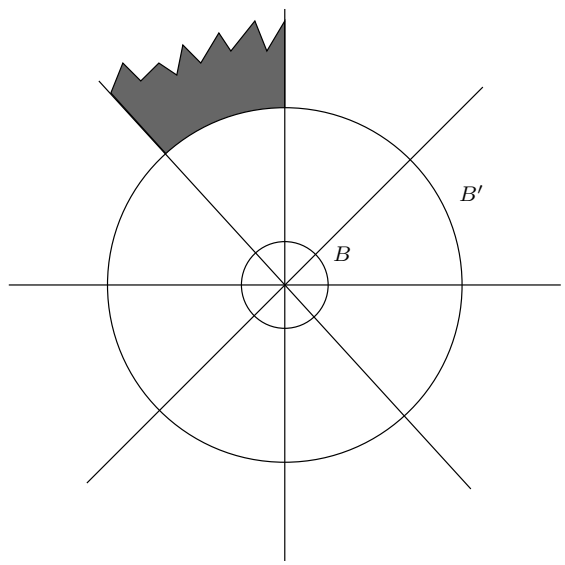


Figure 2.1: At most one disk from OPT can have its center in the shaded area and intersect B.

2.2 The Algorithm in the Planar Case

We describe a polynomial time algorithm to compute an optimal k-cover for a set P of n points in the plane. We will assume that $k \leq n$, since otherwise the optimal k-cover is trivially computed. We start by observing that there is an optimal k-cover of P whose disks are chosen from the set $\mathcal{D}$ of disks whose center is some $p \in \mathcal{D}$ and whose radius is $|pq|$ for some $q \in P$. Note that $|\mathcal{D}| = n^2$. In the rest of the article we will reserve $\mathcal{D}$ to denote this set of disks.

Canonical and Critical lines. We say that a vertical (resp. horizontal) line is *critical* if it passes through a point in P or a point of vertical (resp. horizontal) tangency of some disk in $\mathcal{D}$. (See Figure 2.2.) Note that if l and l' are two vertical (resp. horizontal) noncritical lines with no vertical (resp. horizontal) critical line

between them, then l and l' intersect the same set of disks in $\mathcal{D}$. Motivated by this, we define a set of $\Theta(n^2)$ *canonical* vertical lines that includes any one vertical line between every two consecutive vertical critical lines, one vertical line to the left of the leftmost critical line and one vertical line to the right of the rightmost critical line. We define a set of $\Theta(n^2)$ canonical horizontal lines analogously.

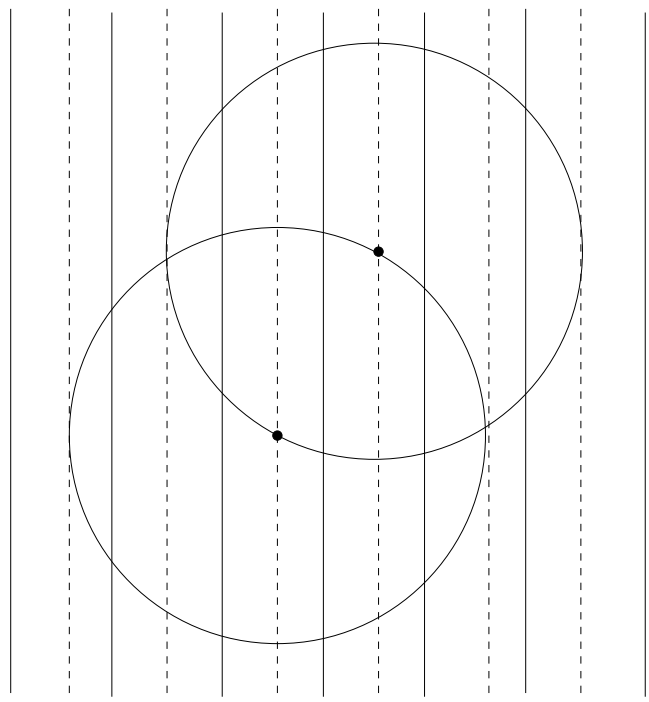


Figure 2.2: Critical (dashed) and canonical (solid) lines.

Balanced Rectangles and Compression. A rectangle is said to be *balanced* if its width is at least a third of its length. We describe a procedure compress(R) that takes as input a balanced rectangle R and returns a balanced rectangle R' such that (a) R' is contained in R, (b) R' contains $P \cap R$, and (c) for any separator for R', there

are points of $P \cap R$ in both the open halfspaces that it bounds (and consequently, any separator for R partitions $P \cap R$ into two nonempty subsets).

To describe the procedure compress(R), we assume without loss of generality that the separators of R are vertical (its length is horizontal). The procedure first checks if there is a point in $P \cap R$ that is strictly to the left of the leftmost separator of R, and if there is a point in $P \cap R$ that is strictly to the right of the rightmost separator of R. If so, the procedure returns $R' = R$.

Otherwise, let R' be the minimal rectangle enclosing $P \cap R$. Let l' be its length and w' its width. Let l and w denote the length and width of R, respectively. If $w' \geq l'/3$, we are done and we simply return R'.

If $w' < l'/3$, we have to make R' wider while still keeping it enclosed in R. There are two cases to consider, depending on whether the longer side of R' is parallel to the longer side of R. If the longer side of R' is parallel to the shorter side of R, we know that $l' \leq w$. We also know that $l \geq w \geq l/3$. The width of R' needs to be expanded to $l'/3$. Since $l'/3 \leq w/3 \leq l/3$, there is enough room for this expansion.

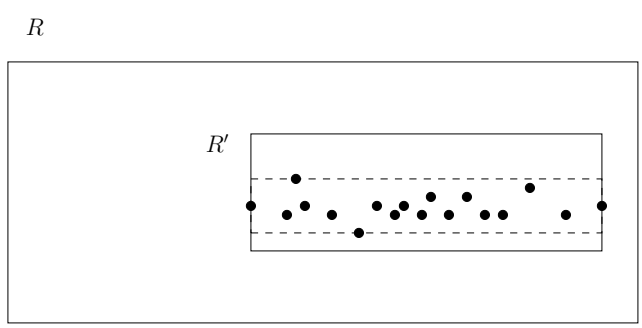


Figure 2.3: R and $R' =$ compress(R). (Dashed is R' before expansion.)

The second case involves the longer side of R' being parallel to the longer side of R. (See Figure 2.3.) In this case, it must be that $l' \leq 2l/3$. The width of R' needs to be expanded to $l'/3$. Since $l'/3 \leq 2l/9 \leq l/3 \leq w$, there is enough room for this expansion.

The Algorithm. We describe a procedure $\texttt{DC}(R, \kappa, T)$ that takes as input a balanced rectangle R, an integer $\kappa \geq 0$, and a subset $T \subseteq \mathcal{D}$. It returns a κ cover of the set

$$Q = \{q \in P \cap R \mid q \text{ is not covered by } T\}.$$

We will later argue that $\texttt{DC}(S, k, \emptyset)$ computes an optimal k-cover for P when S is any balanced rectangle containing P.

The result of a call to procedure $\texttt{DC}(R, \kappa, T)$ is stored in a table entry indexed by $P \cap R, \kappa, T$; the entry Table$(P \cap R, \kappa, T)$ stores a κ-cover for the set Q defined above. It will be useful for the description of the algorithm to define a special "disk" I whose cost is ∞. If an entry Table$(P \cap R, \kappa, T)$ stores a set that contains I, this means that the algorithm has determined that there is no κ-cover for Q.

Here we define the procedure $\texttt{DC}(R, \kappa, T)$

if Table$(P \cap R, \kappa, T)$ has already been created **then**

  return

else

  Create table entry Table$(P \cap R, \kappa, T)$ which will be assigned below.

end if

Let $Q = \{q \in P \cap R \mid q \text{ is not covered by } T\}$.

if $Q = \emptyset$ **then**

let Table$(P \cap R, \kappa, T) \leftarrow \emptyset$,

return

end if

if $\kappa = 0$ **then**

let Table$(P \cap R, \kappa, T) \leftarrow \{I\}$

return

end if

if $|Q| = 1$ **then**

assign to Table$(P \cap R, \kappa, T)$ the set containing the trivial disk consisting of the one point in Q,

return

end if

if $P \cap R$ has at least two points in it **then**

Call compress(R) to obtain a balanced rectangle R' containing $P \cap R$. Let us assume for the purposes of exposition that the separators for R' are vertical. Let $L(R')$ be the set consisting of those vertical canonical lines that are separators for R'. Also include in $L(R')$ the leftmost (resp. rightmost) separator for R' provided it is not critical. Initialize a cover $\mathcal{D}' \leftarrow \{I\}$.

for all choice of a separator $l \in L(R')$ **do**

for all choice of a set $\mathcal{D}_0 \subseteq \mathcal{D}$ of at most 12 disks that intersects l **do**

for all choice of $\kappa_1, \kappa_2 \geq 0$ such that $\kappa_1 + \kappa_2 + |\mathcal{D}_0| \leq \kappa$ **do**

Let R_1 and R_2 be two rectangles into which l partitions R'. Let $T_1 = \{D \in T \cup \mathcal{D}_0 \mid D \text{ intersects } R_1\}$. Let $T_2 = \{D \in T \cup \mathcal{D}_0 \mid D \text{ intersects } R_1\}$.

if $|T_1| \leq \beta$ and $|T_2| \leq \beta$ (β is a large enough constant) **then**

$\texttt{DC}(R_1, \kappa_1, T_1)$ and $\texttt{DC}(R_2, \kappa_2, T_2)$

if $\text{cost}(\mathcal{D}_0 \cup \text{Table}(P \cap R_1, \kappa_1, T_1) \cup \text{Table}(P \cap R_2, \kappa_2, T_2)) < \text{cost}(\mathcal{D}')$

then

update $\mathcal{D}' \leftarrow \mathcal{D}_0 \cup \text{Table}(P \cap R_1, \kappa_1, T_1) \cup \text{Table}(P \cap R_2, \kappa_2, T_2)$.

end if

Assign $\text{Table}(P \cap R, \kappa, T) \leftarrow \mathcal{D}'$

return

end if

end for

end for

end for

end if

See Algorithm 2.2 for the description of the procedure $\texttt{DC}(R, \kappa, T)$.

A couple of remarks about the algorithm are in order. Observe that partitioning a balanced rectangle by a separator results in two balanced rectangles. It follows that R_1 and R_2 are balanced given that R' is balanced.

No separator that the algorithm considers on step 6 passes through a point in

P. We therefore do not have to worry about breaking ties because input points land on a separator.

Suppose that the procedure $\texttt{DC}(R, \kappa, T)$ finds that in step 1, the entry Table($P \cap R, \kappa, T$) has not been created. It then proceeds to create this entry. In between the time this entry is created and subsequently filled in by this procedure, no subproblem will need the entry Table($P \cap R, \kappa, T$). This is because for any call $\texttt{DC}(\hat{R}, \hat{\kappa}, \hat{T})$ that is nested within $\texttt{DC}(R, \kappa, T)$, $P \cap \hat{R}$ is a strict subset of $P \cap R$. Such a call $\texttt{DC}(\hat{R}, \hat{\kappa}, \hat{T})$ will not enquire as to whether the entry Table($P \cap R, \kappa, T$) has been created, nor will it seek to know the content of this entry.

Running Time. We now bound the overall running time of a call to $\texttt{DC}(S, k, \emptyset)$, where S is some balanced rectangle containing P. Note that each table entry is indexed by a set of points $P \cap R$ for some balanced rectangle R, a $\kappa \leq k$, and a set $T \subseteq \mathcal{D}$ such that $|T| \leq \beta$. The number of such $P \cap R$ is $O(n^4)$, the number of such κ is $O(n)$ and the number of such T is $O(n^{2\beta})$. Assuming β is a constant, the number of table entries is therefore bounded by a polynomial in n. We use this to bound the total number of calls $\texttt{DC}(R, \kappa, T)$ made.

The number of calls of the form $\texttt{DC}(R, \kappa, T)$ that create a corresponding table entry Table($P \cap R, \kappa, T$) is polynomially bounded, since the number of table entries is polynomially bounded and each table entry is created only once. Any call of the form $\texttt{DC}(R, \kappa, T)$ that does not create the table entry Table($P \cap R, \kappa, T$) does not make any recursive calls and takes $O(1)$ time. We charge this to the parent instance of

$\texttt{DC}()$ that makes the recursive call $\texttt{DC}(R, \kappa, T)$. Note that this parent instance creates its corresponding table entry. Since any instance of $\texttt{DC}()$ makes only a polynomial number of direct recursive calls, we can conclude using our charging argument that the number of calls of the form $\texttt{DC}(R, \kappa, T)$ that do not create a corresponding table entry $\text{Table}(P \cap R, \kappa, T)$ is also bounded by a polynomial.

Since any instance of $\texttt{DC}()$ takes polynomial time in n not counting the time for the recursive calls, we can conclude that the overall running time of $\texttt{DC}(S, k, \emptyset)$ is bounded by a polynomial in n.

Correctness. Establishing that $\texttt{DC}(S, k, \emptyset)$ returns an optimal k-cover of P is more involved than showing that it runs in polynomial running time, mainly because of the pruning step that ensures that instance $\texttt{DC}(R, \kappa, T)$ that is called has $|T|$ bounded by β. We begin by showing the following fact about the algorithm.

Lemma 6. *Let $\texttt{DC}(R_1, \kappa_1, T_1)$ be a recursive call made inside a sequence of nested recursive calls that involved $\texttt{DC}(G_1, k, \emptyset), \texttt{DC}(G_2, \cdot, \cdot), \ldots, \texttt{DC}(G_t, \cdot, \cdot)$, where $G_1 = S$. Let G'_j be the result obtained by a call to compress(G_j). Let $l(G'_j)$ be the separator chosen for G'_j so that for each $1 \leq j \leq t-1$, G_{j+1} is one of the two rectangles into which $l(G'_j)$ partitions G'_j, and R_1 is one of the two rectangles into which $l(G'_t)$ partitions G'_t. Denote by a the length of R_1. Then the horizontal distance between any two distinct vertical separators $l(G'_i)$ and $l(G'_j)$ is at least $a/3$, and the vertical distance between any two distinct horizontal separators $l(G'_i)$ and $l(G'_j)$ is at least $a/3$.*

Proof. Since $R_1 \subseteq G'_t \subseteq G_t \subseteq G'_{t-1} \subseteq G_{t-1} \subseteq \cdots \subseteq G'_1 \subseteq G_1$, the length of any G'_i is

at least a. Let G'_i and G'_j be such that $i < j$ and $l(G'_i)$ and $l(G'_j)$ are vertical. Note that G'_j is contained in one of the two rectangles into which $l(G'_i)$ partitions G'_i. Thus $l(G'_i)$ is either to the left of G'_j or to its right. Now $l(G'_j)$ lies between the two vertical sides of G'_j at a distance of at least a third of the length of G'_j from either side. Since the length of G'_j is at least a, it follows that the horizontal distance between $l(G'_i)$ and $l(G'_j)$ is at least $a/3$. (See Figure 2.4.)

The case where $l(G'_i)$ and $l(G'_j)$ are horizontal is reasoned similarly.

□

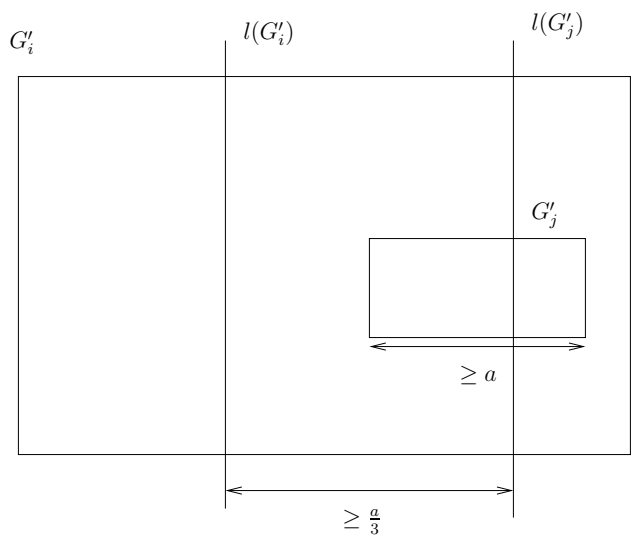


Figure 2.4: Illustration for Lemma 6

The correctness of the algorithm follows from the following lemma; let us fix an optimal k-cover OPT $\subseteq \mathcal{D}$ of P.

Lemma 7. *Suppose the recursive instance* $\texttt{DC}(R, \kappa, T)$ *is called (at some recursion*

depth) by the top-level invocation to $\texttt{DC}(S, k, \emptyset)$. *Suppose also that* $T \subseteq OPT$ *and* T *contains every disk in* OPT *that contains a point in* $P \cap R$ *as well as a point in* $P \setminus R$. *(*T *possibly contains other disks in* OPT *as well.) Let* $Q = \{q \in P \cap R \mid q$ *is not covered by* $T\}$ *and let* OPT' *denote the set of disks in* OPT *that contain points in* Q. *Suppose further that* $\kappa \geq OPT'$. . *Then after the call to* $\texttt{DC}(R, \kappa, T)$ *completes, Table*$(P \cap R, \kappa, T)$ *contains a* κ*-cover for* Q *whose cost is at most cost(*OPT'*).*

Proof. The proof is by induction on $|P \cap R|$. We may assume that Table$(P \cap R, \kappa, T)$ has not been created when $\texttt{DC}(R, \kappa, T)$ is called. For otherwise, this table entry was created by a call $\texttt{DC}(\hat{R}, \kappa, T)$ for some rectangle $\hat{R}$ such that $\hat{R} \cap P = R \cap P$. We check that the suppositions made for T and κ hold with $\hat{R}$ in the place of R, and Q and OPT$'$ are the same in both cases. And we can use the arguments below to conclude that after the call to $\texttt{DC}(\hat{R}, \kappa, T)$ completes, Table$(P \cap R = P \cap \hat{R}, \kappa, T)$ contains a κ-cover for Q whose cost is at most cost(OPT$'$). The subsequent call to $\texttt{DC}(R, \kappa, T)$ returns immediately without changing Table$(P \cap R = P \cap \hat{R}, \kappa, T)$, which therefore continues to contain a κ-cover for Q whose cost is at most cost(OPT$'$).

The cases where the call to $\texttt{DC}(R, \kappa, T)$ is returned via steps 2 or 4 form the base cases of the induction, and are easily established. Note that the call cannot return via step 3, because if $Q \neq \emptyset$, then $\kappa \geq |\text{OPT}'|$ must be at least 1.

The inductive case is when the call returns in step 13. Note that $|P \cap R|, |Q| \geq 2$ in this case. We will assume without loss of generality that the separators of R' are vertical.

We first observe that OPT$'$ is an optimal $|\text{OPT}'|$-cover for Q. This is because

points in $(P \cap R) \setminus Q$ are covered by T, and OPT' does not cover any point in $P \setminus R$ – so the only role that OPT' plays is to cover Q, and it therefore must do this optimally. From Lemma 4, we conclude that R' has a separator l' that interesects at most 12 disks in OPT'. Let $\mathcal{D}_0$ denote this set. Since the set of critical lines is finite, the argument of Lemma 4 tells us that we can assume l' to be line that is not critical. It is easy to see that we can move l' to one of the lines in $L(R')$ (computed in step 5) without changing the set of disks in $\mathcal{D}$ that it intersects. We therefore consider the choice of a separator $l \in L(R')$ whose intersection with OPT' is exactly $\mathcal{D}_0$. Consider the body of the innermost for loop where l and $\mathcal{D}_0$ have been chosen as above and κ_1 is set to be the number of disks in OPT' to the left of l and κ_2 is set to be the number of disks in OPT' to the right of l. Let OPT_1 and OPT_2 denote the disks in OPT' to the left and right of l, respectively. Of course, $|\mathrm{OPT}_1| = \kappa_1$ and $|\mathrm{OPT}_2| = \kappa_2$.

Let R_1, R_2, T_1, and T_2 be exactly as in the algorithm. Note that $|P \cap R_1| < |P \cap R|$, and $|P \cap R_2| < |P \cap R|$. Notice further that $T_1 \subseteq \mathrm{OPT}$ and T_1 contains every disk in OPT that contains a point in $P \cap R_1$ as well as a point in $P \setminus R_1$. Furthermore, OPT_1 is the set of disks in OPT that contain points in $Q_1 = \{q \in P \cap R_1 \mid q \text{ is not covered by } T_1\}$ and $k_1 = |\mathrm{OPT}_1|$. We can invoke the induction hypothesis to claim that *if the call* $\mathtt{DC}(R_1, \kappa_1, T_1)$ *is made*, then after the call, $\mathrm{Table}(R_1 \cap P, \kappa_1, T_1)$ contains a κ_1-cover of Q_1 whose cost is at most $\mathrm{cost}(\mathrm{OPT}_1)$. Similarly, we can invoke the induction hypothesis to claim that if the call $\mathtt{DC}(R_2, \kappa_2, T_2)$ is made, then after the call, $\mathrm{Table}(R_2 \cap P, \kappa_2, T_2)$ contains a κ_2-cover of $Q_2 = \{q \in P \cap R_2 \mid q \text{ is not covered by } T_2\}$ whose cost is at most $\mathrm{cost}(\mathrm{OPT}_2)$. We

will show that $|T_1|$ and $|T_2|$ are bounded by β and that therefore these calls are indeed made. It follows that in step 12, $\mathcal{D}_0 \cup \text{Table}(P \cap R_1, \kappa_1, T_1) \cup \text{Table}(P \cap R_2, \kappa_2, T_2)$ is a κ-cover for Q whose cost is at most $\text{cost}(\mathcal{D}_0) + \text{cost}(\text{OPT}_1) + \text{cost}(\text{OPT}_2) = \text{cost}(\text{OPT}')$. Thus in step 13 $\text{Table}(P \cap R, \kappa, T)$ is assigned a κ-cover of Q with cost at most $\text{cost}(\text{OPT}')$.

To complete the proof we show that $|T_1| \leq \beta$, where β is a sufficiently large constant. The proof for T_2 is similar.

Suppose that we arrived at the (R, κ, T) instance by proceeding through a sequence of nested recursive calls that involved $\texttt{DC}(G_1, k, \emptyset), \texttt{DC}(G_2, \cdot, \cdot), \ldots, \texttt{DC}(G_{t-1}, \cdot, \cdot)$, where $G_1 = S$. Let $G_t = R$. Let G'_j be the result obtained by a call to $\text{compress}(G_j)$. Let $l(G'_j)$ be the separator chosen for G'_j so that for each $1 \leq j \leq t-1$, G_{j+1} is one of the two rectangles into which $l(G'_j)$ partitions G'_j. Finally, let $l(G'_t) = l$ and note that $G'_t = R'$. Thus R_1 is one of the two rectangles into which $l(G'_t)$ partitions G'_t. Furthermore let $\mathcal{D}_0(G_j)$ denote the corresponding choice of $\mathcal{D}_0$ made in the call $\texttt{DC}(G_j, \cdot, \cdot)$, for $1 \leq j \leq t-1$, and let $\mathcal{D}_0(G_t)$ equal the $\mathcal{D}_0$ above. Note that $T_1 \subseteq \cup_{j=1}^{t} \mathcal{D}_0(G_j)$.

Let $\bar{R}$ be a square of side $10a$ that is centered at the center of R_1, where a is the length of R_1. Lemma 6 implies that the number of j for which $l(G'_j)$ intersects $\bar{R}$ is bounded by a constant. It follows that the number of disks in T_1 that belong to $\mathcal{D}_0(G_j)$ for such j is bounded by a constant, since $|\mathcal{D}_0(G_j)| \leq 12$.

Now consider the disks in T_1 that belong to $\mathcal{D}_0(G_j)$ for j such that $l(G'_j)$ does not intersect $\bar{R}$. Any such disk intersects $l(G'_j)$ as well as R_1, so it must have radius at least a. It follows from Lemma 5 that the number of such disks is also bounded

by a constant.

We conclude that $|T_1| \leq \beta$ for a large enough constant β.

$\square$

It is immediate from Lemma 7 that after the call $\texttt{DC}(S, k, \emptyset)$ completes, $\text{Table}(P, k, \emptyset)$ contains a k-cover for P whose cost is at most cost(OPT), i.e. an optimal k-cover.

Theorem 8. *There is a polynomial time algorithm that, given a set P of points in the plane and an integer $k \geq 1$, returns an optimal k-cover of P.*

2.2.1 Relaxing the Assumption on the Computational Model

We now remove the assumption made in deriving Theorem 8 that the costs of two candidate solutions can be compared in polynomial time, and obtain an approximation algorithm. Let OPT continue to denote an optimal k-cover for the point set P. Suppose that for each disk $D \in \mathcal{D}$, we are given a *proxy cost* $\text{pcost}(D) \geq 0$ which is some rational number. Define the proxy cost $\text{pcost}(\mathcal{D}')$ of a set $\mathcal{D}' \subseteq \mathcal{D}$ of disks to be the sum of the proxy costs of the disks in $\mathcal{D}'$.

Let us modify the procedure $\texttt{DC}(R, \kappa, T)$ so that in Step 12, it compares proxy costs rather than the actual costs. That is, the modified predicate checks if $\text{pcost}(\mathcal{D}_0 \cup \text{Table}(P \cap R_1, \kappa_1, T_1) \cup \text{Table}(P \cap R_2, \kappa_2, T_2))$ is less than $\text{pcost}(\mathcal{D}')$. We also consider the proxy cost of the special disk I to be infinity. Arguing along the lines of the previous section we conclude:

Lemma 9. *After the call to the modified procedure $\texttt{DC}(S, k, \emptyset)$ completes, Table$(P, k, \emptyset)$ contains a k-cover for P whose proxy cost is at most pcost(OPT), the proxy cost of*

OPT.

Suppose that the co-ordinates of the input points P are integers whose binary encoding takes at most L bits. Given our tolerance parameter $\epsilon > 0$, we compute for each disk $D \in \mathcal{D}$ a rational number that lies in the interval $[\text{radius}(D), (1+\epsilon)\text{radius}(D)]$ and set $\text{pcost}(D)$ to be this number. This computation is readily accomplished in time polynomial in L and $\log \frac{1}{\epsilon}$ for a single disk D. With the proxy costs set in this manner, we conclude from Lemma 9 that after the call to the modified procedure $\texttt{DC}(S, k, \emptyset)$ completes, $\text{Table}(P, k, \emptyset)$ contains a k-cover for P whose cost is at most $(1+\epsilon)\text{cost}(\text{OPT})$.

Finally, it is straightforward to ensure that the computation of the canonical lines and the lines bounding the rectangles that are encountered as the arguments of our recursive algorithm takes time that is polynomial in the input size of the problem (and independent of ϵ). Also, predicates such as testing whether a disk $D \in \mathcal{D}$ contains a point $p \in P$, or testing whether a disk $D \in \mathcal{D}$ intersects a rectangle that is encountered in the course of the algorithm are also readily implemented in polynomial time.

Theorem 1 follows.

CHAPTER 3
NETWORK SPANNER TOPOLOGY DESIGN IN THE PLANE

Providing Internet connectivity to rural areas in developing regions is essential for enabling access to Information and Communication Technology services. Minimization of construction cost is a major challenge in network deployment here, because traditionally these populations are low-paying. In this context, we investigate efficient algorithms for the minimum cost topology construction problem in rural wireless mesh networks.

It is prohibitively expensive to provide wired connectivity in rural remote areas. Also, traditional wireless network technologies such as cellular data networks (e.g. EV-DO) and upcoming technologies like IEEE 802.16 WiMAX have prohibitively expensive equipment costs. As a result, there has been considerable recent interest in the design of rural mesh networks using IEEE 802.11 (WiFi) equipment. (See e.g. [25], [57], [61], [56], [54].) Current deployments include the Ashwini Project in Andhra Pradesh, [6] the Digital Gangetic Plains (DGP) [10] etc.

In such a network, long-distance wireless links, (typically 7-8 kms), are used to connect villages. The nodes in the topology are fixed (each node is a village). To establish long-distance links, it is essential that Line-of-Sight be maintained between radio antennas at the end-points. For this reason, the antennas need to be mounted on tall towers. The obstacles, in general, maybe in the form of trees, buildings and terrain. The required height of the towers depends on the positions and the heights of the obstacles along the link. The cost of a tower depends upon its height. For

relatively short heights (upto about 12 meters) antenna masts are sufficient. For greater heights, sturdier and much more expensive antenna towers are required. Also for a pair of nodes communicating with each other directly, the transmission power at the source should be sufficient for the signal to be received at the destination while obeying given constraints on the maximum Effective Isotropic Radiated Power (EIRP) at each node.

Obstacle Model. Let the parameter L denote an upper bound on the height of obstacles and d denote the minimum clearance distance around a tower location in a village - i.e. there exists a site in every village such that there is no obstacle within a distance d from this site. Our obstacle model places, for each pair of villages (i, j), obstacles of height L on segment $\bar{ij}$ at a distance of d from i and at a distance d from j. We assume that the distance between any 2 villages is at least $2d$.

Note that under this model, if both towers at the end-points of a link (i, j) have a height greater than (or equal to) L, then Line-of-Sight visibility is achieved regardless of the distance between them. Similarly, if both towers have a height less than L, then Line-of-Sight visibility cannot be achieved. This model was first proposed by Sen and Raman in [61]. It is reasonable to assume that in any village we can locate a spot that has a clearing of length d in all directions that is free of such obstacles. Indeed, in practice, d is at least about 1 Km; and often, not much more. Henceforth, without loss of generality, we shall assume $d = 1$. With this model, we do not have to bother about other obstacles on this link placed further from j.

Let $COVER(h)$ denote the set of pairs of nodes that can form direct links

under the height function h. Given a tower height, a *cost function* c, gives the cost of building a tower of that height. The *total cost* of a height function h is $\sum_{j \in V} c(h(v))$. Among all height functions h such that $COVER(h)$ results in a connected spanner (graph) on the vertex set V, our goal is to find the height function with the minimum total cost.

Tower Cost. It is well established empirically that for all towers other than antenna masts, the cost function can be approximated by the linear function [61]. For all towers of positive height $h > 0$ we thus define the cost function of the form $c(h) = \alpha \cdot h + \beta$. Also we define $c(0) = 0$. α and β are positive constants. Building a mast is as easy as placing a tall water pipe and hence the cost of constructing a mast is negligible in comparison to the cost of taller towers. Taller towers require large infrastructure cost which is accounted for by the constant β in our function defined above. Thus under this height function we can think of the topology as being "elevated" to the height of the masts making the cost of masts 0 for our purposes and the cost of any non-mast tower a linear function of its height. For the sake of exposition, we shall assume in most of the rest of this chapter that the cost of a tall tower is just a scale of its height; hence we attempt to minimize the sum of heights of towers as our objective function i.e. $\sum_{j \in V} h(j)$. Under this assumption, $c(h(j)) = h(j)$. Note, however, that our algorithms and theoretical guarantees about them hold for the more general cost function. Our numerical experiments and results will use the more general cost function.

Formally the problem we consider, the **Tower Cover With Power Con-**

straints (TCPC) problem, is stated as:

Input. A set of points V in the plane, obstacle height L, minimum clearance distance $d(= 1)$ and maximum distance B to which a node can transmit given bounds on transmit power.

Output. A valid height function h such that the set of links in $COVER(h)$ form a connected spanner of V such that the total height(cost) of the towers $\sum_{j \in V} h(j)$ is minimized.

Contribution and Results Our contributions are threefold. Firstly, we formulate the Topology Construction Problem as a geometric problem, the TCPC. This allows us to employ tools developed in the context of geometric set cover and facility location problems to this setting. Moreover, we believe our geometric framework will also allow us to place our solution in the context of the larger topology planning problem framed by Sen and Raman [61] that involves handling inter-link interference, throughput etc. Indeed, in their pioneering work [61], they mention the possible application of Computational Geometry techniques to tackle this problem. To the best of our knowledge we are the first to take this approach. Our approach appears to be robust enough to handle slight variants of the cost function.

Next, we show that the geometric version of this problem (the TCPC) admits a constant factor approximation -i.e. we obtain in polynomial time a solution whose cost is at most a constant factor times the optimal. The result is in contrast to the general graph-theoretic formulation where we cannot hope to obtain a better than

$O(\log n)$ approximation [54].Thus we prove Theorem 2.

Finally, through several experiments and numerical simulations we show that our algorithm in practice gives very good results well within the theoretical bounds we guarantee. In fact in most cases we obtain a solution that is within twice of the computed lower bound. A key idea that underlies these results is that we reduce the connectivity problem TCPC to a facility location/geometric cover problem by ignoring connectivity – restoring overall connectivity turns out to require only a modest increase in cost.

In Section 3.1, we pinpoint the facility location problem that underlies the TCPC problem. In Section 3.2, we present our constant factor approximation algorithm for the TCPC problem. In Section 3.3 we present results from numerical simulations that compare our approximation algorithm for TCPC with a computed lower bound and study the effect of the geometric parameters in the setting.

3.1 A Facility Location Problem

The optimal solution to TCPC consists of "tall towers" - those that are at least as tall as the obstacle height L and "short towers" those that have height less than L. (Note, short towers are distinct from masts.) Clearly, two tall towers see each other, and two short towers don't. Consider two nodes, i and j, such that j has a short tower of height h_1 and i has a tall tower of height h_2; see Figure 1.2. Then we have Line-of-Sight (LOS) clearance iff [61]

$$h_1 * (d(i,j) - d) + h_2 * d \geq L * d(i,j). \tag{3.1}$$

From the inequality, it is clear that the tall tower at i sees any tower of height 0 that is at a distance of at most $r \equiv \frac{h_2}{L}$ from i. Thus, the height of the tall tower is simply a scaled version of the maximum distance to which it can attain LOS connectivity if a tower of height 0 was placed at that distance (since L is an independent parameter). Thus we can think of the tall tower of height h as a disk of radius $r = \frac{h}{L}$ that "covers" all nodes that lie within this disk. We call such a disk an *LOS-disk* and the location of the tall tower the center of the LOS-disk. Henceforth, we use the terms "tall tower" and "LOS-disk" interchangeably, the meaning will be clear from the context.

Now, if $d(i,j) > r = \frac{h_2}{L}$, the required height of the short tower at j to achieve LOS visibility with i is given by

$$h_1 \geq \frac{L * (d(i,j) - r)}{d(i,j) - 1}$$

Thus, the height of the short tower at j is at least $L\frac{(d(i,j)-r)}{d(i,j)-1}$. Note that the height depends upon the distance from the tall tower which serves it and the height of this tall tower $(L \cdot r)$. If the height of the tall tower is fixed, the required height of the short tower increases with the length of the link.

A solution to the TCPC problem consists of tall and short towers so that each short tower has a direct link with some tall tower. We define the Bounded Range Tower Cover Problem (BRTC), which just asks for the minimum-cost solution that guarantees this:

Given a set V of points in the plane, find a set of LOS-disks $I = \{(i_1, r_1), (i_2, r_2) \cdots, (i_k, r_k)\}$ (where i_l is the center and r_l is the radius of disk l

and $r_l \geq 1$ for each l), and a function $\phi : V \rightarrow I$ that assigns each point $q \in V$ to some disk $(i_l, r_l) \in I$ such that $q \in V_l$; so as to minimize

$$L(\sum_{l=1}^{k} r_l + \sum_{l=1}^{k} \sum_{q:\phi(q)=(i_l,r_l)} \max(0, \frac{d(q,i_l) - r_l}{d(q,i_l) - 1}))$$

Here, and in the rest of the paper, V_l denotes the set $\{q \in V | d(q, i_l) \leq B\}$.

The first summation in the objective function corresponds to the heights of the tall towers, and the second to the minimum heights of the short towers given the tall towers. Figure 3.1 depicts a tall tower in a solution to BRTC together with the short towers it "serves". Clearly, the optimal solution to the BRTC has cost no greater than the optimal solution to the TCPC. Note that the BRTC does not enforce global connectivity of the topology and hence, a solution to the BRTC problem results in a height assignment under which certain pairs of points might be disconnected. Our strategy for the TCPC problem is to first solve the BRTC problem, - a facility location problem, and later ensure global connectivity.

3.1.1 Candidate LOS-Disks

Although any radius r (tower height $L \cdot r$) can be chosen for a given LOS-disk (tall tower), every solution for the BRTC is dominated by a solution in which for each chosen radius the corresponding disk has an input point (client node) - that is distinct from itself - on its border. This allows us to restrict ourselves to consider only $n(n-1)$ candidate canonical LOS-disks for our solution. In addition we also consider LOS-disks corresponding to tall towers of height L - i.e. radius 1 centered at each point - in our set of candidate disks giving us n^2 disks to choose from. It can be

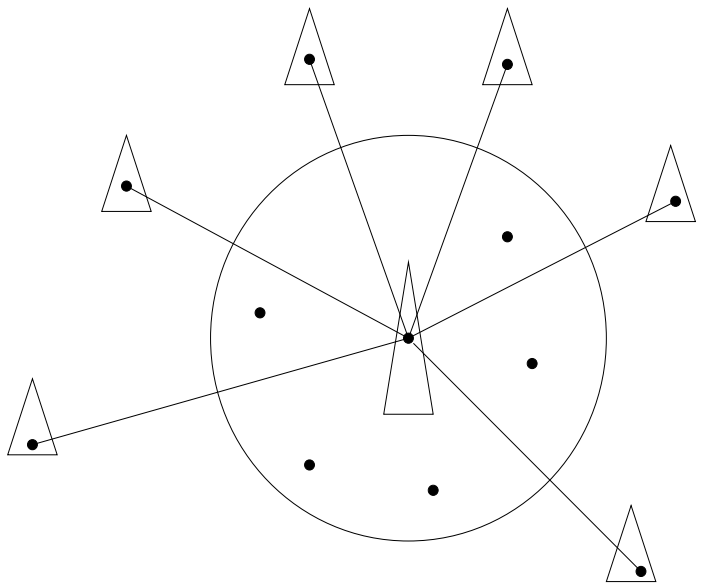


Figure 3.1: Points served by a tall tower in the BRTC: Note the tall tower at the center, the LOS-disk with height 0 towers inside it, and short towers outside it

shown that it is enough to consider only those solutions that contain disks from this candidate set. This is a common idea used in several disk covering problems [48, 32] which we extend to our setting.

3.2 Algorithm

In this section, we describe our algorithm for the TCPC problem. We first solve the BRTC problem and later, in Section 3.2.3, we modify our solution to BRTC by raising the heights of a few towers to ensure global connectivity.

3.2.1 Bounded Range Tower Cover Problem

The cost of our solution is the sum of the tower heights. To simplify our cost function we can factor out the parameter L. Thus our cost function is the sum of the radii of LOS-disks and the corresponding heights of short towers scaled down by

a factor of L.

Let F be the set of n^2 canonical disks defined in Section 3.1.1, each such disk is defined by a center i and radius r and denoted as (i, r). For a point j and a disk $(i, r) \in F$ such that $j \in V_i$, let c_{ij}^r denote the cost of a short tower at j to connect to disk (i, r). Thus $c_{ij}^r = 0$ if $d(i, j) \leq r \leq B$ and $c_{ij}^r = \frac{d(i,j)-r}{d(i,j)-1}$ if $r < d(i, j) \leq B$. Observe that the number of such $(j, (i, r))$ pairs is $O(n^3)$.

Our algorithm for the BRTC problem adapts the primal-dual approach of Jain and Vazirani to the metric uncapacitated facility location problem [39]. We have a non-negative variable α_j for each $j \in V$, and a non-negative variable β_{ij}^r for each disk $(i, r) \in F$ and $j \in V_i$. We compute an *assignment* to the α_js and the β_{ij}^rs that satisfy the following "dual" constraints:

$$\forall (i, r) \in F : \sum_{j \in V_i} \beta_{ij}^r \leq r$$
$$\forall (i, r) \in F, \forall j \in V_i : \alpha_j - \beta_{ij}^r \leq c_{ij}^r$$

We will call an (α, β) assignment that satisfies these dual constraints *dual-feasible*. We can interpret these variables as "paying" for a solution to the BRTC. In particular, α_j is the payment of point j. Suppose j ends up being assigned to disk (i, r) in a solution to BRTC. Then β_{ij}^r is j's contribution to the cost (radius) of LOS-disk (i, r) and the rest of the payment $\alpha_j - \beta_{ij}^r$ goes towards the cost c_{ij}^r of connecting j to (i, r).

Because of dual-feasibility, the total payment made by the points $\sum_{j \in V} \alpha_j$ can be shown to be *a lower bound on the cost of any solution to the BRTC problem.* In our analysis as well as in the numerical simulations in Section 3.3, we shall evaluate the performance of our algorithm against this lower bound. Our algorithm will find

a dual-feasible solution (α, β) and a corresponding solution to the BRTC with cost at most a constant times $\sum_{j \in V} \alpha_j$. This will imply that our BRTC solution is within a constant factor of the optimal.

Claim 10. *Let $I \subseteq F$, $\phi : V \to I$ be any (not necessarily optimal) solution to the BRTC problem and (α, β) be any dual-feasible solution. Then $\sum_{j \in V} \alpha_j \leq \sum_{(i,r) \in I} r + \sum_{j \in V} c_{\phi(j)j}$ where $c_{\phi(j)j} = c^{r'}_{i'j}$ such that $(i', r') = \phi(j)$.*

Proof. We have by the second dual constraint defined above that for each point j,

$$\alpha_j \leq \beta_{\phi(j)j} + c_{\phi(j)j} \text{ (where } \beta_{\phi(j)j} = \beta^{r'}_{i'j} \text{ s.t. } (i', r') = \phi(j))$$

The proof follows by taking the summation over all points. □

Our algorithm will find a dual-feasible solution (α, β) and a corresponding solution to the BRTC with cost at most a constant times $\sum_{j \in V} \alpha_j$. This will imply that our BRTC solution is within a constant factor of the optimal.

3.2.1.1 The Algorithm

The algorithm consists of two phases similar to [39]. The main difference is in Phase 2 and its analysis.

Henceforth, we shall refer to a pair consisting of a disk $(i, r) \in F$ and a point $j \in V_i$ as an "edge" (i, r, j).

Phase 1. A notion of *time* is defined in this phase, so that each event can be associated with the time at which it happened; the phase starts at time 0. Initially, each point in V is defined to be *unconnected* and the values α_j and β^r_{ij} are set to 0.

Throughout this phase, the algorithm raises the dual variable α_j for each unconnected point j uniformly at unit rate, that is α_j will grow by 1 in unit time. When $\alpha_j = c^r_{ij}$ for some edge (i, r, j) (where $j \in V_i$), the algorithm will declare this edge to be *tight*. Henceforth, dual variable β^r_{ij} will be raised uniformly, thus ensuring that the first constraint in the dual LP is not violated. (β^r_{ij} goes towards paying for disk (i, r).) Each edge (i, r, j) such that $\beta^r_{ij} > 0$ is declared *special*.

Disk (i, r) is said to be *paid for* if $\sum_{j \in V_i} \beta^r_{ij} = r$. If so, the algorithm declares this disk *temporarily selected*. Furthermore, all unconnected points having tight edges to this disk are declared *connected* and disk (i, r) is declared the *connecting witness* for each of these points. (Notice that the dual variables α_j of these points are not raised anymore.) In the future, as soon as an unconnected point j gets a tight edge to (i, r), j will also be declared connected and (i, r) will be declared the connecting witness for j (notice that $\beta^r_{ij} = 0$, and so edge (i, r, j) is not special.) When all points are connected, the first phase terminates. If several events happen simultaneously, the algorithm executes them in arbitrary order.

(Note that at the end of Phase 1, a point may have paid towards temporarily selecting several disks. However, we want to ensure that a point pay for only the disk that it is eventually connected to. This is handled by Phase 2, which chooses a subset of temporarily selected disks and scales them up for permanent selection.)

Phase 2. Let F_t denote the set of temporarily selected disks and G denote the graph consisting of all special edges. Let H denote the graph whose vertex set is F_t. In this graph there is an edge between a pair of temporarily selected disks (i, r)

and (i', r') iff there is a point j such that (i, r, j) and (i', r', j) are both special edges - i.e. (i, r) and (i', r') have special edges to a common neighbor j. Find a maximal independent set I' in H in the following manner [17].

1. Initially, T is the set of tight temporarily selected disks.

2. $I' = \Phi$.

3. Repeat steps 3a-3d until $T = \Phi$.

(a) Let (i, r) be the disk of largest radius in T (break ties arbitrarily).

(b) Let N be the set of all disks in F_t that have an edge to (i, r) in H including (i, r).

(c) $I' = I' \cup \{(i, r)\}$.

(d) $T = T \setminus N$.

We call this set I' our intermediate solution.

Now, for each disk in I' scale its radius by a factor of 3 so long as it is less than B. We call this scaled version of the radius r as $\hat{r}$. $\hat{r} = \min\{3r, B\}$. Call this new set of scaled disks I''. All disks in I'' are declared *selected*. (Note here that such disks in I'' do not necessarily come from the canonical set of disks.)

In addition, for each disk $(i, r) \in I'$ we pick (if required) a complementary set of disks. The disk $(i, \hat{r})$ along with these complementary disks constitutes the set $C_{(i,r)}$. We shall describe how we pick this set below.

Assignment. For point j, define

$$\mathcal{F}_j = \{(i,r) \in F_t | (i,r,j) \text{ is special}\}$$

i.e. the set of disks temporarily selected in Phase 1 to which j has special edges. When picking an independent set I' in Phase 2, at most one of the disks in $\mathcal{F}_j$ is selected in I'. If there is a disk $(i,r) \in \mathcal{F}_j$ that is selected, then set $\phi(j) = (i, \hat{r})$ (the scaled version of this disk), and declare city j *directly connected*.

Otherwise, if there is no disk $(i,r) \in \mathcal{F}_j$ that is selected in I', consider the tight edge (i', r', j) such that (i', r') was the connecting witness for j. If $(i', r') \in I'$ again set $\phi(j) = (i', \hat{r'})$ (Here $\hat{r'} = \min\{3r', B\}$) and declare point j *directly connected* (notice that in this case $\beta_{i'j}^{r'} = 0$).

In the remaining case viz. $(i', r') \notin I'$, let (i,r) be any neighbor of (i', r') in H such that $(i,r) \in I'$. We shall assign j to some disk in $C_{(i,r)}$. Say this disk is centered at i''. Set $\phi(j) = (i'', \hat{r})$ and declare point j *indirectly connected*. Note that such a disk exists by construction, which we shall describe below. We shall also show that by making this assignment the connection cost for j (i.e. the cost of the short tower j) is not increased by more than a constant factor.

Constructing a complementary set $C_{(i,r)}$. For any $(i,r) \in I'$, we pick $C_{(i,r)}$ as follows. First add $(i, \hat{r})$ (the scaled version of (i,r)) to $C_{(i,r)}$. Let J be the set of points that are assigned to be indirectly connected to a disk in $C_{(i,r)}$. Pick any point j from this set and put here a disk of radius $\hat{r}$. Include this in $C_{(i,r)}$. In addition, assign all points $j' \in J$ such that $d(j, j') \leq B$ to this disk i.e. set $\phi(j') = (j, \hat{r})$. Next pick any point that is not yet assigned to any disk in $C_{(i,r)}$ and repeat in this manner

until all points in J are assigned to some such disk in $C_{(i,r)}$.

Let $I = \bigcup_{(i,r)\in I'} C_{(i,r)}$. (Note that we include all the disks in I'' in this set.) (I, ϕ) is our final solution to the BRTC problem. Note that if $\phi(j) = (i, r)$, then $d(i, j) \leq B$.

Remark. We need to use complementary disks as an artefact of the power transmission constraint - to ensure that we assign each point to a disk that is centered not more than distance B from it. Also note that, we first try to cover the points directly and only if we fail then we go for indirect connections.

3.2.1.2 Analysis

In this section we shall show that our primal-dual algorithm for the BRTC problem give a constant factor approximation. Thus we prove Theorem 2 which states that given a set of nodes V in the plane, maximum transmission power range B, and an obstacle model as defined with parameters L (obstacle height) and d (obstacle clearance) we find a height assignment function h that assigns a non-negative height value to a tower at each node in V such that $COVER(h)$ results in a connected spanner on V and the total height $\sum_{j\in V} h(j)$ is within a constant factor of the total height in the optimal solution.We start by showing that the cost of the disks in our final solution is a scaled version of the cost of disks in our intermediate solution I' to within a constant factor. Then we show that the intermediate solution is accounted for by the payments α_j's made by the points. These two results are combined to give the approximation factor for the entire algorithm.

Lemma 11. *For a disk $(i, r) \in I'$, the total cost of disks in $C_{(i,r)}$ is at most $147r$.*

Proof. Let j be a point that is to be indirectly connected to (i, r). First, we will show that $d(i, j) \leq 3B$. Let (i', r') be the connecting witness of j whose neigbhor in I' is (i, r). Hence, there must be a point j' such that (i, r, j') and (i', r', j') are both special edges and so it follows that each of the distances $d(i, j'), d(i', j')$ and $d(i', j)$ are at most B. Thus, by triangle inequality, $d(i, j) \leq 3B$.

By construction, the distance between centers of any 2 such disks in $C_{(i,r)}$ is at least B and hence no more than 49 such disks are required to cover J. Thus the total cost of $C_{(i,r)}$ is at most $49 * \hat{r}$.

Since $\hat{r} = \min\{3r, B\}$, $\hat{r} = 3r$ if $r \leq \frac{B}{3}$ and $\hat{r} = B$ if $r > \frac{B}{3}$. In either case, we get that the cost of this set of disks in $C_{(i,r)}$, is at most 147 times the cost r of (i, r). □

Now, we use the above lemma to bound the cost of the disks in our final solution in terms of the cost of those in our intermediate solution.

Lemma 12. *The cost of disks in the final solution I is at most 147 times the cost of disks in intermediate solution I'.*

Note that $I = \bigcup_{(i,r) \in I'} C_{(i,r)}$ and hence the above lemma can be verfied easily.

Proof. We have seen that $I = \bigcup_{(i,r) \in I'} C_{(i,r)}$. .

Clearly, remembering that the cost of a set of disks is the sum of the radii of

the disks in the set,

$$\begin{aligned}
\text{cost}(I) &= \text{cost}(\bigcup_{(i,r)\in I'} C_{(i,r)}) \\
&\leq \sum_{(i,r)\in I'} \text{cost}(C_{(i,r)}) \\
&\leq \sum_{(i,r)\in I'} 147r \qquad\qquad \text{(from Lemma 11)} \\
&\leq 147 \sum_{(i,r)\in I'} r \\
&= 147\ \text{cost}(I')
\end{aligned}$$

□

Next, we will show how the dual variables α_j's pay for the costs (radii) of LOS-disks in intermediate solution I' and connecting points to LOS-disks (short towers). Let us partition the set of points V into V_{dc} - the set of directly connected points and V_{ic} - the set of indirectly connected points. Also define $V^{(i,r)}$ as the set of points $j \in V$ that have a special edge to disk (i, r). First we look only at the costs of the LOS-disks and the costs of direct connections.

Lemma 13. $\sum_{j:j\in V_{dc}} c_{\phi(j)j} + \sum_{(i,r)\in I'} r \leq \sum_{j:j\in V_{dc}} \alpha_j$

The proof of this Lemma is similar to that in [39]. Indeed I' was chosen to be an independent set in H with this Lemma in mind.

Proof. For directly connected point j suppose $\phi(j) = (i, \hat{r})$. Let $\phi'(j) = (i, r) \in I'$ be the disk that gave rise to $(i, \hat{r})$. Note this could have happened either by tripling r so that $\hat{r} = 3r$ (if $r \leq \frac{B}{3}$) or else by increasing r to B. As $\hat{r} \geq r$, $c_{\phi(j)j} \leq c_{\phi'(j)j}$.

(Required height of short tower is less if the tall tower at the other end of the link is made taller.)

$$\begin{aligned}
\sum_{j:j\in V_{dc}} c_{\phi(j)j} + \sum_{(i,r)\in I'} r &\leq \sum_{j:j\in V_{dc}} c_{\phi'(j)j} + \sum_{(i,r)\in I'} r \\
&= \sum_{j:j\in V_{dc}} c_{\phi'(j)j} + \sum_{(i,r)\in I'} \sum_{j\in V^{(i,r)}} \beta^r_{ij} \\
&= \sum_{j:j\in V_{dc}} c_{\phi'(j)j} + \sum_{j:j\in V_{dc}} \beta_{\phi'(j)j} \\
&= \sum_{j:j\in V_{dc}} \alpha_j
\end{aligned}$$

□

Next, we show how α_j's pay for the cost of indirect connections. In particular we show that the cost of the short tower (connection cost) at an indirectly connected point j is at most 3 times its payment α_j.

Lemma 14. *For an indirectly connected point j,*

$$c_{\phi(j)j} \leq 3\alpha_j$$

Proof. Let (i', r') be the connecting witness for point j and $(i, r) \in I'$ its neighbor in H. So by our assignment policy, $\phi(j) \in C_{(i,r)}$; i.e. j is assigned to some $(i'', \hat{r}) \in C_{(i,r)}$. Now, if $r > \frac{B}{3}$, then $\hat{r} = B$ and hence $c_{\phi(j)j} = 0$ by the manner in which complementary set $C_{(i,r)}$ was constructed. Hence in this case, the lemma is proved. In the other case when $r \leq \frac{B}{3}$ observe that $c_{\phi(j)j} \leq c^{3r}_{ij}$. This is because $d(i'', j) \leq B$ (again by the way $C_{(i,r)}$ is constructed) while $d(i, j) \geq B$ (otherwise $\phi(j) = (i, \hat{r}) = (i, 3r)$ and hence $c_{\phi(j)j} = c^{3r}_{ij}$). In the rest of the proof, we show that $c^{3r}_{ij} \leq 3\alpha_j$.

Since j is indirectly connected to $(i'', 3r)$, and $(i'', 3r) \in C_{(i,r)}$ with $(i, r) \in I'$, therefore $((i, r), (i', r'))$ must be an edge in H. In turn, there must be a point j' such that (i, r, j') and (i', r', j') are both special edges. Let t_1 and t_2 be the times at which (i, r) and (i', r') were declared temporarily selected during Phase 1.

Since edge (i', r', j) is tight, $\alpha_j \geq c^{r'}_{i'j}$. We will show that $\alpha_j \geq c^{r}_{ij'}$ and $\alpha_j \geq c^{r'}_{i'j'}$. This implies that $3\alpha_j \geq c^{r'}_{i'j} + c^{r}_{ij'} + c^{r'}_{i'j'}$. Then we will show that $c^{r'}_{i'j} + c^{r}_{ij'} + c^{r'}_{i'j'} \geq c^{3r}_{ij}$ which will prove the lemma.

Since edges (i', r', j') and (i, r', j') are tight, $\alpha_{j'} \geq c^{r}_{ij'}$ and $\alpha_{j'} \geq c^{r'}_{i'j'}$. During Phase 1, $\alpha_{j'}$ stops growing as soon as one of the disks that j' has a tight edge to is selected. Therefore, $\alpha_{j'} \leq \min(t_1, t_2)$. Finally, since (i', r') is the connecting witness for j, $\alpha_j \geq t_2$. Therefore $\alpha_j \geq \alpha_{j'}$ and $3\alpha_j \geq c^{r'}_{i'j} + c^{r}_{ij'} + c^{r'}_{i'j'}$.

It remains to be shown that $c^{r'}_{i'j} + c^{r}_{ij'} + c^{r'}_{i'j'} \geq c^{3r}_{ij}$. If j lies in the interior of $(i, 3r)$ then $c^{3r}_{ij} = 0$ and we are done.

Otherwise,

$$\begin{aligned}
c^{3r}_{ij} &= \frac{d(i,j) - 3r}{d(i,j) - 1} \\
&\leq \frac{d(j,i') + d(i',j') + d(j',i) - 3r}{d(j,i') + d(i',j') + d(j',i) - 1} \\
&\leq \max\{0, \frac{d(j,i') - r}{d(j,i') - 1}\} + \max\{0, \frac{d(i',j') - r}{d(i'j') - 1}\} \\
&\quad + \max\{0, \frac{d(j',i) - r}{d(j',i) - 1}\} \\
&= c^{r}_{i'j} + c^{r}_{i'j'} + c^{r}_{ij'} \\
&\leq c^{r'}_{i'j} + c^{r'}_{i'j'} + c^{r}_{ij'}
\end{aligned}$$

The first step above follows from the triangle inequality while the last inequal-

ity holds because $r \geq r'$. Thus

$$c^{r}_{ij'} + c^{r'}_{i'j'} + c^{r'}_{i'j} \geq c^{3r}_{ij}$$

□

Corollary 15. $\sum_{j:j \in V_{ic}} c_{\phi(j)j} \leq 3 \sum_{j:j \in V_{ic}} \alpha_j$

Proof. The corollary follows from Lemma 14 and taking the summation over all indirectly connected points j. □

From Lemmas 12, 13 and corollary 15 we get that our final solution $(I, \phi : V \to I)$ is at most 147 times $\sum_j \alpha_j$ which is 147 times the optimal by Claim 10.

3.2.1.3 Running Time

In Phase 1, for each disk (i, r) we can compute the anticipated time t_{ir} at which it is completely paid for and we maintain these events in a binary heap. Two types of events can occur that may cause us to update this heap- a) An edge (i, r, j) becomes tight and b) Disk (i, r) is completely paid for. Each such update can be done in $O(\log n^2) = O(\log n)$ time. Note each edge (i, r, j) can cause an update at most twice - once when it goes tight and second, when point j is declared connected. Hence the running time of Phase 1 is $O(n^3 \log n)$.

It is easy to see that this is the phase that dominates the running time of our algorithm and hence the next theorem follows.

Theorem 16. *The Algorithm based on the Primal Dual Schema achieves an approximation factor of* 147 *for the Bounded Range Tower Cover (BRTC) and has a running*

time of $O(n^3 \log n)$.

3.2.2 A Local Optimization

At this juncture, we have a constant factor approximation to the BRTC, but this constant may be quite large for practical purposes. When we tested our algorithm on a few real and synthetic topologies we found that the solution computed is anywhere between 5-15 times the guaranteed lower bound; so we try to improve this using a very natural local optimization.

For each tall tower $t \in I$ in our solution, and point set $U \subseteq V$ assigned to t, we find the best height assignment to U that guarantees that a tall tower at t serves each point in U. We need to search only $|U|$ height assignments.

As a second step, we allow each point to change its server tall tower - i.e. if a point has several tall towers within distance B from it (candidate servers) it can pick one so as to minimize the height of its short tower required to achieve LOS-connectivity. Observe that in both steps above we only improve the cost of the solution while maintaining feasibility.

3.2.3 Connecting The Solution

As mentioned at the start of the section we need to ensure connectivity in our solution to make sure that any pair of villages can communicate with each other along some path P in the topology where the length of each link in the path P is at most B and the tower heights at the end-points of each link are sufficient to allow Line-of-Sight visibility.

Consider a "connected component" induced by the solution I returned by the above primal dual algorithm. Such a connected component consists of a set of points such that any two points within this set have a communication path P as mentioned above. Typically, this might contain one or more disks that our primal dual algorithm output along with points assigned to these. Signal transmission can be achieved (via one or more hops) between any two points in a component. Call such a component a cluster. Let $V_1, V_2 \cdots, V_K$ be the clusters induced on V by our solution.

Let H be the graph whose vertices are clusters. If there exists a pair of points i and j in V such that $d(i,j) \leq B$ and if $i \in V_k$ and $j \in V_l$ lie in separate clusters ($k \neq l$), then place an edge between V_l and V_k in edge set E_H of H. We let the weight of each such edge be 2. This essentially means that if we find two points lying in different components that are close enough to be within power transmission range, but cannot see each other, then we allow them to see each other by making both towers tall (of height L).

We now find a spanning tree T in H. (Since all edges in H are of weight 2 any spanning tree is a Minimum Spanning Tree.) Finally we connect the topology by placing at the end points of each edge in T tall towers of height L i.e. LOS-disks of radius 1. These can now form a direct LOS-link.

The following theorem shows that the cost of connecting up in the solution in this manner is within a constant factor times the cost of the optimal solution to the TCPC problem.

Theorem 17. *Let V be the set of points in the plane and I be the set of LOS-disks in*

the solution (I, ϕ) to the BRTC problem obtained above. Let I^* be the set of LOS-disks in optimal solution to the TCPC problem for input V. There is an efficient algorithm that computes a set I^c of towers(disks) such that $(I \cup I^c, \phi)$ is a feasible solution to the TCPC problem and cost$(I^c) \leq 50$cost(I^*).

Proof. Consider any spanning tree T in H and let I^c be the set of disks of radius 1 placed at the end points of the edges in T exactly as described above. Clearly costI^c is at most the cost of the spanning tree. If the solution (I, ϕ) to our BRTC induces K clusters on V the cost of this spanning tree is $2(K-1)$. Now we show that this cost is at most a factor of 50 times worse than the cost of the LOS-disks I^* in the optimal solution to the TCPC problem, thus completing the proof.

Consider the $K-1$ edges in I^* that connect the K clusters induced on V by (I, ϕ). At least one of the end points of each of these edge has a tall tower; otherwise LOS link could not have been formed. Furthermore a LOS-disk (tall tower) can contribute to at most 25 such edges : Whenever a tall tower contributes to the weight of an edge, it connects two clusters in I whose closest points have graph-theoretic distance at most 2 from it and hence must lie within a distance of $2B$ from it. However each of these clusters will have centers that are at least B apart, otherwise they will collapse into a single cluster because there will be complete LOS and power connectivity between the clusters. Hence there can be at most 25 such connections to which a tower can contribute.

Hence the total cost of these edges in I^* that connect clusters induced by (I, ϕ) is at least $\frac{K-1}{25}$ and so cost$(I^*) \geq \frac{K-1}{25}$. But the cost of our solution obtained

by adding the edges of a spanning tree of H is at most $2(K-1)$ and hence it is most 50 times worse than $\text{cost}(I^*)$. $\square$

We thus yield Theorem 2.

3.3 Evaluation

We evaluate the main aspects of our approach. We explore whether it is practical and what results it gives in real-world scenarios. We carry out several numerical simulations to evaluate our algorithm with respect to the computed lower bound. These are implemented in C++.

3.3.1 Sample Topologies

We evaluate our algorithm on a bunch of synthetic topologies as well as on one real-world topology - the Ashwini topology [6] considered by Sen and Raman [61]. We have tried to model this topology as closely as possible. For the synthetic topologies we consider the village nodes to lie in an area of 50 sq. Km. We generate the co-ordinates of these nodes randomly while maintaining minimum intervillage distance of 2 Km. Also, we generate topologies with varying densitities - viz. topologies with 25, 50, 75 and 100 nodes over the same sized area (50 Sq. Km.) and for each such density we have a set of 5 randomly generated instances. Note that the Ashwini topology has 34 nodes spread across a similar area.

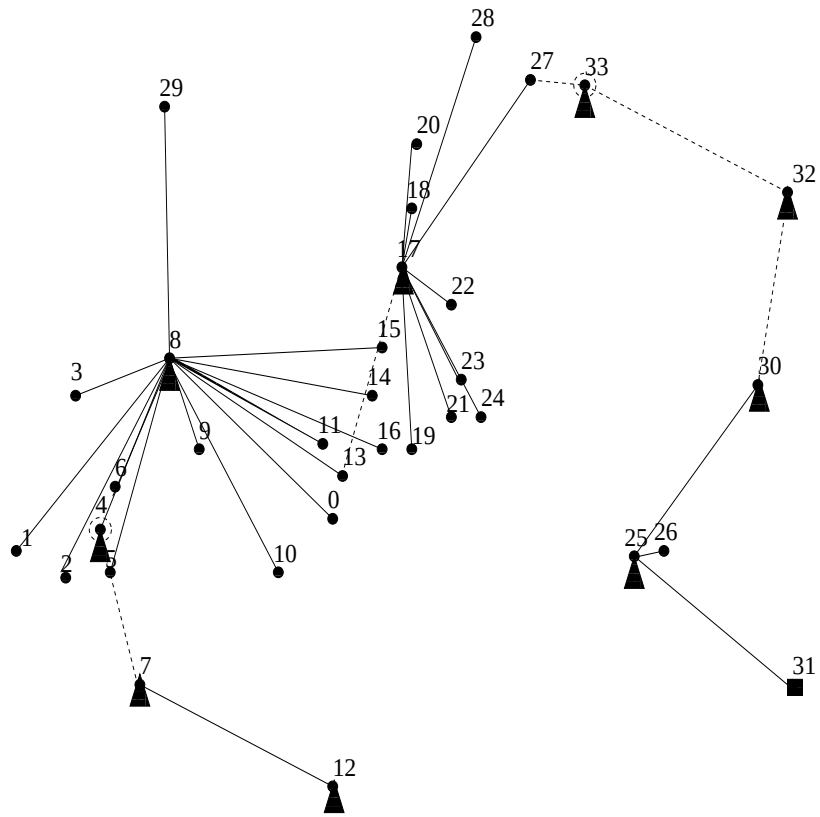


Figure 3.2: Topology Generated For Ashwini

3.3.2 Cost Function

We use a linear cost function for non-mast towers (as mentioned before) - i.e. $c(h) = \alpha \cdot h + \beta$. For a mast of course we assume the cost to be 0. For ease of exposition we scale the cost by a factor of $\frac{1}{L}$.

3.3.3 Real Topologies

The Ashwini topology [6] consists of 34 nodes (including 1 landline node). Figure 3.2 shows the topology generated by our algorithm. We assumed $L = 0.006$, ($6m$ above mast level) and $d = 1Km$ to be uniform for all links. We also restrict our

link length B to $15Km$. [1]

In the figure, the triangles correspond to tall towers, circles correspond to masts while squares are short towers. Also a dotted circle around a tall tower indicates that the height of that tower was raised to ensure connectivity in the last phase while a dotted line indicates a connection between villages served by two different tall towers.

The solution gives two main tall towers (nodes $17, 8$), several masts and a few tall towers of obstacle height (L) some of which are raised during final connectivity step. The overall cost for this topology is about 63.7 and the algorithm runs almost instantaneously on a $2.2GHz$ desktop. The lower bound computed as described in Section 3.2.1 for this topology is about 47 which gives us a solution well within 35% of the optimal. A trivial solution which places height L towers at each of the 34 nodes has cost of about 113.

Importance of local optimization. Without the local optimization, our solution on the same input topology had cost 209. Clearly optimization results in savings of a multiplicative factor of more than $3 - 4$. Observe Figure 3.3. This observation also holds for the synthetic topologies.

3.3.4 Synthetic Topologies

We use the following parameter settings for our evaluation. We consider 4 different values for B - $10Km$, $12Km$, $14Km$ and $16Km$ and 4 different values of obstacle height L (above mast level) - $4m$, $6m$, $8m$, and $10m$. Also we use $d = 1Km$.

[1]Note height of masts is usually about $12m$ or so; hence we consider the worst case scenario for obstacle height.

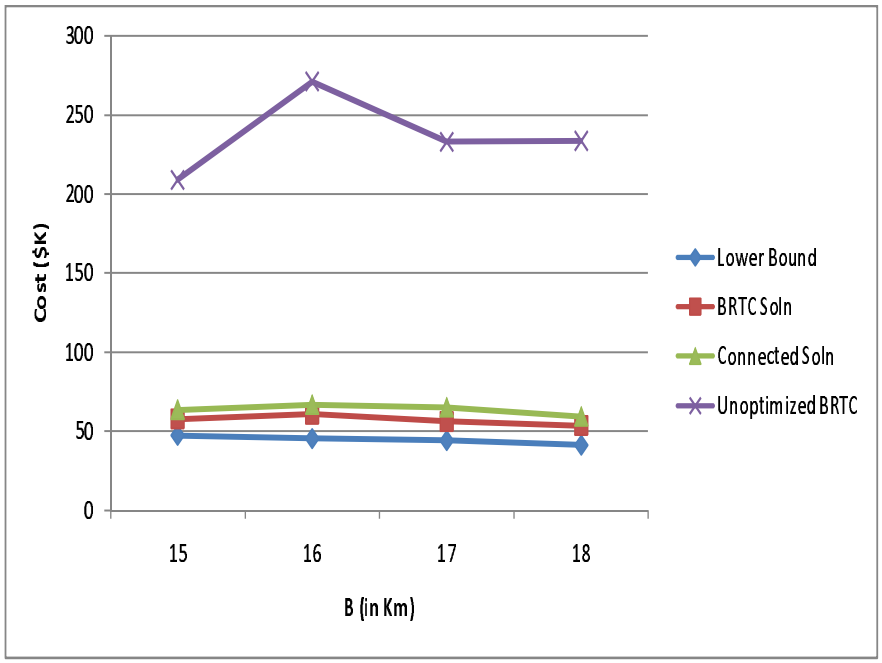


Figure 3.3: Advantage of Optimization (Ashwini)

Comparing with Lower Bound. We compute the lower bound in our algorithm for the BRTC Problem as described in Section 3.2.1. The lower bound is the sum of the payments made by the points towards the construction of towers i.e. $\sum_{j \in V} \alpha_j$. We compare the solution returned by our approximation algorithm with this lower bound for topologies of different densities and with different values of parameters L and B.

The results in Table 3.1 show that all our approximations give ratios that are nearly always within twice the lower bound and in most cases much better. We compare the costs of our solution before and after the final connecting-up phase (these are separated by a comma in the table cells). Note that the lower bound is with regard to the BRTC problem and hence does not take into account the connectivity of the

Table 3.1: Maximum Values for Ratios Before And After Connecting

L \ B	10 Km	12 Km	14 Km	16 Km
4 m	1.79,1.96	1.94,2.00	1.96,2.07	1.95,1.99
6 m	1.69,1.88	1.84,1.90	1.83,1.95	1.90, 1.94
8 m	1.62,1.81	1.79,1.86	1.75,1.85	1.87, 1.91
10 m	1.59,1.73	1.76,1.80	1.69,1.75	1.85,1.90

Table 3.2: Comparison with Greedy Algorithm of [54]

n	Avg. Ratio of cost of Greedy Algorithm vs Ours
25	1.19
50	1.37
75	1.41
100	1.47

solution. Hence in reality our algorithm may be doing even better.

Additional Connectivity Cost. While the lower bound does not account for the connectivity requirements, we show here that additional cost for connectivity is not too high. This can be seen clearly from Table 3.1. This result in a sense validates our approach of looking at the problem as primarily a bounded range tower cover problem postponing the connectivity phase.

Comparison With Panigrahi et. al [54] We compare our approach with the greedy algorithm of [54] on the above synthetic topologies. The results in Table 3.2 show that on average for all topology sizes (and densitites) the greedy algorithm gives solutions that are about $20 - 45$ percent worse. These averages are again taken over varying values of B and L.

3.3.5 Running Time

Our algorithm has a theoretical guarantee of a $O(n^3 \log n)$ for the running time. This is important because it clearly means that our algorithm can scale to larger or denser topologies easily. Even for inputs with 100 nodes our algorithm runs in less than a few seconds. Moreover the run-time of our algorithm is dominated by the size of the topology and hence values of parameters like L,B and d have very little effect.

This is in contrast to the exhaustive search approach proposed by Sen and Raman [61] which does not really scale beyond topologies of size 30 or so as noted. We have implemented the exhaustive search approach of [61] on Ashwini and as expected this takes several hours. It is not reasonable to hope to be able to run this on larger topologies.

3.3.6 Summary

To summarize, our numerical experiments demonstrate that our algorithm performs well within its worst case performance bounds. and much better on topologies that have density and layout similar to real-world topologies. Also it scales efficiently

to larger and denser topologies.

CHAPTER 4
NETWORK SPANNER TOPOLOGY DESIGN ON THE TERRAIN

In this chapter we consider the problem of constructing towers at given villages (nodes) that lie on a 1.5-dimensional terrain so that the topology formed by the underlying links is connected. Problems in computational geometry most notably the guarding problem have been well-studied on the 1.5-D terrain [26, 9, 42, 43, 27, 44, 34]. While the problem is interesting in itself, geometric coverage problems have been found to be hard on the general 2.5-dimensional terrain and results on this simpler terrain could possibly provide hints towards obtaining solutions for these. For instance, the 1.5-D terrain guarding problem is a non-trivial instance of guarding polygons and has implications for the famous art gallery problem.

A 1.5-D terrain is a polygonal chain in the plane that is x-monotone, that is, any vertical line intersects the chain at most once. A terrain T consists of a set of m vertices $\{t_1, t_2, \cdots, t_m\}$. The vertices are ordered in increasing order with respect to their x-coordinates. There is an edge connecting t_i with t_{i+i} for all $i = 1, 2, \cdots m-1$. Given a point a denote its x-coordinate by a^x and y-coordinate by a^y. If $t_1^x \leq a^x \leq t_m^x$ let y^* be the y-coordinate of the point at which the vertical through a hits the terrain T. Now point a is said to lie *above* the terrain if $a^y > y^*$, *on* the terrain if $a^y = y^*$ and *below* the terrain if $a^y < y^*$. For any 2 points a, b, that lie on or above the terrain, we say that a sees b if the line segment $\bar{ab}$ lies entirely above or on the terrain.

The villages (nodes) V in our topology are a finite subset of the points on the terrain T. Let $|V| = n$. A height assignment function h gives an assignment

of tower heights to every node in V. Let $COVER(h)$ denote the set of pairs of nodes that form direct links under the height function h. A pair of nodes (v_i, v_j) in V is in $COVER(h)$ if tower heights $h(i)$ and $h(j)$ are such that the line joining the top of the towers lies entirely on or above the terrain T. See Figure 4.1 for an example. Furthermore, for every vertex $v \in V$ we are given a discrete set of candidate height assignments $H(v)$. We make the assumption that the zero height assignment is always available i.e. $\forall v \in V, 0 \in H(v)$ We consider the problem of finding a height assignment that minimizes the sum of tower heights at all nodes (villages) in the topology such that $COVER(h)$ results in a connected spanner (graph) on the vertex set V. We call this the Min-cost Terrain Spanner problem. We assume that if every node v is assigned the maximum possible allowed height from its candidate set $H(v)$ we get a feasible solution. This discrete version of the problem can be obtained from the continuous version (where the candidate heights need not be discrete) via a polynomial time reduction such that an α-approximation for the discrete version gives an $O(\alpha)$-approximation for the continuous version.

4.1 Our Contribution.

We look for a constant factor approximation algorithm for the discrete version of the problem defined above. We first prove that existing approaches do not do better than a logarithmic approximation. The general graph-theoretic approach of [54] can be applied to our setting here. It is a greedy approach and it cannot do better in this special case than the general guarantee of the logarithmic approximation it

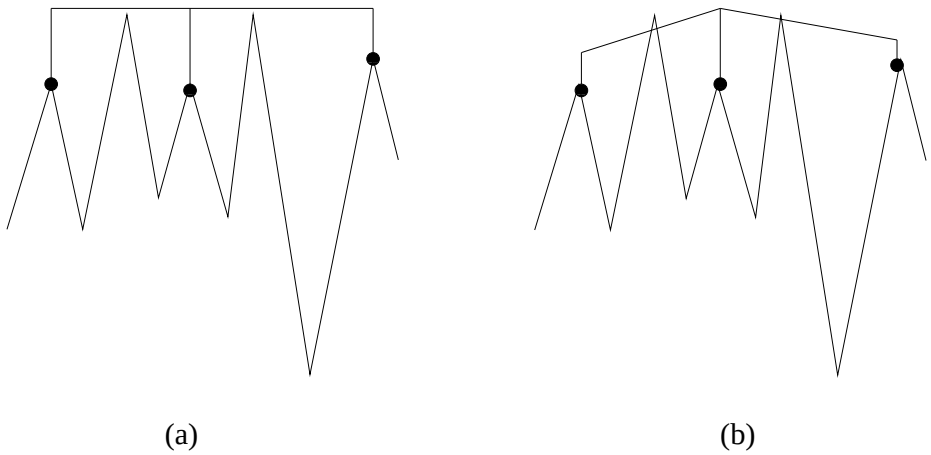


Figure 4.1: Example of height assignments. (a) This height assignment provides a feasible solution. (b) This height assignment does not provide a feasible solution

gives. We prove this by construction of a tight example for this approach.

Also we show that the Min-cost Terrain spanner problem can be reduced to the Node Weighted Steiner Tree problem. This independently implies a logarithmic approximation. A primal-dual algorithm for Node Weighted Steiner Tree problem by Demaine etal [23] gives a constant factor approximation if the graph under consideration is planar. However it cannot do so for our problem. We prove this again by a tight example.

We try to use the geometry of this particular setting to see if we can do better than in the general case - i.e. obtain a sublogarithmic approximation. The 1.5-D terrain has specific geometric properties w.r.t visibility that have been expolited previously to obtain solutions for the terrain guarding problem [26, 9, 34, 42, 43, 27, 44]. However we are unable to exploit these geometric properties to obtain an efficient constant factor approximation algorithm for the topology construction problem. To

the best of our knowledge we are not aware of any other results on this particular problem.

In section 4.2 we discuss the greedy approach of [54] and in Section 4.3 we discuss the linear programming based primal dual approach [23].

4.2 Greedy Algorithm

First we define the general version of the problem as considered by the authors in [54], then describe their algorithm and then provide an example on the 1.5-D terrain where it cannot do better than an $O(\log n)$ approximation.

4.2.1 Problem Definition

Let $G = (V, E)$ where V is the set of vertices and E is the set of edges. Let $n = |V|$. A height assignment function h gives an assignment of tower height to every node of G. Only discrete height assignments are considered - i.e. for each vertex v we have a set of candidate height assignments $\mathcal{H}(v)$. We assume that it is possible to determine efficiently if an edge (u, v) is considered to be covered by a height function (In our application, tower heights $h(u)$ at u and $h(v)$ at v need to be sufficient to obtain a clear *visual line-of-sight* between the antennas placed on top of towers of the respective heights at u and v.) $COVER(h)$ denotes the set of edges that are covered by height function h. The cost function c of building a tower is the height of the tower. The total cost of height function is h is $\sum_{v \in V} h(v)$. The objective is to find h of minimum total cost such that $COVER(h)$ forms a connected spanner of the vertex set V. Note that our Min-cost Terrain Spanner problem is a geometric special case

of this problem.

4.2.2 Algorithm

Consider the following algorithm for the topology cover problem given by [54]. Here $COMP(h)$ denotes the number of components in graph with vertex set V and edge set $COVER(h)$.

In Algorithm 4.2.2,the height function is initialized to 0 at all nodes. Thus at the beginning $COVER(h) = \phi$ and $COMP(h) = n$. Algorithm 4.2.2 proceeds in phases. In each phase a height increment $incr_{best}$ is selected for all nodes which is then added to the current height function. This is repeated until $COVER(h)$ is a connected spanning subgraph.

4.2.2.1 Picking the best height increment in a phase

Every height increment has a cost associated with it, namely the increase in the total cost of the towers because of applying the increment. Roughly speaking, in a phase, the algorithm selects that height increment that requires minimum increase in cost for the maximum reduction in the number of components. This metric is called the cost-to-benefit ratio. We consider a natural space of increments in V over which the best cost-to-benefit ratio can be computed easily.

The algorithm searches over each candidate node v for every candidate height increment via the candidate set $\mathcal{H}(v)$. After this we need to search over the possible height increments at the neighbors of v. This subproblem is called STAR-TC: Given the current height function h, a central node v and its height increment $\delta(= \alpha - h(v))$,

Algorithm 4.1 TC-ALGO (G, c)

```
for each v ∈ V do
    h(v) = 0
end for
while COMP(h) > 1 do
    r_best = ∞
    for each v ∈ V do
        for α ∈ H(v) s.t. α > h(v) do
            (r_tmp, incr_tmp) ← STAR-TC-ALGO(G, h, v, α − h(v));
            if r_tmp < r_best then
                r_tmp ← r_best; incr_best ← incr_temp;
            end if
        end for
    end for
    for each v ∈ V do
        h(v) ← h(v) + incr_best(v);
    end for
end while
return h
```

find a height increment at the neighbors of v that has the lowest cost to benefit ratio.

Now we are ready to describe the procedure STAR-TC-ALGO.

STAR-TC-ALGO. First note that increasing the height at nodes that are in the same component as v does not reduce the number of components. Thus, the height increments are restricted only to those neighbors of v that are in a different component than v in $COVER(h)$. This set of nodes is called nbr. For every node u in nbr the smallest height increment $h^+(u)$ at u that covers edge (u, v) is computed. The nodes in nbr are listed in increasing order of their h^+ values in a list L. Also it is ensured that for every component L contains the height increment corresponding to exactly one node - that with the minimum h^+ value over all nodes in that component.

Now the following $|L|$ height increments are considered. Each increment increases the height only at v and nodes that form a prefix of L. More precisely, the ith height increment increases the height of v by δ, heights of the first i nodes in L by their respective h^+ and all the other nodes by 0. The benefit of such an increment is exactly i. The cost of such an increment is $\delta + \sum_{1 \leq j \leq i} h^+(j^{\text{th}}\text{node in } L)$ Among all these height increments the one that has the lowest cost-to-benefit ratio is returned.

This algorithm achieves an approximation ratio of $O(\log n)$ where $n = |V|$. The proof can be found in [54] and is omitted here.

Now consider the following instance of the topology cover problem on the 1.5-D terrain as shown in Figure 4.2.

Here, $v_1, \cdots v_{n+2}$ represent village locations or nodes. The candidate height

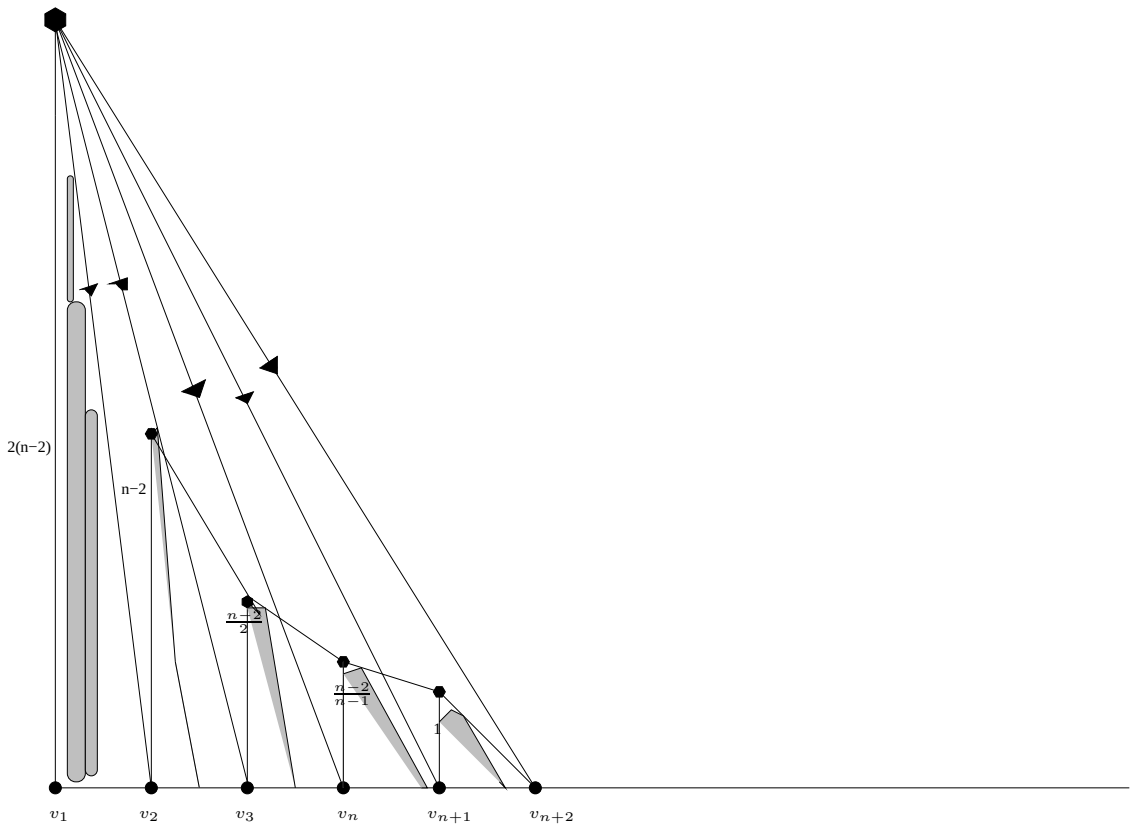


Figure 4.2: Bad Example for the Panigrahi et. al Algorithm (Greedy Algorithm)

functions are defined as follows. $H(v_{n+2}) = \{0\}$, $H(v_{n+1}) = \{0, 1\}$, $H(v_1) = \{0, 2(n-2)\}$ and $\forall 2 \leq i \leq n, H(v_i) = \{0, \frac{n-2}{i-1}\}$. The tower of height $2(n-2)$ at vertex v_1 can see every other candidate tower. $\forall 1 \leq i \leq n+1$ the tower of height 0 at vertex v_i can only see this special tower (and the tower at vertex v_i of non-zero height). The zero height tower at node v_{n+2} can see the height 1 tower at node v_{n+1}. The visibilites between the non-zero height towers at $v_2, \cdots, v_{n+1}$ do not matter. In this example a single tower of height $2(n-2)$ placed at the leftmost node is sufficient to attain line-of-sight connectivity with all other nodes having zero height towers and thus obtain a connected topology. This is in fact the optimal solution and its cost is $2(n-2)$.

However Algorithm 4.2.2 will return a solution consisting of the $n+1$ non-zero height towers at $v_1, \cdots, v_{n+1}$ and a zero height tower at node v_{n+2} as shown in the figure. To see this notice that initially constructing a tower of height 1 at the point second from the right gives a cost-to-benefit ratio of 1 which is the best possible considered for $n > 5$. This follows from the visibilities defined above. Based on the visibilities one can see via an inductive argument that the algorithm will proceed to construct non-zero height towers at nodes $v_{n+1}, v_n, v_{n-1}, \cdots v_1$ in the specified order. Thus the cost of the solution given by this greedy algorithm is $2(n-2) + \sum_{i=2}^{n} \frac{n-2}{i-1} + 1 = 2n - 3 + (n-2)\theta(\log n)$ which is a $\Omega(\log n)$ factor worse than the optimal.

4.3 Primal Dual Algorithm

We show how the Min-cost Terrain spanner problem can be reduced to the Node-weighted Steiner tree problem.

Consider an instance of the problem given as the terrain T, a finite subset V of T and for each point v in V we have a set of candidate height assignments. Now construct a graph $G_s = (V_s, E_s)$ where V_s consists of copies of each point $v \in V$ corresponding to its possible height assignments $\mathcal{H}(v_i)$. Again we assume $\forall v_i \in V, 0 \in \mathcal{H}(v_i)$ Define v_{ih} to be the vertex that corresponds to the copy of vertex v_i and height assignment $h \in \mathcal{H}(v_i)$. The set E_s consists of edges $(v_{ih}, v_{jh'})$ such that a tower of height h at v_i can see a tower of height h' at v_j. In addition we define a cost function $C : V_s \rightarrow \Re$ such that $C(v_{ih}) = h$ - i.e. the cost of a node corresponds to the height of the tower it represents. Now call the set of all vertices v_{i0} the set of *terminals*.

The goal is to find a Steiner tree that contains all such terminals of minimum total cost. It is easy to see that a solution to an instance of this Node-Weighted Steiner Tree Problem gives a solution of exactly the same cost to the instance of the Min-cost Terrain spanner problem.

Klein and Ravi [45] give a logarithmic approximation for the Node-Weighted Steiner Tree problem. The algorithm maintains a node-disjoint set of trees containing all the terminals. Initially, each terminal is a tree by itself. The algorithm uses a greedy strategy to iteratively merge the trees into larger trees until there is only one tree. In each iteration, it selects a node and a subset of trees of size at least two so as to minimize the ratio

$$\frac{\text{cost of the node plus sum of the distances to the trees from node}}{\text{number of trees}}$$

Here the distance along a path does not include the cost of its end points. Thus the choice minimizes the average node-to-tree distance. The algorithm uses the shortest paths between the vertex and the selected trees to merge the trees into one.

It is easy to notice the similarities of the algorithm with the one given by Panigrahi et al [54] in Section 4.2. Moreover, if the example for the algorithm in Section 4.2 is formulated as a Steiner Tree problem as described above it serves as a bad example in this case.

4.3.1 Integer Programming

For an instance of a Node Weighted Steiner Tree Problem as described above with given graph $G_s = (V_s, E_s)$, where V_s is as defined in the previous section and

the set of terminals being the set $\{v_{i0}|\forall v_i \in V\}$ we obtain an integer programming formulation. We then explore possible rounding schemes for the fractional solution obtained by solving a linear programming relaxation. We show that several intuitive rounding schemes can give an arbitrarily bad approximation.

The following integer programming formulation of the problem is the same as Demaine et al. [23]. We use $\Gamma(S)$ to denote the set of nodes that are not in S but have neighbors in S for $S \subseteq V_s$. Each vertex $v \in V_s$ has an associated variable x_v and an associated cost h_v.

$$\min \sum_{v \in V_s} h_v x_v \tag{4.1}$$

$$\text{subject to :} \tag{4.2}$$

$$\sum_{v \in \Gamma(S)} x_v \geq f(S), \forall S \subseteq V_s \tag{4.3}$$

$$x_v \in \{0, 1\}, \forall v \in V_s \tag{4.4}$$

where $f(S) = 1$ if S contains some terminal but not all. Otherwise $f(S) = 0$. The solution to this integer program is a solution to the Steiner Tree instance and hence a solution to the 1.5-D Topology Construction problem.

The linear relaxation of the above integer program is obtained by replacing the constraint $x_v \in \{0, 1\}$ with the constraint $x_v > 0$. The dual of the resulting linear program is as follows:

$$\max \sum_{S \subseteq V_s} f(S) y_S \tag{4.5}$$

$$\text{subject to :} \tag{4.6}$$

$$\sum_{S \subseteq V_s : v \in \Gamma(S)} y_S \leq h_v, \forall v \in V_s \tag{4.7}$$

$$y_S \geq 0, \forall S \subseteq V_s \tag{4.8}$$

There is a dual variable y_S for each subset S of V_s. However, the only such variable that affect the objective function (and therefore the only variables we need to consider) are those variables y_S where $f(S) = 1$. Intuitively, the goal of the dual linear program is to find a maximal size family of node sets S with $f(S) = 1$ subject to the constraint that each node v is the neighbor of at most h_v sets in the family.

Primal-Dual Algorithm. Let $X \subseteq V_s$ be any subset of nodes. A node set S is a *violated connected component with respect to* X if S is a connected component "induced by" X and $f(S) = 1$. A *partial solution* is a set X of nodes containing $\{v : f(v) = 1\}$ such that there is some violated connected component with respect to X. An oracle Viol(.) takes a partial solution X as input and outputs the violated connected components with respect to X. X initially consists of all nodes v such that $f(\{v\}) = 1$ (i.e. V). These nodes of course have cost zero.

The algorithm maintains a dual feasible solution y.

For a set X of nodes of G_s, a set F of nodes is a *feasible augmentation* of X if $F \supseteq X$ and F is a feasible solution. If in addition no proper subset of F is a feasible augmentation of X then F is a *minimal* feasible augmentation of X.

---

1. $y \leftarrow 0$

2. $X \leftarrow \{v : f(\{v\}) = 1\}$

while there is a violated connected component with respect to X **do**

(a) Increase $y(S)$ uniformly for all sets $S \in \text{Viol}(X)$ until the dual linear-program inequality for some v becomes tight: $\sum_{S \subseteq V_s : v \in \Gamma(S)} y_S \leq h_v, \forall v \in V_s$

(b) Add v to X, i.e. $x_v \leftarrow 1$.

end while

for each v in X in the reverse of the order in which they were added during the while-loop: **do**

(a) If $f(S) = 0$ for every connected component S induced by $X - \{v\}$, then remove v from X.

end for

3. Return X.

Since Demaine et al. [23] consider this problem in a planar graph they are able to prove a theorem that bounds the number of additional vertices in a minimal feasible augmentation in terms of the number of violated components. In particular they prove that $\sum\{|F \cap \Gamma(S)| : S \in \text{Viol}(X)\} \leq 6|\text{Viol}(X)|$. However the graph that is obtained from our problem the Min-cost Terrain Spanner problem is not planar. On this graph it is not possible to obtain such a bound. The following bad example illustrates this. See Figure 4.3.

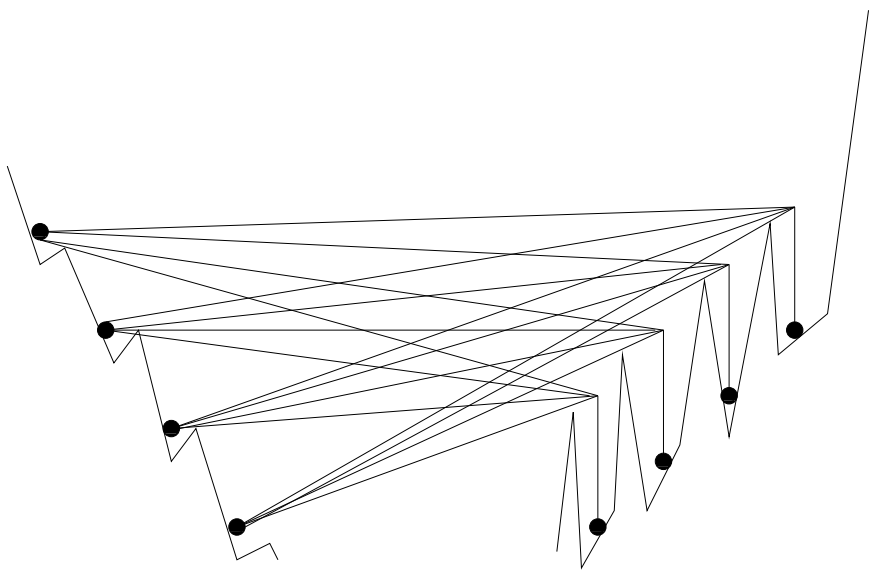


Figure 4.3: Bad Example for the primal-dual algorithm

On this terrain the vertices V are marked with black dots. For each vertex only two distinct height assignments are available - height zero and height H - i.e. $\forall v_i \in V$, $\mathcal{H}(v_i) = \{0, H\}$. Here, the complete solution consists of $\frac{n}{2}$ towers of height 0 on the left side and $\frac{n}{2}$ towers of height H on the right side.

Thus when formulated as Node Weighted Steiner Tree instance, V_s consists of $\{v_{ih}|v_i \in V, h \in \mathcal{H}(v_i)\}$. E_s consists of edges from every vertex in V_s corresponding to the villages on the left side to every vertex v_{iH} for the corresponding villages on the right hand side.

In this example let X be the set of all terminals i.e. $X = V$. Hence every terminal is a violated component - i.e. there are n violated components. Clearly the set $F = \{v_{iH}$ corresponding to all villages on RHS$\}$ is a minimal feasible augmentation of X. To see this note that by building towers of height H at all vertices on right hand side gives us a feasible solution and if any one of these towers is not built the solution no longer remains feasible. Thus $\sum\{|F \cap \Gamma(S)| : S \in \text{Viol}(X)\}$ is $\Omega(n^2)$ while $|\text{Viol}(X)|$ is only $O(n)$. Hence the gap is arbitrarily bad.

We can extend the above example to provide a bad example for the algorithm itself. Consider a modified terrain (Figure 4.4) where at the rightmost vertex a tower of height $8H$ sees all villages and let this be a candidate height assignment. Clearly the optimal solution builds this single height $8H$ tower but the algorithm will not be able to find this solution. This is because the constraints for vertices corresponding to height H towers will become tight earlier in the Node Weighted Steiner Tree formulation - i.e. when the values y_v will become $\frac{H}{(\frac{n}{2}+1)}$ for all $v \in V$. Thus the optimal solution has cost $8H$ while the algorithm gives a solution of cost $\frac{nH}{2}$ - a gap of $\Omega(n)$.

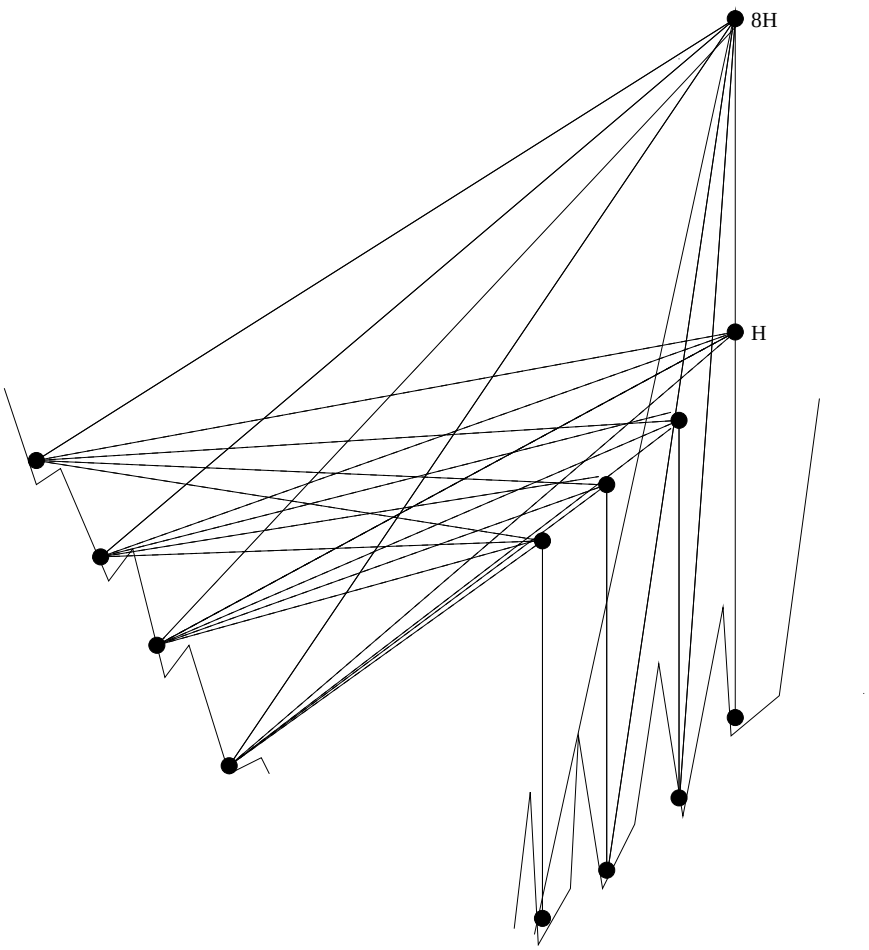


Figure 4.4: Bad Example for the primal-dual algorithm (Figure not to scale). The tall tower at the rightmost vertex is of height $8H$ and is part of the optimal solution. The dark lines indicate visibilities in the optimal solution while the dotted lines indicate visibilities in the solution built by the algorithm

4.4 Remarks

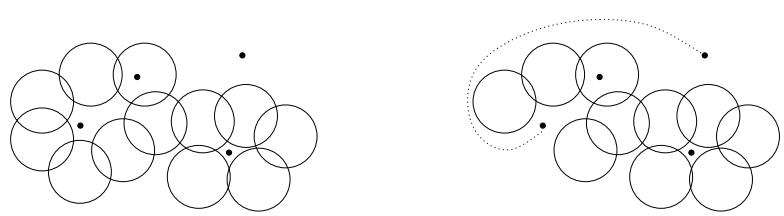
A couple of remarks about this problem - our approaches towards it and its hardness - are in order here. Geometry has been found to be helpful in making it possible to obtain efficient algorithms for a large variety of coverage problems. However, the connectivity requirement of this problem possibly makes it hard even with the help of geometry. In particular it makes it hard to find an appropriate

rounding scheme in the linear programming approach. Also the problem does not seem easily amenable to dynamic programming. There seems to be no easy intuitive way to decompose it into smaller subproblems and this might again be an artefact of the connectivity (in addition to coverage) requirement.

CHAPTER 5
MULTIWAY BARRIER CUT

Wireless sensors are being extensively used in applications to provide barriers as a defense mechanism against intruders at important buildings, estates, national borders etc. Monitoring the area of interest by this type of coverage is called *barrier* coverage [47]. Such sensors are also being used to detect and track moving objects such as animals in national parks, enemies in a battlefield, forest fires, crop diseases etc. In such applications it might be prohibitively expensive to attain blanket coverage but sufficient to ensure that the object under consideration cannot travel too far before it is detected. Such a coverage is called *trap* coverage [7, 60].

Inspired by such applications, we consider the problem of isolating a set of points by a minimum-size subset of a given set of unit radius disks. A unit disk crudely models the region sensed by a sensor, and our work here readily generalizes to disks of arbitrary, different radii.



(a) (b)

Figure 5.1: Example of a multiway barrier cut. (a) This set of disks separates the points because every path connecting any two points must intersect a disk. (b) This set of disks does not separate the points.

Problem Formulation. The input to our problem is a set I of n unit disks, and a set P of k points such that I *separates* P, that is, for any two points $p, q \in P$, every path between p and q intersects at least one disk in I. The goal is to find a minimum cardinality subset of I that separates P. See Figure 5.1 for an illustration of this notion of separation.

For the problem of covering points by the smallest subset of a given set of unit disks, approximation algorithms have been designed that guarantee an $O(1)$ approximation and even a PTAS [12, 53]. These results hold even for disks of arbitrary radii. Our problem can be viewed as a set cover problem where the elements that need to be covered are not points, but paths. However, known results only imply a trivial $O(n)$ approximation when viewed through this set cover lens.

Another example of a problem that has received such attention is the *independent set* problem. For many geometric variants [14, 15, 30], approximation ratios that are better than that for the combinatorial case are known.

Our problem is similar to the node multi-terminal cut problem in graphs [31]. Here, we are given a graph $G = (V, E)$ with costs on the vertices and a subset $U \subseteq V$ of k vertices, and our goal is to compute a minimum cost subset of vertices whose removal disconnects every pair of vertices in U. This problem admits a poly-time algorithm that guarantees an $O(1)$ approximation. We note however that the problem we consider does not seem to be a special case of the multi-terminal cut problem.

Contribution and Related Work. Our main result is a polynomial time algorithm that guarantees an $O(1)$ approximation for the problem. Thus we prove Theorem 3 which states that for a set I of n unit disks and P a set of k points such that I separates P there is a polynomial time algorithm that computes a subset $O \subseteq I$ of disks that also separates P, with the guarantee that $|O|$ is within a multiplicative $O(1)$ of the smallest subset of I that separates P.To the best of our knowledge, this is the first non-trivial approximation algorithm for this problem. Our algorithm is simple and combinatorial and is in fact a greedy algorithm. We first present an $O(1)$ approximation for the following two-point separation problem. We are given a set of unit disks G, and two points s and t, and we wish to find the smallest subset $B \subseteq G$ so that B separates s and t.

Our greedy algorithm to the overall problem applies the two-point separation algorithm to find the cheapest subset B of I that separates some pair of points in P. Suppose that P is partitioned into sets $P_1, P_2, \ldots, P_\tau$ where each P_i is the subset of points in the same "face" with respect to B. The algorithm then recursively finds a separator for each of the P_i, and returns the union of these and B.

The analysis to show that this algorithm has the $O(1)$ approximation guarantee relies on the combinatorial complexity of the boundary of the union of disks. It uses a subtle and global argument to bound the total size of all the separators B computed in each of the recursive calls.

Our approximation algorithm for the two-point separation problem, which is a subroutine we use in the overall algorithm, is similar to fast algorithms for finding

minimum s-t cuts in undirected planar graphs, see for example [59]. Our overall greedy algorithm has some resemblance to the algorithm of Erickson and Har-Peled [28] employed in the context of approximating the minimum cut graph of a polyhedral manifold. The details of the our algorithm and the analysis, however, are quite different from these papers since we do not have an embedded graph but rather a system of unit disks. Sankararaman et al. [60] investigate a notion of coverage which they call *weak coverage*. Given a region $\mathcal{R}$ of interest (which they take to be a square in the plane) and a set I of unit disks (sensors), the region is said to be k-weakly covered if each connected component of $\mathcal{R} - \bigcup_{d \in I} d$ has diameter at most k. They consider the situation when a given set I of unit disks *completely* covers $\mathcal{R}$, and address the problem of partitioning I into as many subsets as possible so that $\mathcal{R}$ is k-weakly covered by every subset. Their work differs in flavor from ours mainly due to the assumption that I completely covers $\mathcal{R}$.

In Section 5.1, we discuss standard notions we require, and then reduce our problem to the case where none of the points in input P are contained in any of the input disks. In Section 5.2, we present our approximation algorithm for separating two points. In Section 5.3, we describe our main result, the constant factor approximation algorithm for separating P. We conclude in Section 5.4 with some remarks.

5.1 Basic Concepts

We will refer to the standard notions of vertices, edges, and faces in arrangements of circles [1]. In particular, for a set R of m disks, we are interested in the faces

in the complement of the union of the disks in R. These are the connected components of the set $\Re^2 - \bigcup_{d \in R} d$. We also need the combinatorial result that the number of these faces is $O(m)$. Furthermore, the total number of vertices and edges on the boundaries of all these faces, that is, the combinatorial complexity of the boundary of the union of disks in R, is $O(m)$ [1]. We make standard general position assumptions about the input set I of disks in this chapter. This helps simplify the exposition and is without loss of generality.

Lemma 18. *Let R be a set of disks in the plane, and Q a set of points so that (a) no point from Q is contained in any disk from R, and (b) no face in the complement of the union of the disks in R contains more than one point of Q. Then $|R| = \Omega(|Q|)$.*

Proof. The number of faces in the in the complement of the union of the disks in R is $O(|R|)$. □ □

Covering vs. Separating. The input to our problem is a set I of n unit disks, and P a set of k points such that I separates P. Let $P_c \subseteq P$ denote those points contained in some disk in I; and P_s denote the remaining points. We compute an α-approximation to the smallest subset of I that covers P_c using a traditional set-cover algorithm; there are several poly-time algorithms that guarantee that $\alpha = O(1)$. We compute a β-approximation to the smallest subset of I that separates P_s, using the algorithm developed in the rest of this chapter. We argue below that the combination of the two solutions is an $O(\alpha + \beta)$ approximation to the smallest subset of I that separates P.

Let OPT $\subseteq I$ denote an optimal subset that separates P. Suppose that OPT covers k_1 of the points in P_c and let $k_2 = |P_c| - k_1$. By Lemma 18, $|\text{OPT}| = \Omega(k_2)$.

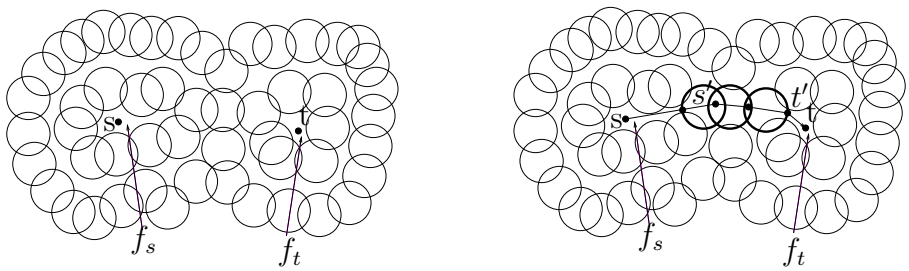
Now, by picking one disk to cover each of the k_2 points of P_c not covered by OPT, we see that there is a cover of P_c of size at most $|\text{OPT}| + k_2 = O(|\text{OPT}|)$. Thus, our α-approximation has size $O(\alpha) \cdot |\text{OPT}|$. Since OPT also separates P_s, our β-approximation has size $O(\beta) \cdot |\text{OPT}|$. Thus the combined solution has size $O(\alpha + \beta) \cdot |\text{OPT}|$.

In the rest of the chapter, we abuse notation and assume that no point in the input set P is contained in any disk in I, and describe a poly-time algorithm that computes an $O(1)$-approximation to the optimal subset of I that separates P.

5.2 Separating Two Points

Let s and t be two points in the plane, and G a set of disks such that no disk in G contains either s or t, but G separates s and t. See Figure 5.2. Our goal is to find the smallest cardinality subset B of G that separates s and t. We describe below a polynomial time algorithm that returns a constant factor approximation to this problem.

Without loss of generality, we may assume that the intersection graph of G is connected. (Otherwise, we apply the algorithm to each connected component for which the disks in the component separate s and t. We return the best solution obtained.) Let f_s and f_t denote the faces containing s and t, respectively, in the arrangement of G. We augment the intersection graph of G with vertices correspond-



(a) (b)

Figure 5.2: Computing the shortest path σ in the intersection graph. (a) The figure shows faces f_s and f_t (b) This figure shows the sequence of disks in σ (their boundaries are bold) and the path π.

ing to s and t, and add an edge from s to each disk that contributes an edge to the boundary of the face f_s, and an edge from t to each disk that contributes an edge to the boundary of the face f_t. We assign a cost of 0 to s, t, and a cost of 1 to each disk in G. We then find the shortest path from s to t in this graph, where the length of a path is the number of the *vertices* on it that correspond to disks in G. Let σ denote the sequence of disks on this shortest path. Note that any two disks that are not consecutive in σ do not intersect.

Using σ, we compute a path π in the plane, as described below, from s to t so that (a) there are points s' and t' on π so that the portion of π from s to s' is in f_s, and the portion from t' to t is in f_t; (b) every point on π from s' to t' is contained in some disk from σ; (c) the intersection of π with each disk in σ is connected. See Figure 5.2.

Suppose that the sequence of disks in σ is $d_1, \ldots, d_{|\sigma|}$. Let s' (resp. t') be a point in d_1 (resp. d_σ) that lies on the boundary of f_s (resp. f_t). For $1 \leq i \leq |\sigma| - 1$,

choose x_i to denote an arbitrary point in the intersection of d_i and d_{i+1}. The path π is constructed as follows: Take an arbitrary path from s to s' that lies within f_s, followed by the line segments $\overline{s'x_1}, \overline{x_1x_2}, \ldots, \overline{x_{|\sigma|-2}x_{|\sigma|-1}}, \overline{x_{|\sigma|-1}t'}$, followed by an arbitrary path from t' to t that lies within f_t.

Properties (a) and (b) hold for π by construction. Property (c) is seen to follow from the fact that disks that are not consecutive in σ do not overlap.

Notice that π "cuts" each disk in σ into two pieces. (Formally, the removal of π from any disk in σ yields two connected sets.) The path π may also intersect other disks and cut them into two or more pieces, and we refer to these pieces as disk pieces. For a disk that π does not intersect, there is only one disk piece, which is the disk itself.

We consider the intersection graph H of the disk pieces that come from disks in G. Observe that a disk piece does not have points on π, since π is removed; so two disk pieces intersecting means there is a point outside π that lies in both of them. In this graph, each disk piece has a cost of 1.

In this graph H, we compute, for each disk $d \in \sigma$, the shortest path between the two pieces corresponding to d. Suppose $d' \in \sigma$ yields the overall shortest path σ'; let D denote the set of disks that contribute a disk piece to this shortest path. Our algorithm returns D as its computed solution. See Figure 5.3.

We note that D separates s and t – in particular, the union of the disk pieces in σ' and the set $\pi \cap d'$ contains a cycle in the plane that intersects the path π between s and t exactly once.

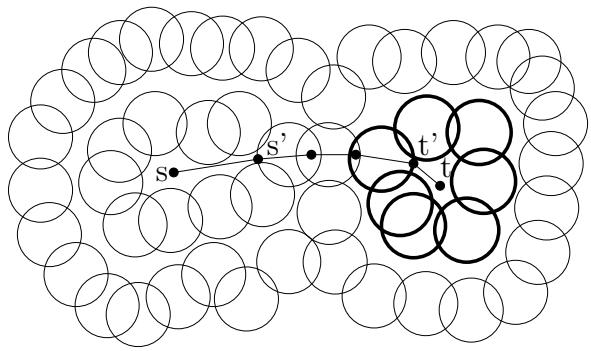


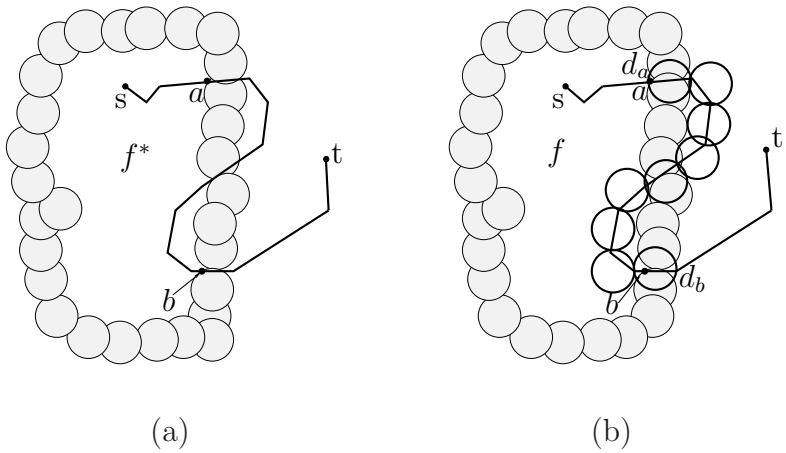
Figure 5.3: This figure continues with the example of Figure 5.2. The disks with bold boundary are the set D computed by our algorithm. The only disk from G with bold boundary has two disk pieces, and the shortest path between them in graph H yields D.

5.2.1 Bounding the Size of the Output

Let B^* denote the smallest subset of G that separates s and t. We will show that $|D| = O(|B^*|)$. Let f^* denote the face containing s in the arrangement of B^*. Due to the optimality of B^*, we may assume that the boundary of f^* has only one component. Let a (resp. b) denote the first (resp. last) point on path π where π leaves f^*. It is possible that $a = b$. We find a minimum cardinality contiguous subsequence $\overline{\sigma}$ of σ that contains the subpath of π from a to b; let d_a and d_b denote the first and last disks in $\overline{\sigma}$. See Figure 5.4.

We claim that $|\overline{\sigma}| \leq |B^*| + 2$; if this inequality does not hold, then we obtain a contradiction to the optimality of σ by replacing the disks in the $\overline{\sigma} \setminus \{d_a, d_b\}$ by B^*.

Consider the face f containing s in the arrangement with $B^* \cup \overline{\sigma}$. Each edge that bounds this face comes from a single disk piece, except for one edge corresponding to d_a that may come from two disk pieces. (This follows from the fact that the portion of π between a and b is covered by the disks in $\overline{\sigma}$.) These disk pieces induce a path in H



(a) (b)

Figure 5.4: The shaded disks are in B^*.(a) This figure shows points a and b where π leaves f^* for the first and last time, respectively. (b) This figure shows the face f containing s in the arrangement with $B^* \cup \overline{\sigma}$

in between the two pieces from d_a, and their cost therefore upper bounds the cost of D. We may bound the cost of these disk pieces by the number of edges on the boundary of f (with respect to $B^* \cup \overline{\sigma}$). The number of such edges is $O(|B^*| + |\overline{\sigma}|) = O(|B^*|)$.

Theorem 19. *Let s and t be two points in the plane, and G a set of disks such that no disk in G contains either s or t, but G separates s and t. There is a polynomial time algorithm that takes such G, s, and t as input, and outputs a subset $B \subseteq G$ that separates s and t; the size of B is at most a multiplicative constant of the size of the smallest subset $B^* \subseteq G$ that separates s and t.*

5.3 Separating Multiple Points

We now present a polynomial time algorithm that yields an $O(1)$ approximation to the problem of finding a minimum subset of I that separates the set P of points thus proving Theorem 3. The algorithm is obtained by calling recSep(P),

where recSep(Q), for any $Q \subseteq P$ is the following recursive procedure:

1. If $|Q| \leq 1$, return $\emptyset$.

2. For every pair of points $s, t \in Q$, invoke the algorithm of Theorem 19 (with $G \leftarrow I$) to find a subset $B_{s,t} \subseteq I$ such that $B_{s,t}$ separates s and t.

3. Let B denote the minimum size subset $B_{s,t}$ over all pairs s and t considered.

4. Consider the partition of Q into subsets so that each subset corresponds to points in the same face (with respect to B). Suppose $Q_1, \ldots, Q_\tau$ are the subsets in this partition. Note that $\tau \geq 2$, since B separates some pair of points in Q.

5. Return $B \cup \bigcup_{j=1}^{\tau} \text{recSep}(Q_j)$.

Clearly, recSep(P) yields a separator for P. To bound the size of this separator, let us define a set $\mathcal{Q}$ that contains as its element any $Q \subseteq P$ such that $|Q| \geq 2$ and recSep(Q) is called somewhere within the call to recSep(P). For any $Q \in \mathcal{Q}$, define B_Q to be the set B that is computed in the body of the call to recSep(Q). Notice that recSep(P) returns $\cup_{Q \in \mathcal{Q}} B_Q$.

Now we "charge" each such B_Q to an arbitrary point within $p_Q \in Q$ in such a way that no point in P is charged more than once. A moment's thought reveals that this is indeed possible. (In a tree where each interval node has degree at least 2, the number of leaves is greater than the number of internal nodes.)

Let OPT denote the optimal separator for P and let $F_Q \subseteq$ OPT denote the disks that contribute to the boundary of the face (in the arrangement of OPT) containing p_Q. We claim that $|B_Q| = O(|F_Q|)$; indeed F_Q separates $p_Q \in Q$ from

any point in P, and thus any point in Q. Thus for any $t \in Q \setminus \{p_Q\}$, we have $|B_Q| \leq B_{p_Q,t} = O(|F_Q|)$.

We thus have

$$\bigcup_{Q \in \mathcal{Q}} |B_Q| \leq \sum_{Q \in \mathcal{Q}} O(|F_Q|) = O(|\mathrm{OPT}|),$$

where the last equality follows from union complexity.

Theorem 3 follows.

5.4 Remarks

We have a presented an $O(1)$-approximation algorithm for finding the minimum subset of a given set I of disks that separates a given set of points P. One way to understand our contribution is as follows. Suppose we had at our disposal an efficient algorithm that optimally separates a single point $p \in P$ from every other point in P. Then applying this algorithm for each point in P, we get a separator for P. That the size of this separator is within $O(1)$ of the optimal is an easy consequence of union complexity. However, we only have at our disposal an efficient algorithm for a weaker task: that of approximately separating two given points in P. What we have shown is that even this suffices for the task of obtaining an $O(1)$ approximation to the overall problem.

Remark on Extension to Disks of Arbitrary Radii. It is easy to see that our algorithm and the approximation guarantee generalize to the case when the disks have arbitrary and different radii. To see this note that our analysis does not use any

property that is specific to unit disks. The only geometric property of disks we use is the union complexity being linear.

CHAPTER 6
CONCLUSIONS AND OPEN PROBLEMS

This document has given a summary of some work done in the area of algorithms for combinatorial optimization problems in a geometric setting. Each problem presents unique geometric properties and as a result the ability to exploit this structure for providing solutions varies in terms of efficiency and quality of approximation. Clearly, there are several related open, interesting and challenging problems in this area. We conclude the document with a discussion of open problems related to our work.

6.1 Clustering to Minimize the Sum of Radii

Our work on geometric clustering to minimize the sum of radii and the related work of [33] for the metric version of the problem are very interesting from a theoretical perspective. The problem seems to be extremely similar to many other variants of clustering which are NP-hard, yet admits an algorithm that is within a factor ϵ of optimal and runs in time polynomial in input size n and $\log \frac{1}{\epsilon}$. Under certain standard assumptions of model of computation it admits an exact polynomial time algorithm. The metric version of the problem can be solved exactly in quasipolynomial time when the aspect ratio is a polynomial in the number of input points. While these results have great theoretical impact, the running times are too large to be of practical interest. It would be interesting to see if the separability properties of the problem can also be used to give exact algorithms or interesting heuristics with polynomial

running times.

6.2 Network Spanner Topology Design in the Plane

Our work on network design on the plane is very interesting from a practical as well as theoretical perspective. We have given a constant factor approximation to the Tower Cover Problem with power constraints using the geometry of the setting thus improving over the previous work of [54]. Their approach is not geometric in nature and as such cannot do better than a logarithmic approximation. Our algorithm is also very efficient and practical and can be applied to real-world scenarios as indicated by our experimental results. We thus improve upon the exhaustive search approach of [61]. However the overall network design problem has a lot more questions that are still open. The tower cover problem is only a part of the overall topology construction problem. Some of the challenging issues that are still open include ensuring that the constructed topology satisfy other constraints regarding maintaining throughput, avoiding interference and a bound on the hop count of transmission.

6.3 Network Spanner Topology Design on the Terrain

Our work on the problem of finding a minimum cost tower height assignment to village nodes lying on a 1.5-dimensional terrain shows some limitations of geometric techniques. We show that the problem of finding a connected spanner by assigning heights to towers in this setting can be reduced to a Node Weighted Steiner Tree

Problem implying a logarithmic approximation. On the other hand this might perhaps help in explaning and addressing some of the issues faced in obtaining a better (constant factor) approximation.

6.4 Multiway Barrier Cut

We formulated the Multiway Barrier Cut problem - this problem is also very interesting form a theoretical and practical point of view. In this problem we seek to minimize the number of wireless unit disk sensors used to ensure that a given set of points is "barrier-separated" - i.e. any path between any pair of points is cut off by some sensor. This problem has practical implications in applications such as border patrol, guarding estates, national parks etc. On the theoretical side this problem is a version of geometric set cover with a different flavor - here the elements to be covered are all possible paths in the plane between any two pair of points and the sets are the disks representing the sensors. We give a constant factor approximation algorithm to this problem that is simple, recursive and greedy and easy to implement. Our algorithm also works for disks of arbitrary radii.

As mentioned in Section 5.4 the tool we have at our disposal is an efficient algorithm for separating a pair of points. This is a weaker tool than one which would separate a single point from all other points. Despite this limitation we are able to efficiently separate all pairs of points. An interesting question then is given a set Z of k pairs of input points X i.e. $Z = \{(x_1, y_1), (x_2, y_2), \cdots (x_k, y_k)\}$ such that every pair $(x_i, y_i) \in Z$ needs to be separated can we still obtain an efficient approximation

with this or some other approach. This problem would then have the flavor of the multi-cut problem which is harder in the traditional graph theoretic setting than the multiway cut problem. It will be interesting to see how well this particular problem can be approximated in this geometric setting. A slight variant would be where the set of pairs of points that need to be separated consists of all pairs that are at least some given distance apart. This models more closely the original motivation of trap coverage.

Finally, implementation of our algorithm and simulation of various scenarios would be an interesting and challenging project. This would involve representing the input in terms of arrangement of disks, building the geometric disk intersection graphs on which shortest path routines are run and verifying separation of points in the ararangements of the disks picked. While this is a polynomial time algorithm one does not know how efficient it will be in practice on real-world scenarions but the algorithm only needs to be run once; so one can afford to spend some time doing this. An interesting idea would be to design several practical input scenarios - for instance, border patrol, college campus, guarding estates etc. With each of these the layout of points and disks maybe slightly different. It would be then interesting to see whether the algorithm does better on some specific input scenario.

REFERENCES

[1] Pankaj K. Agarwal and Micha Sharir. *Davenport-Schinzel Sequences and Their Geometric Applications, Cambridge University Press*. 1998.

[2] Helmut Alt, Esther M. Arkin, Hervé Brönnimann, Jeff Erickson, Sándor P. Fekete, Christian Knauer, Jonathan Lenchner, Joseph S. B. Mitchell, and Kim Whittlesey. Minimum-cost coverage of point sets by disks. In *Symposium on Computational Geometry*, pages 449–458, 2006.

[3] Boris Aronov, Esther Ezra, and Micha Sharir. Small-size epsilon-nets for axis-parallel rectangles and boxes. In *STOC*, pages 639–648, 2009.

[4] Sanjeev Arora. Polynomial time approximation schemes for euclidean tsp and other geometric problems. In *FOCS*, pages 2–11, 1996.

[5] Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. Approximation schemes for euclidean k-medians and related problems. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 106–113, New York, NY, USA, 1998. ACM.

[6] Ashwini. http://www.byrrajufoundation.org/html/ supportmodulae.php?cat=s2.

[7] Paul Balister, Zizhan Zheng, Santosh Kumar, and Prasun Sinha. Trap coverage: Allowing coverage holes of bounded diameter in wireless sensor networks. In *In Proc. of IEEE INFOCOM, Rio de*, 2009.

[8] Mihir Bellare, Shafi Goldwasser, Carsten Lund, and A. Russeli. Efficient probabilistically checkable proofs and applications to approximations. In *STOC*, pages 294–304, 1993.

[9] Boaz Ben-Moshe, Matthew J. Katz, and Joseph S. B. Mitchell. A constant-factor approximation algorithm for optimal terrain guarding. In *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 515–524, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[10] Pravin Bhagwat, Bhaskaran Raman, and Dheeraj Sanghi. Turning 802.11 inside-out. *SIGCOMM Comput. Commun. Rev.*, 34(1):33–38, 2004.

[11] Vittorio Bilo, Ioannis Caragiannis, Christos Kaklamanis, and Panagiotis Kanellopoulos. Geometric clustering to minimize the sum of cluster sizes. In *ESA'05*, pages 460–471, 2005.

[12] Hervé Brönnimann and Michael T. Goodrich. Almost optimal set covers in finite vc-dimension. *Discrete & Computational Geometry*, 14(4):463–479, 1995.

[13] Jaroslaw Byrka and Karen Aardal. An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. *SIAM J. Comput.*, 39(6):2212–2231, 2010.

[14] Parinya Chalermsook and Julia Chuzhoy. Maximum independent set of rectangles. In *SODA*, pages 892–901, 2009.

[15] Timothy M. Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. In *Symposium on Computational Geometry*, pages 333–340, 2009.

[16] Moses Charikar and Sudipto Guha. Improved combinatorial algorithms for the facility location and k-median problems. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:378, 1999.

[17] Moses Charikar and Rina Panigrahy. Clustering to minimize the sum of cluster diameters. *J. Comput. Syst. Sci.*, 68(2):417–441, 2004.

[18] Fabin A. Chudak. Improved approximation algorithms for uncapacitated facility location (extended abstract). pages 180–194. Springer, 1998.

[19] V. Chvatal. A Greedy Heuristic for the Set-Covering Problem. *MATHEMATICS OF OPERATIONS RESEARCH*, 4(3):233–235, 1979.

[20] Gruia Călinescu, Howard Karloff, and Yuval Rabani. An improved approximation algorithm for multiway cut. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 48–52, New York, NY, USA, 1998. ACM.

[21] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiterminal cuts. *SIAM Journal on Computing*, 23:864–894, 1994.

[22] E.D. Demaine, Mitchell J.S.B., and O'Rourke J. The open problems project, problem 33: Sum of square roots.

[23] Erik D. Demaine, MohammadTaghi Hajiaghayi, and Philip N. Klein. Node-weighted steiner tree and group steiner tree in planar graphs. In *ICALP (1)*, pages 328–340, 2009.

[24] Srinivas Doddi, Madhav V. Marathe, S. S. Ravi, David Scot Taylor, and Peter Widmayer. Approximation algorithms for clustering to minimize the sum of diameters. In *SWAT*, pages 237–250, 2000.

[25] Partha Dutta, Sharad Jaiswal, Debmalya Panigrahi, K. V. M. Naidu, Rajeev Rastogi, and Ajay Kumar Todimala. Villagenet: A low-cost, 802.11-based mesh network for rural regions. In *COMSWARE*, 2007.

[26] Stephan Eidenbenz. Approximation algorithms for terrain guarding. *Information Processing Letters*, 82(2):99 – 105, 2002.

[27] Khaled M. Elbassioni, Erik Krohn, Domagoj Matijevic, Julián Mestre, and Domagoj Severdija. Improved approximations for guarding 1.5-dimensional terrains. In *STACS*, pages 361–371, 2009.

[28] Jeff Erickson and Sariel Har-Peled. Optimally cutting a surface into a disk. *Discrete & Computational Geometry*, 31(1):37–59, 2004.

[29] Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

[30] Jacob Fox and Janos Pach. Computing the independence number of intersection graphs. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, 2011.

[31] Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Multiway cuts in node weighted graphs. *Journal of Algorithms*, 50(1):49 – 61, 2004.

[32] Matt Gibson, Gaurav Kanade, Erik Krohn, Imran A. Pirwani, and Kasturi Varadarajan. On clustering to minimize the sum of radii. In *SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 819–825, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[33] Matt Gibson, Gaurav Kanade, Erik Krohn, Imran A. Pirwani, and Kasturi Varadarajan. On metric clustering to minimize the sum of radii. *Algorithmica*, 57:484–498, July 2010.

[34] Matt Gibson, Gaurav Kanade, Erik Krohn, and Kasturi R. Varadarajan. An approximation scheme for terrain guarding. In *APPROX-RANDOM*, pages 140–148, 2009.

[35] Sudipto Guha and Samir Khuller. Greedy strikes back: improved facility location algorithms. In *SODA '98: Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*, pages 649–657, Philadelphia, PA, USA, 1998. Society for Industrial and Applied Mathematics.

[36] Sariel Har-Peled. Being fat and friendly is not enough. *CoRR*, abs/0908.2369, 2009.

[37] Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In *STOC '02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 731–740, New York, NY, USA, 2002. ACM.

[38] Kamal Jain and Vijay V. Vazirani. Primal-dual approximation algorithms for metric facility location and k-median problems. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:2, 1999.

[39] Kamal Jain and Vijay V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *J. ACM*, 48(2):274–296, 2001.

[40] David S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci.*, 9(3):256–278, 1974.

[41] David R. Karger, Philip N. Klein, Clifford Stein, Mikkel Thorup, and Neal E. Young. Rounding algorithms for a geometric embedding of minimum multiway cut. *Math. Oper. Res.*, 29(3):436–461, 2004.

[42] James King. A 4-approximation algorithm for guarding 1.5-dimensional terrains. In *LATIN*, pages 629–640, 2006.

[43] James King. Vc-dimension of visibility on terrains. In *CCCG*, 2008.

[44] James King and Erik Krohn. Terrain guarding is np-hard, 2009.

[45] Philip N. Klein and R. Ravi. A nearly best-possible approximation algorithm for node-weighted steiner trees. *J. Algorithms*, 19(1):104–115, 1995.

[46] Madhukar R. Korupolu, C. Greg Plaxton, and Rajmohan Rajaraman. Analysis of a local search heuristic for facility location problems. In *SODA '98: Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*, pages 1–10, Philadelphia, PA, USA, 1998. Society for Industrial and Applied Mathematics.

[47] Santosh Kumar, Ten H. Lai, and Anish Arora. Barrier coverage with wireless sensors. In *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*, pages 284–298, New York, NY, USA, 2005. ACM.

[48] Nissan Lev-Tov and David Peleg. Polynomial time approximation schemes for base station coverage with minimum total radii. *Computer Networks*, 47(4):489–501, 2005.

[49] L. Lovsz. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13(4):383 – 390, 1975.

[50] Carsten Lund and Mihalis Yannakakis. The approximation of maximum subgraph problems. In *ICALP '93: Proceedings of the 20th International Colloquium on Automata, Languages and Programming*, pages 40–51, London, UK, 1993. Springer-Verlag.

[51] Mohammad Mahdian, Yinyu Ye, and Jiawei Zhang. Improved approximation algorithms for metric facility location problems. In *APPROX*, pages 229–242, 2002.

[52] Joseph S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric tsp, k-mst, and related problems. *SIAM J. Comput.*, 28(4):1298–1309, 1999.

[53] Nabil H. Mustafa and Saurabh Ray. Ptas for geometric hitting set problems via local search. In *Symposium on Computational Geometry*, pages 17–22, 2009.

[54] D. Panigrahi, P. Duttat, S. Jaiswal, K.V.M. Naidu, and R. Rastogi. Minimum cost topology construction for rural wireless mesh networks. pages 771 –779, apr. 2008.

[55] Jianbo Qian and Cao An Wang. How much precision is needed to compare two sums of square roots of integers? *Information Processing Letters*, 100(5):194 – 198, 2006.

[56] Bhaskaran Raman. Digital gangetic plains(dgp): 802.11-based low-cost networking for rural areas. Technical report, IIT, Kanpur, 2004.

[57] Bhaskaran Raman and Kameswari Chebrolu. Design and evaluation of a new mac protocol for long-distance 802.11 mesh networks. In *MOBICOM*, pages 156–169, 2005.

[58] Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. In *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 475–484, New York, NY, USA, 1997. ACM.

[59] John Reif. Minimum s-t cut of a planar undirected network in $o(n \log^2 n)$ time. *SIAM Journal on COmputing*, 12:71–81, 1983.

[60] Swaminathan Sankararaman, Alon Efrat, Srinivasan Ramasubramanian, and Javad Taheri. Scheduling sensors for guaranteed sparse coverage. *CoRR*, abs/0911.4332, 2009.

[61] Sayandeep Sen and Bhaskaran Raman. Long distance wireless mesh network planning: problem formulation and solution. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 893–902, New York, NY, USA, 2007. ACM.

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

[63] Maxim Sviridenko. An improved approximation algorithm for the metric uncapacitated facility location problem. In *IPCO*, pages 240–257, 2002.

[64] Kasturi R. Varadarajan. Epsilon nets and union complexity. In *Symposium on Computational Geometry*, pages 11–16, 2009.