Spring 2017

Multi-covering problems and their variants

Santanu Bhowmick
University of Iowa

MULTI-COVERING PROBLEMS AND THEIR VARIANTS

by

Santanu Bhowmick

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

May 2017

Thesis Supervisor: 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

Santanu Bhowmick

has been approved by the Examining Committee for the thesis requirement for the Doctor of Philosophy degree in Computer Science at the May 2017 graduation.

Thesis Committee: ______________________________
Kasturi Varadarajan, Thesis Supervisor

____
Sriram Pemmaraju

____
Sukumar Ghosh

____
Samuel Burer

____
Aaron Stump

ACKNOWLEDGEMENTS

Working on my Ph.D. has been one of the most rewarding experiences of my life. I'm indebted to many people for making this journey an unforgettable one.

First and foremost, I'd like to thank my advisor, Professor Kasturi Varadarajan, for his guidance and encouragement. His endless patience and didactic attitude, combined with his wide knowledge, makes him a wonderful advisor. It has been a great honor to have Professor Varadarajan as my advisor, without whose expertise and persistent help this dissertation would not have been possible.

I would like to thank my co-authors, Sayan Bandyapadhyay and Tanmay Inamdar, for their invaluable insights into the problems that we worked on together. Life at the university would have been significantly less interesting without the discussions we had about algorithms and theory, both in the Algorithms Reading Group that we had and otherwise.

I owe a debt of gratitude to the faculty at Indian Statistical Institute, Kolkata, who played a crucial role in my academic career and kindled the desire to embark on my doctoral journey.

I would like to thank my parents for their invaluable support throughout my life, without which I would not be here. Last but not the least, I'm grateful to my lovely wife, Sanchita, who has been with me all these years and has made them the best years of my life.

ABSTRACT

In combinatorial optimization, covering problems are those problems where given a ground set and a family of subsets of the ground set, the objective is to find a minimum cost set of subsets whose union contains the ground set. We consider covering problems in the context of Computational Geometry, which is a subfield of Computer Science that deals with problems associated with geometric objects such as points, lines, disks, polygons etc. In particular, geometric covering is an active research area, where the ground set and the family of subsets are induced by geometric objects. Covering problems in combinatorial optimizations often have a geometric analogue that arises naturally, and though such problems remain NP-hard, it is often possible to exploit the geometric properties of the set system to obtain better approximation algorithms.

In this work, the fundamental problem that we consider is a generalization of geometric covering, where each element in the ground set may need to covered by more than one subset. To be precise, the problem is defined as follows: given two sets of points X, Y and a coverage function $\kappa : X \to \mathbb{Z}_+ \cup \{0\}$, construct balls centered on the points in Y such that each point in X is covered by at least $\kappa(x)$ distinct balls. The objective in this problem is to minimize the total cost, which is a function of the radii of the balls. This problem is termed as the *metric multi-cover* (MMC) problem.

We first consider version of the MMC problem where $\kappa(x) = 1$ for all clients, i.e. the 1-covering case. The known results that give a constant factor approximation for this problem are variations of LP-based primal-dual algorithm. We use a modified local search technique, motivated by geometric idea, to derive a simple, constant-factor approximation for this problem in Chapter 2.

We then look at the MMC problem where the point sets X, Y are in the Euclidean plane, and each client $x \in X$ needs to be covered by at least $\kappa(x)$ distinct disks centered on the points in Y. In Chapter 4, we give the first polynomial time constant factor approximation for this problem, in which the constant is independent of the coverage function κ. Our solution also has an *incremental* property, which allows the algorithm to handle increases in the coverage requirement by increasing the radii of the current server disks, without affecting the approximation factor.

In the next problem, we move from the Euclidean plane to arbitrary metric spaces where we consider the *uniform* MMC problem. In this problem, each client x has the demand $\kappa(x) = k$, where $k > 0$ is an integer. We give the first constant factor approximation (independent of k) for this problem. The key contribution that led to this result is the formulation of a *partitioning* scheme of the servers in the uniform MMC problem, that reduces the uniform MMC problem to k instances of 1-covering problem, while preserving the optimality of the solution to a constant multiplicative factor. We present the partitioning scheme as an independent result in Chapter 5, which we then use to solve the uniform MMC problem in Chapter 6.

The MMC problem with arbitrary coverage function κ is then considered in Chapter 7. The key challenge that the non-uniform version presents is that the symmetry of the server partitioning scheme breaks down as the coverage demands of

clients are independent of each other. We present a constant factor algorithm for this problem in Chapter 7.

The last problem that we consider is the t-MMC problem, which is a restricted version of the uniform MMC problem. The objective is to compute a cover in which each client is covered by at least k distinct server disks, using *atmost* t server disks in total. This problem is a generalization of the clustering problem (where $k = 1$), and to our knowledge this is the first time this generalization has been considered. We give a constant factor approximation for this problem in Chapter 8.

PUBLIC ABSTRACT

In this dissertation, we examine the metric multi-cover (MMC) problem and its variants. We are given two point sets X (clients) and Y (servers) in a metric space and a non-negative integer k. The goal is to find an assignment of radii to the servers, such that each client is covered by at least k disks centered at the servers, while minimizing the sum of the area of the disks. The MMC problem arises in the design of wireless sensor networks, where each sensor has a limited energy source. The energy consumed is considered to be linear in the area of the region covered by each sensor. Some applications may require multiple sensors monitoring an area for event detection, whereas in other applications, security or fault-tolerance requirements may mandate a multi-cover of the clients. Prior works described algorithms that gave an O(k) approximation to the problem, i.e., the cost of the solutions obtained increased linearly with k, making them impractical for high values of k.

In this work, we derive constant-factor approximations for the MMC problem, in which the constant factor guarantee is independent of the demand k. We extend the same approximation guarantee for some variants of the MMC problem, which include a version having non-uniform coverage requirement of the clients, and one in which there is a restriction in the number of servers that can be used. We also give a simple geometric constant-factor approximation for regular covering, in order to facilitate deeper understanding of metric covering problems.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Background

This dissertation examines certain fundamental geometric optimization problems that lie in the intersection of the areas of clustering and covering. An example of a clustering problem is the following: given a set X of points in an arbitrary metric space and an integer k, and the goal is to cover the points by k balls so as to minimize the sum of radii of the balls. In a typical covering problem, we are given a set P of points in the plane and a set D of geometric objects (e.g. disks, rectangles or triangles) that contain the points in P, and the objective is to find a minimum cardinality subset of D whose union contains P. Clustering is used for inferring useful patterns from data in fields as diverse as natural sciences, psychology, medicine, engineering, economics and marketing. Geometric covering problems arise in many contexts, such as wireless and sensor networks and surveillance of a set of objects or an area. Most versions of clustering and covering problems are $\mathbb{NP}$-hard i.e. under widely believed assumptions, it is not possible to obtain an *optimal* solution for every instance of the problem efficiently. Hence, in this dissertation, we examine clustering and covering problems with the focus being on obtaining efficient *approximation algorithms*. (An f-approximation algorithm gives a solution whose cost is at most f times the cost of an optimal solution).

Before proceeding further, we define some basic concepts pertaining to combinatorial optimization problems. An instance of an optimization problem has a set of valid solutions, known as its *feasible* solutions. Associated with each feasible solution is a numeric value known as its weight. In a minimization problem (which is what we confine ourselves to in this thesis), the objective is to find a feasible solution that has minimum weight, which is denoted as its *optimal* solution. For example, an instance of the Set Cover problem consists of a ground set $\mathcal{X}$ and a set of subsets $\mathcal{F}$ of $\mathcal{X}$. Each set $S \in \mathcal{F}$ has a non-negative weight $w(S)$ associated with it - if $w(S) = 1$ for all $S \in \mathcal{F}$, we have the *unweighted* Set Cover Problem. A feasible solution to a set cover instance is a collection of sets $T \subseteq \mathcal{F}$ such that $\cup_{S \in T} S = \mathcal{X}$. Such a feasible solution is called a *set cover*, whose weight is the sum of the weights of the sets in the cover. The objective is to find a minimum weight set cover for any instance $(\mathcal{X}, \mathcal{F}, w)$. If the problem is $\mathbb{NP}$-hard, then an optimal solution cannot be determined in polynomial time unless $\mathbb{P} = \mathbb{NP}$. In such cases, we seek efficient algorithms that compute near optimal solutions, known as *approximation algorithms*. A c-approximation algorithm for a minimization problem returns a feasible solution whose weight is at most c times that of an optimum solution. An algorithm that for any $\varepsilon > 0$, computes a $(1 + \varepsilon)$ approximation for an optimization problem is termed as a *Polynomial Time Approximation Scheme* (PTAS).

We begin with a brief description of the history and origins of the Set Cover problem, leading to its consideration in the geometric setting.

Set Cover is one of the 21 $\mathbb{NP}$-complete problems presented in [49], and its study has led to the development of fundamental techniques for the entire field of approximation algorithms [64]. The standard greedy heuristic for set cover was shown to have an $O(\log n)$ approximation (where $n = |\mathcal{X}|$) [48]. A series of subsequent results [14, 33, 57, 59] have established that Set Cover is hard to approximate to better than a log-factor (under standard complexity-theoretic assumptions) i.e. the set cover problem cannot be approximated to within a $o(\log n)$ factor unless $\mathbb{P} = \mathbb{NP}$, thus settling the computational complexity for the set cover problem.

The Set Cover problem becomes more tractable in geometric settings i.e. when the the family of subsets $\mathcal{F}$ are induced by geometric objects (such as disks, half-planes, triangles etc) - for example, covering points using disks in the Euclidean plane. Most geometric set cover problems remain $\mathbb{NP}$-hard, and there are instances in which it is hard to approximate as well [30, 42]. However, it is possible to get a better approximation factor for several versions of the geometric set cover. Some of these improved results have been achieved by exploiting the combinatorial properties that are specific to the set system [15, 53]. Several other results have relied on more generic properties of the geometric set system (such as low VC dimension or small union complexity). A sequence of results [20, 27, 31, 44] led to the formulation of $O(\log |OPT|)$ approximation algorithms for (unweighted) geometric set cover using objects such as disks, triangles and axis-parallel rectangles, where $|OPT|$ denotes the cardinality of an optimal solution. Clarkson and Varadarajan [28] subsequently proved that exploiting the union complexity of the set system gives improved approximation bounds for covering using fat objects (fat objects are objects for which ratio of its smallest enclosing circle and largest enclosed circle is bounded by a constant). Specifically, they gave a $O(\log\log|OPT|)$ approximation guarantee for fat triangles and a constant factor approximation for pseudodisks. For fat triangles, a series of improvements [5, 62]) followed, culminating in the current best approximation guarantee of $O(\log\log^*|OPT|)$ given by Ezra et al. [32]. For covering points using minimum number of disks in the plane, the approximation guarantee had been improved recently by Mustafa and Ray [58], who showed that a simple local search based PTAS exists for the problem.

A generalization of the Set Cover problem is the Set Multi-cover problem, in which each element x in the ground set $\mathcal{X}$ has a positive demand $\kappa(x)$. A subset $S \in \mathcal{F}$ is a feasible multi-cover if each $x \in \mathcal{X}$ is covered by at least $\kappa(x)$ distinct sets in $\mathcal{F}$. It is well known that the natural greedy algorithm yields a logarithmic approximation for Set Multi-cover problem as well. For the case of unweighted Set Multi-Cover Problem using geometric objects of finite VC-dimension, Chekuri et al. [24] gave a $O(\log|OPT|)$ approximation. They also gave constant factor approximations for multi-covering points in the plane using (unweighted) pseudo-disks and other fat objects. Bansal and Pruhs [10] built on the techniques in [22, 63] to give an $O(1)$ approximation for the **weighted** Set Multi-Cover Problem for covering points using disks on the plane (and other set systems having low union complexity).

In the geometric covering problems discussed so far, the family of subsets are "immutable", and the complexity of the problem stems from the combinatorial geometry of the objects. Har-Peled and Lee [43] consider a version of covering points using disks in the plane, where the disks are allowed to expand by a fixed fraction specified as an input parameter. The authors give a PTAS for this problem. A more "flexible" covering problem (in terms of flexibility to alter the size of the subsets in the family $\mathcal{F}$) is the *minimum cost covering* (MCC) problem, which is described as follows.

Before defining the MCC problem formally, we briefly define some of the associated terminology. A metric space on a set of points P is the ordered pair (P, d) where $d : P \times P \rightarrow \mathbb{R} \cup \{0\}$ is a metric i.e. for any $x, x', z \in P$, the following holds: (1) $d(x, x') = d(x', x)$, and (2) $d(x, z) \leq d(x, x') + d(x', z)$. A *ball* $\delta(x, r)$ is the subset of a set of points P whose distance from x is at most r i.e. $\delta(x, r) = \{x' \in P \mid d(x, x') \leq r\}$, where x is denoted as the center of the ball and r is the radius of the ball. The input to the MCC problem are two point sets, X (clients) and Y (servers) and a metric d on $X \cup Y$. For $z \in X \cup Y$ and $r > 0$, the ball $\delta(z, r)$ is the set $\{z' \in X \cup Y \mid d(z, z') \leq r\}$. A cover for X is a set of balls centered on the points in Y (servers), the union of which contains X. The *cost* of a ball $\delta(y, r)$ is r^α, where $\alpha \geq 1$ is a parameter of the MCC problem (but not of a problem instance). The objective of the MCC problem is to find a cover whose cost is minimum.

Motivation for studying the MCC problem lies in the domain of wireless network design. The servers i.e points in Y can be considered to be the locations of mobile towers, and the points in X can represent (possible) locations of clients. Each tower $y \in Y$ can be configured such that it covers all clients within a certain distance $r(y)$, and the service cost $r(y)^\alpha$ increases with the distance - this corresponds to the additional transmission energy needed to serve more distant points. For modeling realistic wireless networks, the value of α is usually considered to be at least 1.

The MCC problem has been extensively studied [3, 8, 19, 54] - we would discuss the results in detail in Section 1.2. A natural extension to the MCC problem lies in the domain of fault-tolerant wireless network design. Assume that a set of mobile users need to have access to a cellular network. A communications company has a set of potential locations where it can place its base stations. Each base station has an operational cost that is proportional to the area covered by that station. Further, each client may need to be covered more than once, the coverage requirement of each client being representative of its priority or level of service desired. Establishment of base stations that minimizes the operating cost while providing adequate coverage to the users leads to the formulation of the fundamental optimization problem considered in this thesis, which we formally define as follows.

Definition 1. Given are a set of points X (clients), a set of points Y (servers), a metric d on the set $X \cup Y$, a coverage function $\kappa : X \to \mathbb{Z}_+ \cup \{0\}$ and $\alpha \geq 1$. An assignment $r : Y \to \mathbb{R}^+$ corresponds to "building" a disk of radius r_y centered at each $y \in Y$. For any positive integer j, we say a client $x \in X$ is j-covered if x is contained in at least j server disks i.e.

$$|\{y \in Y \mid d(y, x) \leq r(y)\}| \geq j.$$

The goal is to find an assignment r that $\kappa(x)$ covers each client $x \in X$ and minimizes $\sum_{y \in Y} r_y^{\alpha}$. We call this the *Metric Multi-Cover* (MMC) problem.

Notice that the MCC problem is a special case of the MMC problem, where $\kappa(x) = 1$ for each client $x \in X$. The MMC problem also arises naturally in the design of wireless sensor networks, where the energy source of each sensor is often limited to an attached battery cell. The energy consumed can be considered to be linear in the area of the region covered by the corresponding sensor. Some applications of such networks may require multiple sensors monitoring an area for event detection, whereas in other applications, security or fault-tolerance requirements may mandate a multi-cover of the client set. The MMC problem was shown to be $\mathbb{NP}$-hard even for $\kappa(x) = 1, \forall x \in X$, $\alpha > 1$, the set of clients and servers being points on the Euclidean plane $\mathbb{R}^2$ [3, 19].

It would seem that the MMC problem is a special case of the Set Multi-Cover problem, using the following reduction from MMC to the Set Multi-Cover problem: for each $y \in Y$ and $x \in X$, add a ball centered at y with radius $d(x, y)^\alpha$, and let X be the set of points that need to be covered. The reason this reduction does not work is that we have to add an additional constraint saying that we can use only one ball centered at each $y \in Y$. Notice that this additional constraint is not an issue for the case $k = 1$, since here if the returned solution uses two disks centered at the same $y \in Y$, we can simply discard the smaller one. Since MMC is not a case of the Set Multi-Cover, we cannot use the result of Bansal and Pruhs [10] for multi-covering using disks in the plane for the MMC problem in the plane.

We consider multiple variants of the MMC problem, in increasing order of complexity. We first look at the MMC problem where the underlying metric space is the plane i.e. the point sets X, Y are points on the Euclidean plane, and the distance between any two points is the Euclidean distance between two points. We refer to this version of the problem as the *planar multi-covering* problem (PMC), for which we give the first constant factor algorithm in which the constant is independent of the coverage function κ. Our approximation algorithm has an interesting property as well, in that it can handle incremental increases in the coverage requirement by simply increasing the radii of one or more disks in the erstwhile cover, while staying with the approximation guarantee.

The next version of the MMC problem that we consider is the *uniform* MMC problem, in which each client has a demand of k, the latter being a positive integer. We give a constant factor approximation for this problem, the crux of our contribution being the formulation of a partitioning scheme that reduces the problem to several instances of the MCC problem while preserving optimality up to a multiplicative constant factor. Our approach is robust enough to generalize to the MMC problem, where each client has a possibly different demand. Using some additional ideas, we obtain an $O(1)$ approximation for this problem as well.

We also consider a variant of the MMC problem that shares some key features with *clustering* problems, which we discuss next. **Clustering** is a ubiquitous problem that arises naturally in applications as diverse as data mining, image processing, machine learning and bioinformatics. The input to a clustering problem instance is a set X of n points in a *metric space* (X, d) and a positive integer t. The objective is to cover the points of X using at most t balls centered on points in X, such that the cost of the cover is minimized. The cost of a cover is sum of the cost of the balls in the cover, where the cost of a ball is defined in exactly the same way as in the case of MCC i.e, if the radius of a ball is r, its cost is r^α, where $\alpha \geq 1$. This problem is known as the t-clustering problem, and it is one of the most well-studied optimization problems both from theoretical and practical perspectives [8, 19, 23, 29, 37, 38]. The t-clustering problem can be seen as a variant of the MCC problem, where the set of servers Y is the same as the set of clients X, with the added restriction of using at most t balls in the solution. Motivated by this problem, we consider the following generalization in the multi-covering domain.

Definition 2. Given are a set of points X (clients), a set of points Y (servers), a metric d on the set $X \cup Y$, a coverage function $\kappa : X \rightarrow \mathbb{Z}_+ \cup \{0\}$, a positive integer t and a constant $\alpha \geq 1$. The goal is to find an assignment r that $\kappa(x)$ covers each client $x \in X$ and minimizes $\sum_{y \in Y} r_y^\alpha$, using at most t balls. We call this the t-MMC problem.

Finally, we re-visit the MCC problem from the standpoint of techniques used to give an approximate solution. For arbitrary metric spaces, the constant factor algorithms that are known are based on the primal-dual scheme [12, 13, 23]. Motivated by geometric ideas, we give a constant factor approximation for the MCC problem using an adaptation of the Local Search technique.

We now give an overview of the thesis structure as a roadmap for the reader.

1.1.1 Thesis Structure

In this document, we look at approximation algorithms for the MMC problem and its variants. We first provide a local search algorithm for the MCC problem in Chapter 2 as an alternative to the LP relaxation based approach that is known in the literature. We consider the multi-covering problems next. An important concept for multi-covering problems is the concept of an *outer-cover*, which is a 1-cover in which the balls used to cover clients have additional constraints. We discuss this concept in Chapter 3. We consider the problem of multi-covering in the plane (Planar Multi-Covering) in Chapter 4, which uses the concept of an outercover. For multi-covering in an arbitrary metric space, we introduce the concept of partitioning a multi-covering problem into several instances of the MCC problem while staying within a constant multiplicative factor of the optimal cover. Chapter 5 describes the partitioning algorithm in detail, while Chapter 6 and Chapter 7 use the partitioning algorithm to give constant factor approximations for the uniform and non-uniform multi-cover problem, respectively. The t-MMC problem is considered in Chapter 8, where we give the first constant factor approximation for it by using ideas presented in Chapter 5 and deriving a modified outer-cover bound. Finally, we give a summary of the open problems posed by our thesis in Chapter 9.

In the rest of this chapter, we look at the problems considered in this thesis. For each problem, we provide additional background, review previous work and describe our contributions to the problem.

1.2 A Local Search Algorithm for the MCC Problem

We look at the MCC problem from the point of view of techniques used to solve the problem. Specifically, our objective is to find a constant factor approximation for the MCC problem using a simple geometric approach, without using LP-based methods. To understand why we look for a combinatorial algorithm for the MCC problem, let us consider the closely related problem of *metric facility location* (MFL), that is defined as follows.

Definition 3. We are given two point sets F (facilities) and C (clients). Each facility $f \in F$ has an opening cost $cost(y)$ and connecting a client $c \in C$ to an open facility f incurs a cost $\bar{d}(c, f)$. A feasible solution consists of a subset of open facilities $O \subseteq F$, in which each client is connected to the nearest open facility in O. The cost of any feasible solution O is the sum of opening costs of the facilities in O and the connection costs of each client $c \in C$. The objective is to find a feasible solution of minimum cost. When the connection costs of the clients form a metric, this problem is known as the metric facility location problem.

The *metric facility location* (MFL) problem is a classical $\mathbb{NP}$-hard optimization problem that has been extensively studied [64]. Under standard complexity assumptions, Guha and Khuller [39] showed that MFL cannot have an approximation guarantee of 1.463 or better.

The main distinction between the two problems is in the nature of the connection costs - for MFL problem, the connection cost incurred by each server is the sum of the connection costs of the individual clients connected to it, whereas for the MCC problem, the connection cost charged to a server is the maximum connection cost amongst the clients connected to that server. Informally, while each open facility is compelled to pay the connection costs for each client that it serves, the servers in the

MCC problem only pay for the connection cost of the **farthest** client that it serves. This aspect of the MFL problem influences the algorithmic techniques that can be used to derive approximations for it.

1.2.1 Previous Work

For MFL problem, the two main algorithmic paradigms that have been used to derive constant factor approximations are (a) standard linear programming (LP) relaxation techniques, such as filtering [56], randomized rounding [25, 60], primal-dual framework [46], and dual fitting [47], culminating in the current best approximation guarantee of 1.488 [55], and (b) combinatorial techniques such as Local Search [7, 52]. Both these approaches have yielded constant factor polynomial time approximations for the MFL problem.

We focus on the Local Search technique, which has proved to be a popular heuristic for hard combinatorial problems due to their simplicity. Such a heuristic starts out with a *initial* feasible solution, then iteratively improves it by "simple" changes to the existing solution till no further improvement is possible. If I be an instance of the minimization problem, let global(I) denote the cost of the global optimum solution whereas the cost of the solution returned by a local search heuristic be denoted by local(I). The supremum of the ratio local(I)/global(I) is termed as the *locality gap* of the local search procedure.

In the context of the MFL problem, an example of an operation that alters the existing solution is addition, deletion or swap of an open facility with a closed facility. Korupolu et al. [52] showed that a local search algorithm using such an operation in the improvement step has a locality gap of at most 5. The analysis of this local search algorithm was subsequently improved by Arya et al. [7] who showed that the locality gap is at most 3 and that the gap is tight.

Prior to our work on the MCC problem, the only known polynomial time constant factor approximations used variations of the primal-dual framework [13, 23] to give a 3^α approximation. Such LP relaxation based approaches are difficult to extend to solve variants of the MCC problem (such as involving capacitated servers for example). We note that even when the underlying metric is the Euclidean norm, a combinatorial constant factor approximation is known only for the case $\alpha \geq 2$ [1]. This approach exploits the fact that for $\alpha = 2$, the objective function is trying to minimize the sum of the areas of the disks. We note that despite the ostensible similarities between the two problems MFL and MCC, the latter has not been amenable to any local search based approximation algorithms prior to our work.

1.2.2 Our Contribution

Our main contribution is a polynomial time algorithm that gives a constant factor approximation for the MCC problem, without using a LP relaxation based approach. We give a simple geometric algorithm that modifies the local search technique for the MCC problem. This is a joint work with my advisor, Kasturi Varadarajan.

We first discuss why the standard local search algorithm does not work for the MCC problem, in order to develop the intuition needed for the modification.

Any feasible solution for an instance of the MCC problem consists of balls drawn from the set $\mathcal{B}$ of at most $|X| \times |Y|$ distinct balls. For the MCC problem, traditional local search works as follows - we start with a feasible solution $C \subseteq \mathcal{B}$, and we see if the cost of the solution can be decreased with small changes to the cover i.e. by bringing in a constant number of balls in C and removing a constant number of balls from C while retaining feasibility of the cover. Let $k > 0$ be the integer that represents the constant for the standard local search algorithm. Then, a straightforward implementation of such a scheme would check all possible $O(n^{2k})$ $2k$-tuples, incurring a naive run time of $O(n^{2k})$. Such an approach yields a polynomial time algorithm only if k is a constant independent of $|X|, |Y|$.

The aforementioned approach yields constant factor approximations for the MFL problem, but fails to do so for the MCC problem. To see why, we consider the following example depicted in Figure 1.1. The filled (blue) circles represent clients in X, while the (red) crosses represent the servers in Y, and let $\alpha = 1$. The distance between the central server and the clients is r, while the distance between any other server and its adjacent client is an arbitrarily small number ϵ. The optimal cover in this configuration would consist of the balls of radius ϵ, centered at each of the servers in the periphery, incurring a total cost of $n \cdot \epsilon$ (assuming $r >> n \cdot \varepsilon$). Now, consider the iteration in the traditional local search algorithm when the feasible solution consists of the single ball centered on the server in the center, whose cost is r. This cost of this solution cannot be decreased if the algorithm considers less than n balls in a single iteration, which makes it an infeasible approach. This example thus suggests that a successful approach would consider all cheap balls before considering the inclusion of an expensive ball, instead of adding them in any arbitrary order. We discuss this approach (of considering the balls in a certain order determined by their size) in Chapter 2.

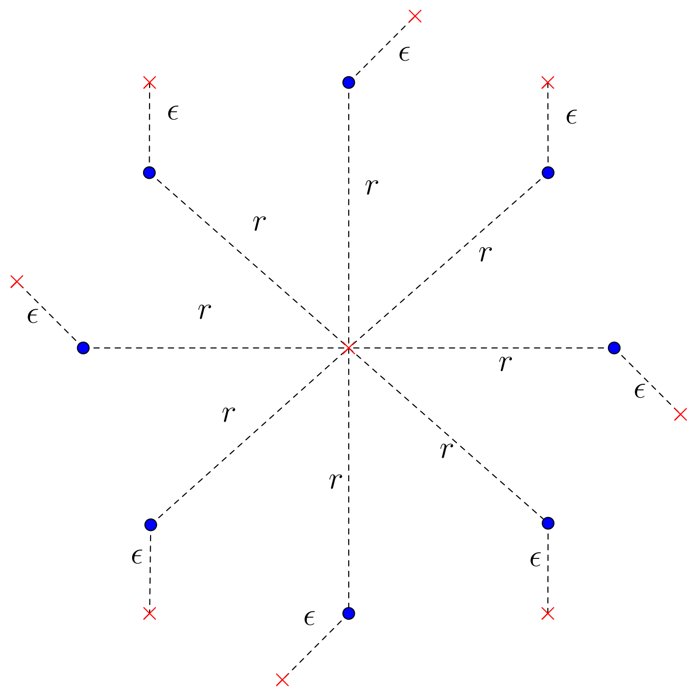

Figure 1.1: Clients are filled (blue) circles, servers are (red) crosses, and the labels represent inter-point distances, where $r \in \mathbb{R} \cup \{0\}, \epsilon > 0$. If a feasible solution includes the ball centered on the server in the middle, then an improvement to that solution can only be made if the local search algorithm considers all balls of radius ϵ centered on the remaining servers in a single improvement step.

1.3 The Planar Multi Covering (PMC) Problem

We are given two point sets Y (servers) and X (clients) in the plane, a coverage function $\kappa : X \to \mathbb{N}$, and a constant $\alpha \geq 1$. An assignment $r : Y \to \mathbb{R}^+$ of *radii* to the points in Y corresponds to "building" a disk of radius r_y centered at each $y \in Y$. For an integer $j \geq 0$, let us say that a point $x \in X$ is j-covered under the assignment if x is contained in at least j of the disks, i.e.

$$|\{y \in Y \mid ||y - x||_2 \leq r_y\}| \geq j$$

The goal is to find an assignment that $\kappa(x)$-covers each point $x \in X$ and minimizes $\sum_{y \in Y} r_y^\alpha$. We call this the *planar multi cover* (PMC) problem, for which we want to obtain a constant factor approximation algorithm.

The version of this problem where $\kappa(x) = k$, $\forall x \in X$, for some given $k > 0$, has received particular attention. Here, all the clients have the same coverage requirement of k. We will refer to this as the *uniform PMC* problem. In the context of the uniform PMC, we will refer to a j-cover as an assignment of radii to the servers under which each client is j-covered.

1.3.1 Previous Work

The (uniform) PMC problem was considered in two recent papers, motivated by fault-tolerant sensor network design that optimizes energy consumption. Abu-Affash et al. [1] considered the case $\alpha = 2$, which corresponds to minimizing the sum of the areas of the server disks. They gave an $O(k)$ approximation for the problem using mainly geometric ideas. Bar-Yehuda and Rawitz [13] gave another algorithm that achieves the same approximation factor of $O(k)$ for any α, using an analysis based on the local ratio technique.

1.3.2 Our Contribution

In this article, we obtain an $O(1)$ approximation for the PMC problem. That is, we demonstrate an approximation bound that is independent of κ. This is a joint work with Shi-Ke Xue and Kasturi Varadarajan [16, 17].

In the interests of clarity, we briefly describe our approach for the simpler case of the uniform PMC problem, where $\kappa(x) = k$ for all clients $x \in X$. Our approach revolves around the notion of an *outer cover*. This is an assignment of radii to the servers under which each client $x \in X$ is covered by a disk of radius at least $||y^k(x) - x||_2$, where $y^k(x)$ is the k-th nearest neighbor of x in Y. To motivate the notion, consider any k-cover, and in particular, the optimal one. Consider the set of disks obtained by picking, for each client $x \in X$, the largest disk covering x in the k-cover. (Several clients can contribute the same disk.) This set of disks is seen to be an outer cover.

We provide a mechanism for extending any $(k-1)$-cover to a k-cover so that the increase in objective function cost is bounded by a constant times the cost of an optimal outer cover. This naturally leads to our algorithm in Section 4.2 – recursively compute a $(k-1)$-cover and then extend it to a k-cover. To bound its approximation ratio, we argue that the optimal solution can be partitioned into a $(k-1)$-cover and another set of disks that is almost an outer cover. Finally, we need a module for computing an approximately optimal outer cover. We show in Chapter 3 that an existing primal-dual algorithm for 1-covering can be generalized for this purpose.

The idea of an outer cover has its origins in the notion of *primary disks* used by Abu-Affash et al. [1]. Our work extends on this idea in a significant manner, and this is partly what enables our $O(1)$ approximation bound.

1.4 The Metric Multi-Cover Problem

The Metric Multi-Cover (MMC) problem is a generalization of the PMC problem described in the previous section, such that the input point sets X, Y are set in an arbitrary metric space $(X \cup Y, d)$. We have defined the problem formally in Definition 1, and we present the first constant factor approximation (independent of the coverage function κ) in this document. The results on metric multi-cover are joint work with Tanmay Inamdar and Kasturi Varadarajan [18].

1.4.1 Previous Work

Bar-Yehuda and Rawitz [13] were the first to give an approximation algorithm for the MMC problem in which X, Y are points in an arbitrary metric space. They presented a $3^\alpha \cdot k$ approximation guarantee, using the local-ratio technique. They also consider the non-uniform version of the problem, where the coverage demand of each client is an arbitrary integer that is not necessarily related to the demands of the other clients. They obtain a $3^\alpha \cdot k_{\mathtt{max}}$ approximation for this version, where $k_{\mathtt{max}}$ is the maximum client demand. Their guarantees also hold for minimizing a more general objective function $\sum_{y \in Y} (w_y r(y))^\alpha$, where weight $w_y \geq 0$ is specified for each server y as part of the input; we do not address this objective function here.

Fault tolerant versions of other related problems have also been studied in the literature – facility location [21, 40, 61], t-median [41], and t-center [51]. Constant factor approximations are known for all these problems in the metric setting. In particular, the results for facility location solve the natural LP-relaxation and perform a clever rounding. These rounding methods do not readily extend to our setting. One reason for this is the fact that we are dealing with a covering problem; another reason is the additional constraint that one has to write in the LP saying that each server can house at most one ball.

1.4.2 Our Contribution

We present a polynomial-time algorithm that reduces the MMC to several instances of the 1-covering version, where $k = 1$. This reduction preserves optimality to within a constant multiplicative factor. More specifically, our reduction outputs *pairwise disjoint* subsets $Y_1, Y_2, \ldots, Y_k$ of servers such that computing an optimal 1-cover of the clients X using each Y_i and combining the 1-covers results in a solution whose cost is $O(1)$ of the optimal cost.

Using a known constant factor approximation algorithm for computing a 1-cover, we obtain an $O(1)$ approximation for the MMC problem in any metric space, achieving a guarantee that is independent of the coverage demand k. This resolves a problem left open by Bhowmick et al. [17], whose approximation guarantee in the Euclidean setting depends on the dimension. Concretely, our approximation guarantee is $2 \cdot (108)^\alpha$. We have not attempted to optimize the constants, as our focus is on answering the question of whether a guarantee independent of k and the dimension is possible.

This result represents a major advance over [17] in our understanding of the MMC problem. To explain this, we first reiterate the overall approach of [17]. They first compute covers ρ_i for X, for $1 \leq i \leq k$. Each ρ_i is actually a special type of cover called a level i *outer cover*, as described precisely in Chapter 3. They show that $\sum_{i=1}^{k} \texttt{cost}(\rho_i)$ is, up to a constant factor, a lower bound on the cost of the optimal solution to the MMC. Note that a server in Y may contribute a ball to many of the ρ_i, so the union of the ρ_i is not a feasible solution to the MMC.

Their algorithm for computing a k-cover is recursive – it first computes a $(k-1)$-cover and then extends it to a k-cover. They bound the increase in cost in going from an $(i-1)$-cover to an i-cover by $c_d \cdot \texttt{cost}(\rho_i)$, where c_d is a constant that depends on the dimension d and α. To extend their approach to the metric setting, one would have to bound the increase in cost in going from an $(i-1)$-cover to an i-cover by $c \cdot \texttt{cost}(\rho_i)$, where c is a constant (that depends only on α).

This is not possible, as the following “high-dimensional” example shows. Suppose that k, the coverage requirement, is even and $\alpha = 1$. Let

$$X = Y = \{\pm e_j \mid 1 \leq j \leq k/2\},$$

where e_j is the point in $\mathbb{R}^{k/2}$ with 1 in the j-th coordinate and 0 in the other coordinates. Thus, each of the k points is both a client and a server; each client has $k-2$ points at distance $\sqrt{2}$ from it, and one ‘antipodal’ point at distance 2 from it. Here, we can let ρ_i, the level i outer cover, for each i, to be the singleton set $\{\delta(e_1, 2)\}$, where $\delta(p, r)$ denotes the ball of radius r centered at p.

Now suppose the $(k-1)$-cover we have at hand is $\{\delta(y, \sqrt{2}) \mid y \in Y\}$. That is, the radius assigned to each $y \in Y$ is $\sqrt{2}$. Now an optimal k-cover is $\{\delta(y, 2) \mid y \in Y\}$, since in any k-cover each client needs to be contained in the ball centered at its antipodal server. Thus the increase in cost incurred by the algorithm in going from the $(k-1)$-cover to a k-cover is at least $(2-\sqrt{2})k$, which is larger than $\texttt{cost}(\rho_k) = 2$ by a multiplicative factor of $\Omega(k)$. Thus, the curse of dimensionality afflicts the analysis framework of [17].

In this article, we take a different approach. We extract k *disjoint* subsets $Y_1, Y_2, \ldots, Y_k$ of Y. For each subset Y_i, we compute a set of balls centered at Y_i that covers X, using as a black-box any constant factor approximation algorithm for 1-covering [23, 34]. This gives us k covers that result in a feasible solution for the MMC problem, as the k server subsets are disjoint.

To obtain an approximation guarantee using this approach, we require that the disjoint subsets $Y_1, Y_2, \ldots, Y_k$ satisfy the following property: for each client from some representative family, Y_i should contain at least one server from among the i servers nearest to the client. This concept appears to be of interest beyond its application to the MMC problem. Notice that the property becomes stricter as i gets smaller. A bad choice of servers Y_i can mean that there is no feasible choice of Y_j for $j < i$. Thus it is not clear that the desired family of k subsets exists. One of our main technical contributions is a constructive proof that a family satisfying some closely related property does indeed exist - this has been extracted into a separate result in Chapter 5.

Having constructed the k disjoint subsets, we bound the approximation factor by first showing that the cost of an optimal cover of X using servers Y_i is within a constant of the cost of any i-level outer cover. This is straightforward given the property that Y_i satisfies. To finish, we use the result of [17], that there exists an i-level outer cover ρ_i for each $1 \leq i \leq k$ such that sum of the costs of these k outer covers serves as a lower bound for the optimal MMC solution. Notice that unlike [17], we use the notion of outer covers only in the analysis and not the algorithm itself.

Our approach is robust enough to generalize to the non-uniform version of the MMC problem, where each client has a possibly different demand. We obtain an $O(1)$ approximation for this problem as well. The approach tracks that of the uniform case very closely, but needs one additional idea to overcome a technical complication. We describe the generalization to the non-uniform MMC in Chapter 7.

1.5 The t-MMC Problem

To our knowledge, the t-MMC problem, where we are given a bound t on the number of servers that can be opened, has not been studied in its generality. The special case of 1-covering ($k = 1$) has, however, received considerable attention. Here, one wants to find t server balls to cover the clients, and minimize the sum of (the α-th powers of) the radii of the balls; this may be compared to the t-center problem, where one wants instead to minimize the maximum radius. For this special case of

t-MMC, Charikar and Panigrahy [23] address the metric setting and give an $O(1)$ approximation. Although they explicitly address only the case $\alpha = 1$, their guarantee generalizes to any $\alpha \geq 1$. Exploiting the special structure for the case $\alpha = 1$, Gibson et al. [38] give a polynomial time algorithm for solving the problem exactly in $\mathbb{R}^d$ if the underlying metric is ℓ_∞ or ℓ_1; for the ℓ_2 metric they obtain a polynomial time approximation scheme.

1.5.1 Our Contribution

For the t-MMC, which is addressed in Chapter 8, the computation of disjoint server subsets is identical to that of the MMC. However, the reduction to 1-covering is subtler as we have to worry about how many open servers are allowed for each 1-covering instance. One tool we develop to address this issue is the extraction of k outer covers from the optimal solution with additional guarantees on the number of servers opened in each outer cover. It is worth pointing out that in the case of 1-covering in the context of t-MMC, the only known approximation is the somewhat involved algorithm of Charikar and Panigrahy [23]. Thus, it is especially fortuitous that we are able to deal with the t-MMC by reducing to the 1-covering case, for which we can use their algorithm as a black box.

CHAPTER 2
A LOCAL SEARCH ALGORITHM FOR THE MCC PROBLEM

2.1 A Directed Local Search Algorithm

The input consists of a set of clients X and a set of servers Y in an arbitrary metric space $(X \cup Y, d)$, along with a constant $\alpha \geq 1$. Let $\delta(y, r)$ denote the ball of radius r centered on the server y, i.e. $\delta(y, r) = \{u \in X \cup Y \mid d(y, u) \leq r\}$. The objective is to cover the clients in X at least once using balls centered on the servers in Y, such that the sum of the α-th powers of the radii of the balls is minimized. Without loss of generality, we assume that the distances in the metric space are scaled such that the minimum inter-point distance is 1. We define the *level* of any ball $\delta(y, r)$, $L(\delta(y, r))$, as the minimum positive integer i such that $r \leq 16^i$. If the level of a ball $\delta(y, r)$ is i, then we refer to the ball $\delta(y, 16^i)$ as the *levelled* ball corresponding to the ball $\delta(y, r)$.

We now describe the pre-processing step. We first observe that any feasible solution for an instance of the MCC problem consists of balls drawn from the set $\mathcal{B}$ of at most $|X| \times |Y|$ distinct balls. For each ball $\delta(y, r) \in \mathcal{B}$, we add its corresponding levelled ball to $\bar{\mathcal{B}}$. Let $L_{\max} = \max\limits_{\delta \in \mathcal{B}} L(\delta)$ i.e. the maximum level of any ball in $\mathcal{B}$. We refer to all balls in $\bar{\mathcal{B}}$ having the same *level* i by $\bar{\mathcal{B}}(i)$. Obviously, $\bigcup\limits_{i=0}^{L_{\max}} \bar{\mathcal{B}}(i) = \bar{\mathcal{B}}$. The balls in $\bar{\mathcal{B}}$ are input to Algorithm 2.1.

Before describing the algorithm for computing a feasible cover that approximates any optimal cover, we introduce needed concepts. We first define a r-net as follows. For a positive integer r, an r-net of a graph $G = (V, E)$ is a set $S \subseteq V$ such that every path in G between any two vertices in S has at least r edges in it, and for every $u \in V \setminus S$, there exists a vertex $v \in S$ such that u is reachable from v using a path in G having at most $r - 1$ edges. An r-net is a fairly well-known concept; for instance, a 2-net is simply an independent set that is maximal by inclusion.

Algorithm 2.1 Directed LS

```
Require: The set of balls $\bar{\mathcal{B}}$.
 1: $C \leftarrow$ The levelled balls corresponding to the primary cover $C_{pr}$.
 2: for i = 0 to $L_{\max}$ do
 3:     $N_i \leftarrow \varnothing$.
 4:     $\bar{C}(i) \leftarrow \varnothing$.
 5:     for all $\bar{\delta} \in \bar{\mathcal{B}}(i)$ do
 6:         $\bar{C} \leftarrow \{\delta \in C \mid L(\delta) \le L(\bar{\delta}), \delta \cap \bar{\delta} \neq \varnothing\}$.
 7:         if $11^{\alpha} \cdot \texttt{cost}(\bar{\delta}) < \texttt{cost}(\bar{C})$ then
 8:             $N_i \leftarrow N_i \cup \bar{\delta}$.
 9:             Add all balls in $\bar{C}$ to the set $\bar{C}(i)$.
10:     Let $G_i = (N_i, E_i)$ be a graph whose vertices are the balls in $N_i$ and $(\delta, \delta') \in E_i$
        iff there is a ball $\bar{\delta} \in \bar{C}(i)$ such that $\delta \cap \bar{\delta} \neq \varnothing$ and $\delta' \cap \bar{\delta} \neq \varnothing$.
11:     $R_i \leftarrow$ A maximal 3-net of $G_i$.
12:     $\bar{R}_i \leftarrow \{11 \cdot \bar{\delta} \mid \bar{\delta} \in R_i\}$.
13:     $C \leftarrow C \setminus \bar{C}(i) \cup \bar{R}_i$.
14: return $L \leftarrow$ The set of balls in $C$.
```

For each client $x \in X$, we denote the nearest server to x in the set Y as its *primary server*. For every server $y \in Y$, we construct a ball $\delta(y, d(y, x'))$, where x' is the farthest client in X for which y is a primary server. The set of all such balls constitute a feasible solution, termed as the *primary cover*. Note that in any feasible solution, each client is covered by a ball that is at least as far away as its primary server.

Before describing the directed local search approach, let us consider the following simpler algorithm. We start with the primary cover C. The algorithm attempts to improve the cover C by testing against all candidate balls (in $\mathcal{B}$) in non-decreasing order of size (and thus cost) as follows. Let δ_c be a candidate ball. If $\delta_c \in C$, no action is taken and the algorithm chooses the next candidate ball according to the aforementioned ordering. If $\delta_c \notin C$, we try to determine if the cover C can be improved (i.e. cost of C can be decreased) by the addition of δ_c to C. Let N_c be the set of smaller balls in C that intersect δ_c. If $3 \cdot \texttt{cost}(\delta_c) < \texttt{cost}(N_c)$, then the cover C is modified by adding δ_c (expanded by a factor of 3) to C and removing N_c from C. The expansion of δ_c is needed to retain feasibility of the cover C, so as to cover all clients in X that were previously covered by balls in N_c. The step is repeated for all candidate balls in non-decreasing order of size to get the final cover.

Unfortunately, it is possible to end up with a solution whose cost is unbounded with respect to an optimal solution if we adopt the above algorithm without any modifications. Algorithm 2.1 is a refinement of the above idea in order to ensure that the resultant solution has a constant factor approximation guarantee.

The algorithm starts off with a feasible cover C, obtained from the levelled balls in the primary cover C_{pr}. It then tries to improve the cost of C using the set of all possible balls $\bar{\mathcal{B}}$. The outer loop (lines 2 to 13) is used to iterate over the levels of the balls in $\mathcal{B}$. In lines 5 to 9, we identify cheap candidate balls in $\bar{\mathcal{B}}(i)$. After all balls in $\bar{\mathcal{B}}(i)$ have been considered, the set N_i consists of all candidate balls for which a cost improving swap exists. We choose a subset of balls N_i for inclusion in the cover C in Lines 10 to 13, as described in the following. We construct the graph G_i in which the vertices are the balls in N_i and there is an edge between two balls $\delta_1, \delta_2 \in N_i$ if there is a ball $\delta' \in \bar{C}(i)$ such that $\delta_1 \cap \delta' \neq \varnothing$ and $\delta_2 \cap \delta' \neq \varnothing$. The algorithm computes a maximal 3-net on the graph G_i, and the balls corresponding to the vertices of the 3-net are denoted by R_i. The improved solution is then created by swapping out the balls in $\bar{C}(i)$ and adding the expanded balls corresponding to R_i (denoted by $\bar{R}_i$) to C (lines 10 to 13). We note that the feasibility of the cover C is maintained after the swap, as formalised in the following observation.

Observation 2.1. After each iteration of the outer loop, the balls in C form a feasible cover for X.

Proof. Before the first iteration, C was initialized using the levelled primary cover, and thus was a feasible cover. We assume C is feasible after the $(i-1)$-th iteration, and consider its feasibility during iteration i. C is modified only in Line 13, when the balls in $\bar{C}(i)$ are replaced by the balls in the net R_i. We show that C remains a feasible cover after the swap in the following. Let $x \in X$ be an arbitrary client. If x was covered by any ball in $C \setminus \bar{C}(i)$ prior to the swap, then x remains covered after removing $\bar{C}(i)$ from C. Consider the case where x was covered by $\delta \in \bar{C}(i)$ prior to the swap. Let $\bar{\delta} \in \bar{\mathcal{B}}(i)$ be the candidate disk such that $\bar{\delta} \in N_i$ and $\delta \cap \bar{\delta} \neq \varnothing$. If $\bar{\delta} \in R_i$, then x remains covered by the ball $11 \cdot \bar{\delta}$ that is added to C during the swap, since $L(\bar{\delta}) \geq L(\delta)$ by definition of $\bar{C}(i)$. Finally, we consider the case where $\bar{\delta} \in N_i \setminus R_i$. Since R_i is a maximal 3-net constructed from the G_i (the intersection graph of N_i), and $\bar{\delta} \notin R_i$, there exists some ball $\delta' \in R_i$ that is two hops away from $\bar{\delta}$ in G_i. By definition of G_i, any pair of adjacent disks in G_i are of level i and intersect a common ball in C whose level is at most i. Hence, the ball $11 \cdot \delta'$ contains the ball $\bar{\delta}$, ensuring that x is covered after the swap.

After considering each candidate ball in $\bar{\mathcal{B}}$, the algorithm returns the resultant cover C. Note that in this approach, each ball gets tested exactly once for inclusion in the final cover. This is unlike traditional local search algorithms - in which each ball may be considered multiple times in arbitrary order before the algorithm's termination.

2.2 Approximation Guarantee

In this section, we shall see how the set of balls returned by Algorithm 2.1 is a constant factor approximation to the MCC problem. We introduce the concept of a *directed local-search forest* (DLF) which forms a key component of the analysis.

$\mathcal{F}$ comprises of a set of rooted trees $\{\tau_1, \tau_2, \ldots, \tau_j\}$, where each node of any tree corresponds to a ball centered on servers in Y. For any tree $\tau \in \mathcal{F}$, let $\rho(\tau)$ denote the ball corresponding to the root of the tree τ. The forest is constructed as follows. For each ball δ in the levelled primary cover C_{pr}, (added in line 1 in Algorithm 2.1), a tree τ is added to $\mathcal{F}$ with δ as $\rho(\tau)$ (with δ being the only node in τ). The forest is updated after each iteration of the outer loop in Algorithm 2.1 as follows. Consider any iteration i. We maintain the invariant that the roots of the forest $\mathcal{F}$ correspond to the balls in the current feasible solution C. Each ball $\bar{\delta}$ in $\bar{R}_i$ is added to $\mathcal{F}$ as the parent of the all balls in $\bar{C}(i)$ that intersect the ball $\delta \in R_i$ corresponding to the ball $\bar{\delta}$. Once each ball in $\bar{R}_i$ has been assigned as a parent to some existing tree in $\mathcal{F}$, we remove all trees $\tau \in \mathcal{F}$ for which $\rho(\tau)$ is removed from C *and* no parent from $\bar{R}_i$ has been assigned as a parent to $\rho(\tau)$ in $\mathcal{F}$. The forest $\mathcal{F}$ thus constructed has the property that after updation of $\mathcal{F}$ at the end of each iteration of the outer loop, the set of balls corresponding to the roots of $\mathcal{F}$ constitute the current feasible solution C. It follows that when the algorithm terminates, the solution returned corresponds to the roots of the trees in $\mathcal{F}$.

We define some terminologies that will be used subsequently. Consider a tree $\tau \in \mathcal{F}$. For any ball δ that is a node in τ, let $\texttt{child}(\delta)$ denote the set of balls for which δ is a parent in τ. For a set of balls D, let $\texttt{child}(D) = \{\texttt{child}(\delta) \mid \delta \in D\}$ denote the children of D in $\mathcal{F}$. Consider $\tau \in \mathcal{F}$, and let $\Pi(\tau) = \{\pi_1, \pi_2, \ldots\}$ denote the set of all root to leaf paths in τ, where π_i consists of the set of balls in the i-th root to leaf path in τ. By a slight abuse of notation, we use $\Pi(\mathcal{F})$ to denote the set of all root to leaf paths for all $\tau \in \mathcal{F}$.

Let OPT denote the set of balls that form any optimal solution to the given instance of the MCC problem, and let OPT' be the levelled balls corresponding to OPT. Obviously, $OPT' \subseteq \bar{\mathcal{B}}$.

We are now ready to state the main lemma that is used in the analysis.

Lemma 1. Let L be the set of balls returned by Algorithm 2.1, and let OPT' be any levelled optimal solution for the MCC instance (i.e. the radius of each ball is set to its nearest power of 16). Let $\mathcal{F}$ be the directed search forest constructed as described before, and let $\Pi(\mathcal{F})$ be the set of all root to leaf paths in $\mathcal{F}$. Then, for each ball δ_o in OPT', it is possible to find a set $\alpha(\delta_o)$ of balls in $\mathcal{F}$ such that the following properties are satisfied:

- $\mathcal{A} = \bigcup\limits_{\delta_o \in OPT'} \alpha(\delta_o)$ is a hitting set for $\Pi(\mathcal{F})$ i.e. for each path $\pi \in \Pi(\mathcal{F})$, there is at least one ball in π that also belongs to $\mathcal{A}$.
- $\texttt{cost}(\mathcal{A}) \leq O(1) \cdot \texttt{cost}(OPT')$.

Proof. We describe the computation of the set of balls that constitute the hitting set that satisfies the properties in Lemma 1. Let δ_o be any ball in the levelled optimal solution OPT'. Since $OPT' \subseteq \bar{\mathcal{B}}$, the algorithm would have considered δ_o for inclusion in the solution at iteration $i = L(\delta_o)$. We define $\alpha(\delta_o)$ as follows:

1. If δ_o was not added to N_i (in lines 5 to 9), then for the set of balls in $\bar{C}$ corresponding to δ_o, we have $11^\alpha \cdot \texttt{cost}(\delta_o) \geq \texttt{cost}(\bar{C})$ holds and $\alpha(\delta_o) \leftarrow \bar{C}$.
2. Consider the case when δ_o was added to N_i but not subsequently to R_i. Let $\bar{C}$ be the set of balls in C of level at most i that intersect δ_o. Let $\bar{C}_{\mathcal{F}} \subseteq \bar{C}$ be the set of balls in $\bar{C}$ that are nodes in $\mathcal{F}$ i.e. each ball in $\bar{C}_{\mathcal{F}}$ lies in a root to leaf path in $\Pi(\mathcal{F})$. Let $\bar{\Pi}$ be the set of paths in $\Pi(\mathcal{F})$ which have at least one node in $\bar{C}_{\mathcal{F}}$. If $\bar{\Pi} = \varnothing$, then set $\alpha(\delta_o) \leftarrow \varnothing$. If $\bar{\Pi} \neq \varnothing$, then we claim there exists a single ball δ^R such that $\bar{C}_{\mathcal{F}} \subseteq \texttt{child}(\delta^R)$. Since δ_o was added to N_i, causing the balls in $\bar{C}_{\mathcal{F}}$ to be removed from C, none of the balls in $\bar{C}_{\mathcal{F}}$ are in the final solution L and hence cannot be the root in any root to leaf path in $\Pi(\mathcal{F})$. Hence, there must be one or more parent nodes in $\Pi(\mathcal{F})$ that contain balls in $\bar{C}_{\mathcal{F}}$. Let δ^R and δ^P be two disks in $R_i \cap \Pi(DLF)$ such that $\bar{C}_{\mathcal{F}}$ is contained within the expanded

disks $11 \cdot \delta^R$ and $11 \cdot \delta^P$ i.e $\bar{C}_{\mathcal{F}} \subseteq \texttt{child}(11 \cdot \delta^R)$ and $\bar{C}_{\mathcal{F}} \subseteq \texttt{child}(11 \cdot \delta^P)$. As R_i is a maximal 3-net constructed from the intersection graph G_i, the hop distance between δ^R and δ^P must be at least 3. But this contradicts the fact that δ_o intersects both δ^R and δ^P (as $\delta_o \cap \bar{C}_{\mathcal{F}} \neq \varnothing$), thus proving that $\bar{C}_{\mathcal{F}}$ can have exactly one parent node $11 \cdot \delta^R$. We set $\alpha(\delta_o) \leftarrow \{11 \cdot \delta^R\}$, and note that $\texttt{cost}(\alpha(\delta_o)) = 11^{\alpha} \cdot \texttt{cost}(\delta_o)$.

3. Finally, we consider the case when δ_o is in R_i. If there is a path π from δ_o to one of the roots of $\mathcal{F}$, then $\pi \in \Pi(\mathcal{F})$ and is hit by δ_o. In this case, we set $\alpha(\delta_o) \leftarrow \{11 \cdot \delta_o\}$. Otherwise, if δ_o does not lie on a path in $\Pi(\mathcal{F})$, then $\alpha(\delta_o) \leftarrow \varnothing$.

It remains to prove that $\mathcal{A}$ thus created is a hitting set for all paths in $\Pi(L)$. Consider any path $\pi \in \Pi(\mathcal{F})$, that terminates in a leaf node δ_p corresponding to some expanded primary ball. Any such primary ball must intersect at least one ball in an optimal solution OPT', δ_o, that is of the same or higher level as δ_p. We consider the iteration of the outer loop of Algorithm 2.1 for $i = L(\delta_o)$. We recall that by construction of the forest $\mathcal{F}$, any non-leaf node δ contains the disks corresponding to its child nodes in $\mathcal{F}$, and $\texttt{cost}(\delta) < \texttt{cost}(\texttt{child}(\delta))$. Let δ' be the first node on π whose level is at most i i.e. $L(\delta') \leq L(\delta_o) = i$. Since π is in the set of all root to leaf paths remaining after the algorithm terminates, then it must be that δ' is in C at the beginning of iteration i. Thus, δ' belongs to $\bar{C}$ corresponding to δ_o since δ' intersects δ_o. If δ_o was not added to N_i, then $\delta' \in \bar{C} = \alpha(\delta_o)$ and this δ' "hits" the path π. If δ_o was added to N_i but not to the net R_i, then the parent of δ' is added to $\alpha(\delta_o)$.

Finally, if δ_o was added to N_i and then subsequently to R_i, then $11 \cdot \delta_o$ is a parent of δ' in $\mathcal{F}$ and hits π. Thus, $\mathcal{A}$ is shown to be a hitting set for $\Pi(\mathcal{F})$, which completes the proof.

We can now show that Algorithm 2.1 returns a constant factor approximation i.e. $\texttt{cost}(L) \leq O(1) \cdot \texttt{cost}(OPT)$. Consider any ball $\delta_l \in L$, and let τ_l denote the tree in $\mathcal{F}$ such that $\rho(\tau_l) = \delta_l$. Let $\Pi(\tau_l) = \{\pi_1^l, \pi_2^l, \ldots\}$ denote the set of all root to leaf paths in τ_l. By Lemma 1, for each set of balls corresponding to any path $\pi \in \Pi(\tau_l)$, there is one ball in π that is also present in $\mathcal{A}$. By construction, each tree in $\mathcal{F}$ has the property that for any non-leaf node δ, the cost of the disk δ at that node is strictly less than the sum of the costs of immediate child nodes of δ i.e. $\texttt{cost}(\delta) < \texttt{cost}(\texttt{child}(\delta))$. Thus, we have the following relation using the last two statements:

$$\texttt{cost}(\delta_l) < \texttt{cost}(\tau_l \cap \mathcal{A}).$$

Summing up over all balls in L, we have by Lemma 1

$$\texttt{cost}(L) = \sum_{\delta_l \in L} \texttt{cost}(\delta_l) < \sum_{\delta_l \in L} \texttt{cost}(\tau_l \cap \mathcal{A}) \leq \texttt{cost}(\mathcal{A}) \leq O(1) \cdot \texttt{cost}(OPT).$$

We summarize our result in the following theorem.

Theorem 2. *Given point sets X, Y in a metric space $(X \cup Y, d)$ and a constant $\alpha \geq 1$, Algorithm 2.1 is a combinatorial algorithm that runs in polynomial time and returns a cover of X that is a constant factor approximation of any optimal cover of X.*

CHAPTER 3
OUTERCOVER

In this chapter, we look at a special variant of 1-cover (called an *OuterCover*), that is useful in deriving approximations for multi-covering problems.

Definition 4. Given point sets X, Y in a metric space $(X \cup Y, d)$, let $\kappa' : X \to \mathbb{Z}^+$ be a coverage function where we assume $\kappa'(x) \leq |Y|$ for any client x. For each $x \in X$, fix an ordering of the points in Y that is non-decreasing in terms of the distance d to x. We denote the $\kappa'(x)$-th nearest server of client x in this ordering by $\texttt{nn}(x, \kappa')$.

A κ'-outer cover is an assignment $\rho : Y \to \mathbb{R}^+$ of radii to the servers such that for each client $x \in X$ for which $\kappa'(x) > 0$, there is a server $y \in Y$ such that

1. The ball $\delta(y, \rho(y))$ contains x i.e. $d(y, x) \leq \rho(y)$.
2. Radius of the ball at y is large, that is, $\rho(y) \geq d(x, \texttt{nn}(x, \kappa'))$.

The objective function that we seek to minimize while computing an outer cover is the function $\sum_{y \in Y} (\rho(y))^\alpha$.

Given a level κ'-outer cover ρ, and a client $x \in X$ with $\kappa'(x) > 0$, any server y that satisfies the two conditions in the definition above is said to *serve* x; we also say that the corresponding ball $\delta(y, \rho(y))$ serves x. Observe that we do not require that a client x with $\kappa'(x) = 0$ be covered or served. When κ' is the constant function that takes on the value i for all $x \in X$, we denote a κ'-outer cover as an outer cover of level i. An illustration of an outer cover of level 2 is given in Figure 3.1.

We note that it is $\mathbb{NP}$-hard to compute an optimal κ'-outer-cover, even when

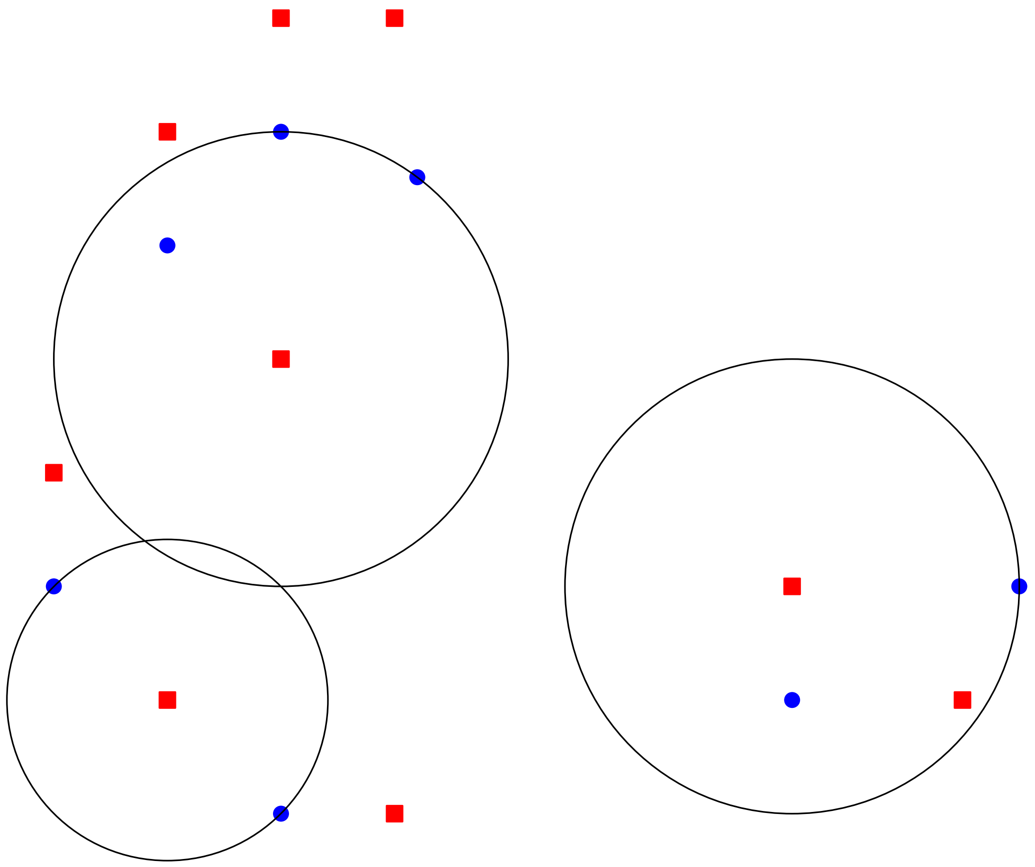

Figure 3.1: An example of OuterCover of level 2. The (blue) circles represent clients in X, the (red) squares represent servers in Y. The disk covering each client in X has a radius that is at least as large is its distance to its second nearest server in Y.

$\kappa'(x)$ is a constant for all $x \in X$. This follows from the $\mathbb{NP}$-hardness result in [9 authors], where it is shown that computing a level 1 outer-cover for points in the plane is $\mathbb{NP}$-hard.

We now look at the properties of a κ'-outer cover in the context of the MMC problem. In Section 3.1, we derive a lower bound for the cost of an optimal solution for the MMC problem in terms of optimal outercover costs. In the subsequent section (Section 3.2), we give a polynomial time algorithm that computes a constant factor approximation for a κ'-outer cover. Thus, combining the results in this chapter, we get the foundations for approximating the MMC problem.

3.1 The Outer Cover Lower Bound for the MMC Problem

In this section, we derive a lower bound on the cost of an optimal cover for the MMC problem with coverage function $\kappa : X \leftarrow \mathbb{Z}^+$. We first describe the intuition behind the existence of such a lower bound, by considering the case when $\kappa(x) = k$, for every $x \in X$. Let $\mu_i : X \leftarrow \mathbb{Z}^+$ denote the coverage function for an outer cover of level i i.e $\mu_i(x) = i$ for all $x \in X$. Let ρ_i denote the corresponding outer cover assignment, and let B_i be the set of balls created by the assignment ρ_i. If the set of balls B_i, B_j were to be disjoint for all $i \neq j$, then the collection of balls $\cup_{i=1}^{k} B_i$ would form a valid solution for the uniform MMC problem where each client has the coverage demand k - but its not immediately apparent how the *cost* of an optimal MMC solution can be bounded using individual outer cover balls. In the remainder of the section, we show that it is indeed possible to do so, by allowing a constant blowup in the total cost.

The following theorem is adapted from [17], which gives a lower bound on the cost of the optimal solution for the MMC problem.

Recall that for $1 \leq i \leq k$, the coverage function $\lambda_i : X \rightarrow \mathbb{Z}^+$ is defined by by $\lambda_i(x) = \max\{0, \kappa(x) - (i-1)\}$.

Theorem 3. *Let $\kappa : X \leftarrow \mathbb{Z}^+$ be the coverage function for the MMC problem on point sets X, Y on the metric space $(X \cup Y, d)$, and let $k = \max_{x \in X} \kappa(x)$ denote the maximum coverage demand. For $1 \leq i \leq k$, the coverage function $\lambda_i : X \rightarrow \mathbb{Z}^+$ is* *defined by by* $\lambda_i(x) = \max\{0, \kappa(x) - (i-1)\}$*.*

Let $r' : Y \to \mathbb{R}^+$ *be any assignment that constitutes a feasible solution to the MMC problem. For each* $1 \leq i \leq k$*, let* μ_{λ_i} *denote the cost of an optimal* λ_i*-outer cover. Then*

$$\sum_{i=1}^{k} \mu_{\lambda_i} \leq 3^\alpha \cdot \texttt{cost}(r').$$

Proof. Let $B = \{\delta(y, r'(y)) \mid y \in Y\}$ denote the set of balls corresponding to the assignment r'. We show that it is possible to form subsets $B_i \subseteq B$, for each $1 \leq i \leq k$ such that:

1. $\mu_{\lambda_i} \leq 3^\alpha \cdot \texttt{cost}(B_i)$.
2. $B_i \cap B_j = \varnothing$, for each $1 \leq i \neq j \leq k$.
3. No two balls in B_i intersect, for each $1 \leq i \leq k$.

If we show this, Theorem 3 follows because

$$\sum_{i=1}^{k} \mu_{\lambda_i} \leq 3^\alpha \cdot \sum_{i=1}^{k} \text{cost}(B_i) \leq 3^\alpha \cdot \text{cost}(B) = 3^\alpha \cdot \text{cost}(r').$$

We create the set of balls B_i in a top-down manner as described in Algorithm 3.1.

We thus have a set of balls $B_i, 1 \leq i \leq k$. It is clear that $B_i \cap B_j = \varnothing$ (Property 2), and no two balls in B_i intersect (Property 3).

We now verify that each B_i also satisfies Property 1. For this, consider L_i, the set of balls obtained by increasing the radius of each ball in B_i by a factor of 3. We argue that L_i is a λ_i-outer cover for X.

Fix $x \in X$, and consider the ball $\text{largest}_i(x)$ in Line 3 of iteration i. At this

Algorithm 3.1 Compute-Balls

```
Require: The set of balls $B$ corresponding to a $\kappa$-cover assignment $r'$
Ensure: The set of balls $B_i, 1 \leq i \leq k$.
 1: for $i = 1$ to $k$ do
 2:    Let $\text{largest}_i(x) \leftarrow$ The largest ball in $B$ that contains $x$.
 3:    Let $B_i{}' = \{\text{largest}_i(x) \mid x \in X\}$.
 4:    $B_i \leftarrow \varnothing$.
 5:    while $B_i{}' \neq \varnothing$ do
 6:       Let $b$ be the largest ball in $B_i{}'$.
 7:       $N \leftarrow$ Set of balls in $B_i{}'$ that intersect $b$. {Note: $b \in N$.}
 8:       $B_i \leftarrow B_i \cup \{b\}$.
 9:       $B_i{}' \leftarrow B_i{}' \setminus N$.
10:    $B \leftarrow B \setminus B_i$.
```

point, the balls in $\bigcup_{j=1}^{i-1} B_j$ have been removed from the original B, which had at least $\kappa(x)$ balls containing x. Since no two balls in B_j intersect, there is at most one ball in each B_j that contains x. Thus, at this point, there are at least $\kappa(x) - (i-1) = \lambda_i(x)$ balls left in B that contain x. Thus, the radius of $\text{largest}_i(x)$ is at least $d(x, \texttt{nn}(x, \lambda_i))$.

1. If $\text{largest}_i(x) \in B_i$, then the corresponding ball in L_i has radius at least $d(x, \texttt{nn}(x, \lambda_i))$.
2. If $\text{largest}_i(x) \notin B_i$, then there is an even larger ball b in B_i that intersects $\text{largest}_i(x)$. The ball obtained by multiplying the radius of b by 3 is in L_i; it contains $\text{largest}_i(x)$ and thus x; and it has radius at least $d(x, \texttt{nn}(x, \lambda_i))$.

Thus, L_i is a λ_i-outer cover for X. We infer that

$$\mu_{\lambda_i} \leq \text{cost}(L_i) \leq 3^\alpha \cdot \text{cost}(B_i).$$

Thus, Property 1 holds.

3.2 OuterCover: Algorithm to Generate a Preliminary Cover

Our goal in this section is to compute an outer cover that minimizes the cost $\sum_{y \in Y} (\rho(y))^\alpha$. In the rest of this section, we describe and analyze a procedure OuterCover(X', Y, κ, α) that returns an outer cover $\rho : Y \to \mathbb{R}^+$ whose cost is $O(1)$ times that of an optimal outer cover.

The procedure OuterCover(X', Y, κ, α) is implemented via a modification of the primal-dual algorithm of Charikar and Panigrahy [23]. Note that their algorithm can be viewed as solving the case where $\kappa(x) = 1$ for each $x \in X'$. As we will see, their algorithm and analysis readily generalize to the problem of computing an outer cover.

3.2.1 Linear Programming Formulation

We begin by formulating the problem of finding an optimal outer cover as an integer program. For each server $y_i \in Y$ and radius $r \geq 0$, let $z_i^{(r)}$ be an indicator variable that denotes whether the ball $\delta(y_i, r)$ is chosen in the outer cover.[1] For any server $y_i \in Y$ and client $x_j \in X'$, we define the *minimum eligible radius* $R_{\min}(y_i, x_j)$ to be:

$$R_{\min}(y_i, x_j) = \max(d(y_i - x_j), d(y^\kappa(x_j) - x_j))$$

A ball centered at y_i serves x_j in an outer cover exactly when its radius is at least $R_{\min}(y_i, x_j)$. Finally, let $C_i(r) = \{x_j \in X' \mid r \geq R_{\min}(y_i, x_j)\}$. The set $C_i(r)$

[1]For a server $y_i \in Y$, only the balls whose radius is from the set $\{d(y_i - x_j) \mid x_j \in X'\}$ will play a role in much of our algorithm. For describing the algorithm, however, it will be convenient to allow any $r \geq 0$.

consists of those clients that $\delta(y_i, r)$ can serve.

The problem of computing an optimal outer cover is that of minimizing

$$\sum_{i,r} r^\alpha \cdot z_i^{(r)}, \tag{3.1}$$

subject to the constraints

$$\sum_{i,r:x_j \in C_i(r)} z_i^{(r)} \geq 1, \ \forall x_j \in X' \tag{3.2}$$

$$z_i^{(r)} \in \{0, 1\}, \ \forall i, r. \tag{3.3}$$

The first constraint, equation (3.2), represents the condition that for every client $x_j \in X'$, at least one ball that is capable of serving it is chosen. The second constraint, equation (3.3), models the fact that the indicator variables $z_i^{(r)}$ can only take boolean values $\{0, 1\}$. By relaxing the indicator variables to be simply non-negative, i.e.

$$z_i^{(r)} \geq 0, \ \forall i, r, \tag{3.4}$$

we get a linear program (LP), which we call the primal LP for the problem.

The dual of the above LP has a variable β_j corresponding to every client $x_j \in X'$. The dual LP seeks to maximize

$$\sum_{x_j \in X'} \beta_j, \tag{3.5}$$

subject to the constraints

$$\sum_{x_j \in C_i(r)} \beta_j \leq r^\alpha, \ \forall y_i, r \tag{3.6}$$

$$\beta_j \geq 0, \ \forall x_j \in X' \tag{3.7}$$

3.2.2 A Primal Dual Algorithm

The primal dual algorithm is motivated by the above linear program. The algorithm maintains a dual variable β_j for each client x_j. This variable will always be non-negative and satisfy the dual constraints (3.6). If at some point in the algorithm, the dual constraint (3.6) holds with equality for some y_i and r, the ball $\delta(y_i, r)$ is said to be *tight*. A client x_j is said to be tight if there is some tight ball $\delta(y_i, r)$ such that $x_j \in C_i(r)$. (Note that β_j is then part of the dual constraint (3.6) that holds with equality.)

Our algorithm, OuterCover(X', Y, κ, α), initializes each β_j to 0, which clearly satisfies (3.6). The goal of the while loop in lines 1 and 2, which we refer to as the *covering phase* of the algorithm, is to ensure that each client in X' becomes tight, that is, covered by some tight ball. It is easy to see that the covering phase achieves this. We note in passing that since the β_j are never decreased in the covering phase, a client or ball that becomes tight at some point remains tight for the rest of the phase.

Steps 3–9 constitute the *coarsening phase* of the algorithm. This phase starts with the set T of tight balls computed by the covering phase. It computes a subset $F \subseteq T$ of pairwise disjoint balls by considering the balls in T in non-increasing order of radii, and adding a ball to F if it does not intersect any previously added ball.

Step 10 constitutes the *enlargement phase*. Each ball in F is expanded by a factor of 3, and the resulting set of balls is returned by the algorithm. Note that for $y_i \in Y$, F contains at most one ball centered at y_i; thus the assignment in Step 10 is

Algorithm 3.2 OuterCover(X', Y, κ, α)

```
 1: while $\exists\, x_j \in X'$ that is not tight do
 2:     Increase the non-tight variables $\beta_j$ arbitrarily till some constraint in (3.6) becomes tight.
 3: Let $T$ be the set of tight balls.
 4: $F \leftarrow \varnothing$
 5: while $T \neq \varnothing$ do
 6:     $\delta(y_i, r) \leftarrow$ The ball of largest radius in $T$
 7:     $N \leftarrow$ Set of balls that intersect $\delta(y_i, r)$
 8:     $F \leftarrow F \cup \{\delta(y_i, r)\}$
 9:     $T \leftarrow T \setminus N$
10: Assign $\rho : Y \to \mathbb{R}^+$ as follows:
```

$$\forall\ y_i \in Y, \rho(y_i) = \begin{cases} 3r, & \text{if } \delta(y_i, r) \in F \\ 0, & \text{if } F \text{ contains no ball centered at } y_i \end{cases}$$

well defined.

We argue that the balls returned by OuterCover(X', Y, κ, α) form an outer cover. Consider any client $x_j \in X'$. Since x_j is tight at the end of the covering phase, there is a tight ball $\delta(y_i, r)$ such that $x_j \in C_i(r)$. Thus x_j is served in case $\delta(y_i, r)$ was added to F in the coarsening phase. If $\delta(y_i, r)$ was not added to F, then it must have been intersected by some ball $\delta(y_{i'}, r')$ that was added to F, such that $r' \geq r$. Clearly, $x_j \in \delta(y_{i'}, 3r')$. Furthermore, $3r' \geq r \geq d(y^{\kappa}(x_j) - x_j)$ Thus, $x_j \in C_{i'}(3r')$, and x_j is served by the output of OuterCover(X', Y, κ, α).

3.2.3 Approximation Ratio

Let the set of balls in an optimal outer cover be denoted by OPT. We now show that the cost of the outer cover returned by OuterCover(X', Y, κ, α) is at most $3^{\alpha} \cdot \text{cost}(OPT)$. We begin by lower bounding $\text{cost}(OPT)$ in terms of the β_j. We have

$$\text{cost}(OPT) \geq \sum_{\delta(y_i,r)\in OPT} \left(\sum_{x_j \in C_i(r)} \beta_j \right) \geq \sum_{x_j \in X'} \beta_j. \tag{3.8}$$

The first inequality follows because the β_j satisfy (3.6); the second is because each client in X' is served by at least one ball in OPT, and the β_j are non-negative.

Let C denote the cost of the solution returned by OuterCover(X', Y, κ, α). We have

$$C = 3^\alpha \cdot \text{cost}(F) = 3^\alpha \sum_{\delta(y_i,r)\in F} \left(\sum_{x_j \in C_i(r)} \beta_j \right) \leq 3^\alpha \sum_{x_j \in X'} \beta_j \leq 3^\alpha \cdot \text{cost}(OPT).$$

Here, the second equality is because each ball in F is tight; since the balls in F are pairwise disjoint, each client $x_j \in X'$ is contained in at most one ball in F, from which the next inequality follows; the final inequality is due to Inequality (3.8).

Thus, we may conclude:

Theorem 4. *The algorithm OuterCover(X', Y, κ, α) runs in polynomial time and returns an outer cover whose cost is at most 3^α times that of an optimal outer cover.*

CHAPTER 4
MULTI-COVERING IN THE PLANE

In this chapter, we consider the version of the MMC problem in which the input point sets are on the Euclidean plane. We formally describe the problem statement as follows.

Definition 5. Given point sets X and Y in the plane, a coverage function $\kappa : X \to \{0, 1, 2, \ldots, |Y|\}$, and $\alpha \geq 1$, the objective is to find an assignment of radii $r : Y \to \mathbb{R}^+$ such that

- The disks corresponding to the assignment r forms a $\kappa(x)$ covering for all clients $x \in X$,
- The cost of assignment, $\sum\limits_{y \in Y} (r(y))^\alpha$ is minimized.

We refer to this version of the MMC problem as *Planar Multi-Covering* (PMC) problem.

In the remaining sections, we describe an algorithm that returns a feasible assignment for any input (X, Y, κ, α) such that the cost of the assignment is at most a constant factor times the cost of an optimal assignment. This algorithm has the additional property of being an *incremental* algorithm, explained as follows. For some input (X, Y, κ, α) to the PMC problem, let r be the assignment returned by the algorithm, whose cost is a constant times the optimal κ-cover. Now, let the coverage demand of some clients in X be increased to get a new coverage function κ', where $\kappa'(x) \geq \kappa(x)$ for all $x \in X$. The *incremental* property of our algorithm is that to get a constant factor approximation of an optimal κ'-cover, it is sufficient to increase the radii of some of the disks in the assignment r corresponding to the erstwhile κ-cover. Thus, any increase in the coverage function can be handled swiftly without having to recompute the corresponding cover from scratch.

4.1 Preliminaries

For convenience, we solve the variant of the Planar Multi-Covering problem where we have l_∞ disks rather than l_2 disks. The input is two point sets Y and X in $\mathbb{R}^2$, a coverage function $\kappa : X \to \mathbb{N} \cup \{0\}$, and the constant $\alpha \geq 1$. (It will be useful to allow $\kappa(x)$ to be 0 for some $x \in X$.) Throughout the remainder of this chapter, we would use $k = \max_{x \in X} \kappa(x)$ to denote the *maximum coverage* requirement of any client in X. We assume that $k \leq |Y|$, for otherwise there is no feasible solution.

Given an assignment $r : Y \to \mathbb{R}^+$ for each $y \in Y$, we will say that a point $x \in X$ is *j-covered* if at least j disks cover it, that is,

$$|\{y \in Y \mid ||x - y|| \leq r(y)\}| \geq j.$$

We will sometimes say that x is κ-covered to mean that it is $\kappa(x)$-covered. Similarly, if we have a assignment of radii to each $y \in Y$ such that for a set of points $P \subseteq X$, every point $x \in P$ is covered by at least $\kappa(x)$ disks, we say that P is κ-covered.

We describe an algorithm for returning an assignment $r : Y \to \mathbb{R}^+$ for each $y \in Y$, with the guarantee that for each $x \in X$, there are at least $\kappa(x)$ points $y \in Y$ such that the l_∞ disk of radius $r(y)$ centered at y contains x. In other words the guarantee is that for each $x \in X$,

$$|\{y \in Y \mid ||x - y||_\infty \leq r(y)\}| \geq \kappa(x)$$

Our objective is to minimize $\sum_{y \in Y} r(y)^\alpha$. For this optimization problem, we will show that our algorithm outputs an $O(1)$ approximation. Clearly, this also gives an $O(1)$ approximation for the original problem, where distances are measured in the l_2 norm. We will use $|| \cdot ||$ to denote the l_∞ norm.

For each $x \in X$, fix an ordering of the points in Y that is non-decreasing in terms of l_∞ distance to x. For $1 \leq j \leq |Y|$, let $y^j(x)$ denote the j-th point in this ordering. In other words, $y^j(x)$ is the j-th closest point in Y to x. For brevity, we denote $y^{\kappa(x)}(x)$ by $\mathtt{nn}(x, \kappa)$.

Let $\delta(p, r)$ denote the l_∞ disk of radius r centered at p. The *cost* of a set of disks is defined to the sum of the α-th powers of the radii of the disks. The cost of an assignment of radii to the servers is defined to be the cost of the corresponding set of disks.

4.2 Computing a Covering for the PMC Problem

Given the coverage function κ, we define a family of $k+1$ coverage functions $\lambda_i(x) = \max\{0, \kappa(x) - (i-1)\}$ $\forall 0 \leq i \leq k$. (Recall that k is the maximum demand of any client in κ). We note that λ_1 is equivalent to the coverage function κ, and λ_{k+1} denotes the trivial cover in which no client needs to be covered. We would use this family of coverage functions to describe our recursive approach, as follows.

Algorithm 4.1 is an algorithm that takes as input $(X', Y, \kappa, i, \alpha)$ and returns an assignment of radius $r(y)$ to each server $y \in Y$ such that each client $x \in X$ is λ_i-covered. This algorithm is recursive, bottoming out at $i = k+1$ where the demand function is λ_{k+1} where $\lambda_{k+1}(x) = 0$ for each $x \in X$. The reader should bear in mind that the topmost call is $(X, Y, \kappa, 1, \alpha)$, and works calls itself recursively till it bottoms out and then works its way up the stack. We now discuss the steps of the algorithm during the i-th invocation of the recursive call i.e when the parameters are $(X, Y, \kappa, i, \alpha)$.

In the base case, the radius $r(y)$ is assigned to 0 for each $y \in Y$. Otherwise, we recursively call Cover$(X, Y, \kappa, i+1, \alpha)$ to compute an assignment that λ_{i+1}-covers each $x \in X$. We then compute $X' \subseteq X$, the set of points that are not λ_i-covered. We compute an outer cover $\rho_i : Y \to \mathbb{R}^+$ for X' using the procedure OuterCover$(X', Y, \lambda_i, \alpha)$ described in Section 3.2. For any client $x \in X'$, the outer cover has a disk $\delta(y, \rho_i(y))$ that serves it. That is, x is contained in $\delta(y, \rho_i(y))$ and $\rho_i(y) \geq ||x - \mathtt{nn}(x, \lambda_i)||$.

The goal of the while-loop is to increase some of the $r(y)$ to ensure that each $x \in X'$, which is currently λ_{i+1}-covered, is also λ_i-covered. To do this, we iterate via the while loop over each disk $\delta(\overline{y}, \rho_i(\overline{y}))$ returned by OuterCover$(X', Y, \kappa, i, \alpha)$. We add all points in X' that are served in the outer cover by $\delta(\overline{y}, \rho_i(\overline{y}))$ to a set XC$_{\overline{y}}$. That is, XC$_{\overline{y}}$ consists of all $x' \in X'$ that are contained in $\delta(\overline{y}, \rho_i(\overline{y}))$ and $\rho_i(\overline{y}) \geq ||x' - \mathtt{nn}(x, \lambda_i)||$. The set YC$_{\overline{y}}$ contains, for each $x \in$ XC$_{\overline{y}}$, the $\lambda_i(x)$ nearest neighbors of x in Y. For purposes of analysis, we add $\overline{y}$ to a set $\overline{Y}$ as well.

Algorithm 4.1 Cover$(X, Y, \kappa, i, \alpha)$

```
1: Define $\lambda_i(x) = \max\{0, \kappa(x) - (i-1)\}$.
2: if $\forall x \in X, \lambda_i(x) = 0$ then
3:     Assign $r(y) \leftarrow 0$ for each $y \in Y$, and return.
4: Recursively call Cover$(X, Y, \kappa, i+1, \alpha)$.
5: Let $X' \leftarrow \{x \in X \mid x$ is not $\lambda_i(x)$-covered $\}$
6: Call procedure OuterCover$(X', Y, \lambda_i, \alpha)$ to obtain an outer cover $\rho_i : Y \to \mathbb{R}^+$.
7: Let $Y' \leftarrow Y$.
8: Let $\overline{Y} \leftarrow \varnothing$.
9: while $X' \neq \varnothing$ do
10:    Choose $\overline{y} \in Y'$.
11:    $\overline{Y} \leftarrow \overline{Y} \cup \{\overline{y}\}$.
12:    Let $\mathrm{XC}_{\overline{y}} \leftarrow \varnothing$, $\mathrm{YC}_{\overline{y}} \leftarrow \varnothing$.
13:    for all $x' \in X'$ do
14:        if $x' \in \delta(\overline{y}, \rho_i(\overline{y}))$ and $\rho_i(\overline{y}) \geq ||x' - y^{\kappa}(x')||$ then
15:            $\mathrm{XC}_{\overline{y}} \leftarrow \mathrm{XC}_{\overline{y}} \cup \{x'\}$.
16:            $\mathrm{YC}_{\overline{y}} \leftarrow \mathrm{YC}_{\overline{y}} \cup \{\mathtt{nn}(x', 1), \mathtt{nn}(x', 2), \ldots, \mathtt{nn}(x', \kappa)\}$.
17:    Let $\mathrm{YC}'_{\overline{y}} \subseteq \mathrm{YC}_{\overline{y}}$ be a set of at most four points such that
           $$\bigcap_{y \in \mathrm{YC}'_{\overline{y}}} \delta(y, r(y)) = \bigcap_{y \in \mathrm{YC}_{\overline{y}}} \delta(y, r(y)).$$
18:    For each $y \in \mathrm{YC}'_{\overline{y}}$, increase $r(y)$ by the smallest amount that ensures $\mathrm{XC}_{\overline{y}} \subseteq$
       $\delta(y, r(y))$.
19:    Remove $\overline{y}$ from $Y'$ and remove from $X'$ any points $x$ that are $\kappa(x)$-covered.
```

Next, we identify a set $\mathrm{YC}'_{\overline{y}} \subseteq \mathrm{YC}_{\overline{y}}$ of at most 4 points such that

$$\bigcap_{y \in \mathrm{YC}'_{\overline{y}}} \delta(y, r(y)) = \bigcap_{y \in \mathrm{YC}_{\overline{y}}} \delta(y, r(y)).$$

Why does such a $\mathrm{YC}'_{\overline{y}}$ exist? If, on the one hand, the intersection of disks $\bigcap_{y \in \mathrm{YC}_{\overline{y}}} \delta(y, r(y))$ is empty, then Helly's Theorem tells us that there are three disks (or maybe even two) whose intersection is empty. On the other hand, if the intersection $\bigcap_{y \in \mathrm{YC}_{\overline{y}}} \delta(y, r(y))$ is non-empty, then it is a rectangle (as these are l_∞ disks) and therefore equal to the intersection of four of the disks.

We enlarge the radius $r(y)$ of each $y \in \mathrm{YC}'_{\overline{y}}$ by the minimum amount needed to ensure that $\mathrm{XC}_{\overline{y}} \subseteq \delta(y, r(y))$. We argue that after this each point in $\mathrm{XC}_{\overline{y}}$ is λ_i-covered. To see why, consider any $x' \in \mathrm{XC}_{\overline{y}}$. Notice that $|\mathrm{YC}_{\overline{y}}| \geq \lambda_i(x')$, since the $\lambda_i(x')$ nearest neighbors of x' are included in $\mathrm{YC}_{\overline{y}}$. Thus before the enlargement, x' does not belong to $\bigcap_{y \in \mathrm{YC}_{\overline{y}}} \delta(y, r(y))$. (Recall that no point in $\mathrm{XC}_{\overline{y}}$ was κ-covered.) Therefore, x' does not belong to $\bigcap_{y \in \mathrm{YC}'_{\overline{y}}} \delta(y, r(y))$. It follows that there is at least one $y \in \mathrm{YC}'_{\overline{y}}$ such that $\delta(y, r(y))$ did not contain x' before the enlargement. As a consequence of the enlargement, $\delta(y, r(y))$ does contain x'. Since x' was $\lambda_{i+1}(x)$-covered before the enlargement, it is now $\kappa(x')$-covered.

After increasing $r(y)$ for $y \in \mathrm{YC}'_{\overline{y}}$ as stated, we discard from X' all points that are now λ_i-covered. The discarded set contains $\mathrm{XC}_{\overline{y}}$ and possibly some other points in X'. We remove $\overline{y}$ from Y'. We go back and iterate the while loop with the new X' and Y'.

Since any point in X' as computed in line 5 is served by some disk in the outer cover, it appears in $\mathrm{XC}_{\overline{y}}$ in some iteration of the while loop (if it has not already been λ_i-covered serendipitously). At the end of that iteration of the while loop, it gets λ_i-covered. Thus, when $\mathrm{Cover}(X, Y, \kappa, i, \alpha)$ terminates, each point $x \in X$ is $\lambda_i(x)$-covered.

4.3 Approximation Ratio

In this section, we bound the ratio of the cost of the solution returned by $\mathrm{Cover}(X, Y, \kappa, 1, \alpha)$ and the cost of the optimal solution.

For this purpose, the following lemma is central. It bounds the increase in cost incurred by Cover$(X, Y, \kappa, i, \alpha)$ in going from a λ_{i+1}-cover to a λ_i-cover by the cost of the outer cover ρ_i for X'.

Lemma 4.1. The increase in the objective function $\sum_{y\in Y} r(y)^\alpha$ from the time Cover$(X, Y, \kappa\ ,\ i+1, \alpha)$ completes to the time Cover$(X, Y, \kappa, i, \alpha)$ completes is at most $4 \cdot 3^\alpha \cdot \sum_{y\in Y} (\rho_i(y))^\alpha$, for all $1 \le i \le k$, where $k = \max_{x\in X} \kappa(x)$.

Proof. Let us fix an $\overline{y} \in \overline{Y}$, and focus on the iteration when $\overline{y}$ was added to $\overline{Y}$. Notice that there is exactly one such iteration, since $\overline{y}$ is removed from Y' in the iteration it gets added to $\overline{Y}$.

We will bound the increase in cost during this iteration. For this, we need two claims.

Claim 4.1. For any $x' \in \mathrm{XC}_{\overline{y}}$, we have

$$||\overline{y} - x'|| \le \rho_i(\overline{y})$$

Proof. Recall that x' is in $\mathrm{XC}_{\overline{y}}$ because $x' \in \delta(\overline{y}, \rho_i(\overline{y}))$.

Claim 4.2. For any $y' \in \mathrm{YC}_{\overline{y}}$, we have

$$||y' - \overline{y}|| \le 2 * \rho_i(\overline{y})$$

Proof. Let y' be added to $\mathrm{YC}_{\overline{y}}$ when $x' \in X'$ was added to $\mathrm{XC}_{\overline{y}}$. Hence

$$\begin{aligned} ||y' - x'|| &\le ||x' - y^\kappa(x')|| \\ &\le \rho_i(\overline{y}), \end{aligned}$$

since $\delta(\overline{y}, \rho_i(\overline{y}))$ serves x' in the outer cover (line 14 of Algorithm 4.1). Also, since $x' \in \delta(\overline{y}, \rho_i(\overline{y}))$, $||x' - \overline{y}|| \leq \rho_i(\overline{y})$. Therefore,

$$\begin{aligned} ||y' - \overline{y}|| &\leq ||y' - x'|| + ||x' - \overline{y}|| \\ &\leq \rho_i(\overline{y}) + \rho_i(\overline{y}) \\ &= 2\rho_i(\overline{y}) \end{aligned}$$

Fix a $y \in \mathrm{YC}'_{\overline{y}}$. If $r(y)$ was increased in this iteration, it now equals $||y - x'||$ for some $x' \in \mathrm{XC}_{\overline{y}}$. By the above two claims,

$$\begin{aligned} ||y - x'|| &\leq ||y - \overline{y}|| + ||\overline{y} - x'|| \\ &\leq 3 * \rho_i(\overline{y}) \end{aligned}$$

Thus the increase in $r(y)^\alpha$ is at most $3^\alpha(\rho_i(\overline{y}))^\alpha$. Since $r(y)$ is increased in this iteration only for $y \in \mathrm{YC}'_{\overline{y}}$, and $|\mathrm{YC}'_{\overline{y}}| \leq 4$, the increase in the objective function $\sum_{y \in Y}(r(y))^\alpha$ (in the iteration of the while loop under consideration) is at most $4 \cdot 3^\alpha \cdot (\rho_i(\overline{y}))^\alpha$.

We conclude that the increase in $\sum_{y \in Y}(r(y))^\alpha$ over all the iterations of the while loop is at most

$$4 \cdot 3^\alpha \cdot \sum_{\overline{y} \in \overline{Y}} (\rho_i(\overline{y}))^\alpha = 4 \cdot 3^\alpha \cdot \sum_{y \in Y} (\rho_i(y))^\alpha$$

We can now bound the approximation ratio of the algorithm.

Lemma 4.2. Let $r' : Y \to \mathbb{R}^+$ be any assignment of radii to the points in Y under which each point $x \in X$ is $\kappa(x)$-covered. We define γ_i as the *cost* of the assignment

r after the call Cover$(X, Y, \kappa, i, \alpha)$ has completed execution. Then

$$\gamma_1 \leq 4 \cdot 27^\alpha \cdot \sum_{y \in Y} (r'(y))^\alpha .$$

Proof. We first note that for a given coverage function κ, there would be exactly k recursive calls to the algorithm Cover$(X, Y, \kappa, i, \alpha)$, for $i = 1, 2, \cdots, k$, in that order. (Recall that k is the maximum coverage demand of any client). We look at the increase in the cost of the objective function in between successive recursive calls to Algorithm 4.1 in order to compute the final cost of the assignment r. In order to do so, we combine Theorem 4 from Chapter 3 and Lemma 4.1 to get the following relation (recall that μ_{λ_i} denotes the cost of an optimal λ_i-outercover):

$$\gamma_i - \gamma_{i+1} \leq 4 \cdot 3^\alpha \cdot \sum_{y \in Y} (\rho_i(y))^\alpha \leq 4 \cdot 9^\alpha \cdot \mu_{\lambda_i}. \tag{4.1}$$

Using the above equation, we get the following set of equations:

$$\begin{aligned} \gamma_1 - \gamma_2 &\leq 4 \cdot 9^\alpha \cdot \mu_{\lambda_1} \\ \gamma_2 - \gamma_3 &\leq 4 \cdot 9^\alpha \cdot \mu_{\lambda_2} \\ &\vdots \\ \gamma_k - \gamma_{k+1} &\leq 4 \cdot 9^\alpha \cdot \mu_{\lambda_k} \end{aligned} \tag{4.2}$$

Summing up and noting that $\gamma_{k+1} = 0$ as γ_{k+1} is the cost of the trivial cover where all clients have a demand of 0, we have:

$$\gamma_1 \leq 4 \cdot 9^\alpha \cdot \sum_{i=1}^{k} \mu_{\lambda_i} \leq 4 \cdot 27^\alpha \cdot \sum_{y \in Y} (r'(y))^\alpha \qquad \text{(Using Theorem 3)}$$

We conclude with a statement of the main result of this article. In this statement, cost refers to l_2 rather than l_∞ disks. Since (a) an l_2 disk of radius r is contained in the corresponding l_∞ disk of radius r, and (b) an l_∞ disk of radius r is contained in an l_2 disk of radius $\sqrt{2}r$, the approximation guarantee is increased by $(\sqrt{2})^\alpha$) when compared to Lemma 4.2.

Theorem 5. *Given point sets X and Y in the plane, a coverage function $\kappa : X \to \{0, 1, 2, \ldots, |Y|\}$, and $\alpha \geq 1$, the algorithm Cover$(X, Y, \kappa, 1, \alpha)$ runs in polynomial time and computes a κ-cover of X with cost at most $4 \cdot (27\sqrt{2})^\alpha$ times that of the optimal κ-cover.*

4.4 Concluding Remarks

Our result generalizes to the setting where X and Y are points in $\mathbb{R}^d$, where d is any constant. The approximation guarantee is now $(2d) \cdot (27\sqrt{d})^\alpha$. To explain, the intersection of a finite family of l_∞ balls equals the intersection of a sub-family of at most $2d$ balls. That is why the 4 in the approximation guarantee of Theorem 5 becomes $2d$. In the transition from l_2 to l_∞ balls in $\mathbb{R}^d$, we lose a factor of $(\sqrt{d})^\alpha$.

This generalization naturally leads to the next question – what can we say when X and Y are points in an arbitrary metric space? Our approach confronts a significant conceptual obstacle here, since one can easily construct examples in which the cost of going from a $(k-1)$-cover to a k-cover (for the uniform MCMC) cannot be bounded by a constant times the cost of an optimal outer cover. Thus, new ideas seem to be needed for obtaining an $O(1)$ approximation for this problem. The subsequent chapters give an affirmative answer to this question.

CHAPTER 5
PARTITIONING SERVERS IN METRIC SPACE

In this chapter, we specify the partitioning scheme that lies at the core of deriving constant factor approximations for multi-covering in the metric space. The partitioning scheme outputs *pairwise disjoint* subsets $Y_1, Y_2, \ldots, Y_k$ of servers such that computing an optimal 1-cover of the clients X using each Y_i and combining the 1-covers results in a solution whose cost is $O(1)$ of the optimal cost. This idea of reducing a multi-covering problem to several 1-covering problems works even when the coverage demands are non-uniform, albeit with some necessary pre-processing (Chapter 7). This scheme works even when the optimal 1-covers can use at most a constant number of servers from Y_i (as used in Chapter 8).

5.1 Overview

We now explain some key ideas of our partitioning scheme. For a client x, and any $1 \leq i \leq |Y|$, let us define the i-neighborhood of x, $N_i(x)$, to be the set consisting of the i nearest servers of x. At the core of our reduction is an analysis of the neighborhoods of the clients that may be of independent interest. In order to motivate this analysis, we first need to explain our high level plan for the server subsets $Y_1, Y_2, \ldots, Y_k$ in the MMC. As observed in Chapter 3, the optimal MMC solution can be viewed, up to a constant factor approximation, as a sequence $\rho_1, \rho_2, \ldots, \rho_k$, where each ρ_i is a cover of X. In particular, ρ_i is a special type of cover, called an outer cover of level i. This means that for each client x, there is a large ball in ρ_i that contains x – a ball whose radius is at least as large as the distance from x to its i-th nearest server.

Our plan for the server subsets $Y_1, Y_2, \ldots, Y_k$ is that for each $1 \leq i \leq k$, Y_i shall be "almost" a hitting set for ρ_i. If this can be achieved, then we can obtain a cover of X using just the servers in Y_i by moving each ball in ρ_i to a server in Y_i that hits it, and expanding the ball slightly. The cost of this cover is within a constant of that of ρ_i. Doing this for each $1 \leq i \leq k$, we get k covers of X whose total cost is within a constant of the optimal MMC solution. Furthermore, the fact that the subsets $Y_1, Y_2, \ldots, Y_k$ are pairwise disjoint implies that these k covers together form a valid MMC solution.

Thus, we would like each Y_i to be a hitting set for the corresponding outer cover ρ_i. Note, however, that we do not know anything about ρ_i, as it comes from the unknown MMC optimum. Therefore, we aim for an equivalent goal – we would like Y_i to be a hitting set for the i-neighborhoods of the clients. More concretely, we ask: can we extract k pairwise disjoint server subsets $Y_1, Y_2, \ldots, Y_k$ such that for each $1 \leq i \leq k$ and each client x, $N_i(x) \cap Y_i \neq \varnothing$?

This specification is too stringent, and the answer to this question is "no", as demonstrated by the example in Figure 5.1. Thus, we need a weaker specification for the Y_i that is still sufficient for our purposes. To describe it, we need one more notion. Let $G_i = (X, E_i)$ be the intersection graph of i-neighborhoods of X i.e. $(x_1, x_2) \in E_i$ iff $N_i(x_1) \cap N_i(x_2) \neq \varnothing$. What we are able to show is the following.

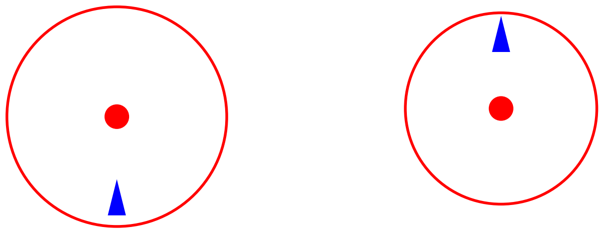

Figure 5.1: Let $k = 2$, and suppose there are two clients at distance 1 from each other, one server that is co-located with the first client, and a second server that is co-located with the second client. In this example, both servers would have to be in Y_1, leaving no server for Y_2.

Lemma 6. Assume k is even. We can efficiently compute a set

$$\left(\bigcup_{i=\frac{k}{2}+1}^{k} Y_i^s\right) \cup \left(\bigcup_{i=\frac{k}{2}+1}^{k} Y_i^p\right)$$

of k pairwise disjoint server subsets such that for each $\frac{k}{2}+1 \leq i \leq k$ and each client $x \in X$, there is a client x' within two hops of x in G_i such that $N_i(x') \cap Y_i^s \neq \varnothing$ (resp. $N_i(x') \cap Y_i^p \neq \varnothing$).

Note that we have weakened the original specification in two ways. First, instead of considering i-neighborhoods for each $1 \leq i \leq k$, we only consider i-neighborhoods for each $\frac{k}{2}+1 \leq i \leq k$, but now require two hitting sets for each such i. Second, for a fixed $\frac{k}{2}+1 \leq i \leq k$, we do not require that Y_i^s hits the i-neighborhood of *every* client in X. We only require that for any client x, there is some client x'

that is 'near' x such that Y_i^s intersects the i-neighborhood of x'. The requirement for Y_i^p is also relaxed in this way. The notion of 'near' is a natural one – that of being within a distance of 2 in the intersection graph G_i of the i-neighborhoods.

The proof of Lemma 6, which we prove subsequently in this chapter, is delicate. We construct the family $Y_k^s, Y_k^p, Y_{k-1}^s, Y_{k-1}^p, \ldots, Y_{\frac{k}{2}+1}^s, Y_{\frac{k}{2}+1}^p$ in that order, but we have to be careful while picking the earlier subsets to ensure that there are suitable servers left for building the later subsets. We give an algorithm for this construction in the next section.

5.2 Computing Disjoint Server Subsets

Suppose that we are given two point sets Y (servers) and X(clients) in an arbitrary metric space $(X \cup Y, d)$, and a positive integer k that represents the coverage demand of each client, and the constant $\alpha \geq 1$. We first re-iterate some notations and some needed tools from prior chapters for ease of reference.

Let $\delta(p, r)$ denote the ball of radius r centered at p, i.e., $\delta(p, r) = \{u \in X \cup Y \mid d(p, u) \leq r\}$. For brevity, we slightly abuse the notation and write $\delta(p, d(p, q))$ as $\delta(p, q)$. The *cost* of a set B of balls, denoted $\texttt{cost}(B)$, is defined to be the sum of the α-th powers of the radii of the balls.

Any assignment $r : Y \to \mathbb{R}^+$ corresponds to the set of balls $\{\delta(y, r(y)) \mid y \in Y\}$. Note that the cost of assignment r is the same as the cost of the corresponding set of balls. Instead of saying that r j-covers X, we will often say that the corresponding set of balls j-covers X. We will say that a set of balls *covers* X instead of saying it

1-covers X.

For each $x \in X$ and $1 \leq j \leq |Y|$, we define $y_j(x)$ to be the j-th closest point in Y to x using distance d. The ties are broken arbitrarily. For any $x \in X$, we define the i-neighborhood ball of a client x as $\delta(x, y_i(x))$. We define the i-neighborhood of x, $N_i(x)$, as $\{y_j(x) \mid 1 \leq j \leq i\}$.

For $1 \leq i \leq k$, let $G_i = (X, E_i)$ be the intersection graph of i-neighborhoods of X i.e. $(x, x') \in E_i$ iff $N_i(x) \cap N_i(x') \neq \varnothing$.

We now proceed to establish the following result, which is Lemma 6 restated so as to also address the case where k is odd.

Lemma 7. Let $l = \lceil k/2 \rceil$. We can efficiently compute a family $\mathcal{F}$ of k server subsets such that

1. $\mathcal{F}$ contains two subsets Y_i^s and Y_i^p for each $l+1 \leq i \leq k$, and, if k is odd, one additional subset Y_l^p.
2. $\mathcal{F}$ is a pairwise disjoint family, i.e., any two subsets in $\mathcal{F}$ are disjoint.
3. Suppose that (a) $l+1 \leq i \leq k$ and Y_i is either Y_i^s or Y_i^p, or (b) k is odd, $i = l$, and $Y_i = Y_i^p$. For any client $x \in X$, there is a client x' within two hops of x in G_i such that $N_i(x') \cap Y_i \neq \varnothing$.

Before describing the algorithm for computing the family $\mathcal{F}$, we introduce needed concepts. For a positive integer r, an r-net of a graph $G = (V, E)$ is a set $S \subseteq V$ such that every path in G between any two vertices in S has at least r edges in it, and for every $u \in V \setminus S$, there exists a vertex $v \in S$ such that u is reachable from v

using a path in G having at most $r-1$ edges. An r-net is a fairly well-known concept; for instance, a 2-net is simply an independent set that is maximal by inclusion.

We note that for any G_i, G_j such that $l \leq i < j \leq k$, G_i is a sub-graph of G_j since the i-neighborhood of any client is contained within its j-neighborhood. Motivated by the statement of Lemma 7, we would like to compute a 3-net X_i of G_i, for each $1 \leq i \leq k$. This would ensure that for any client $x \in X$, there is a client $x' \in X_i$ that is within two hops of x in G_i. For the rest of this section, we refer to a 3-net as simply a net.

Claim 5.1. There is a polynomial time algorithm that, given X, Y, and k, computes a hierarchy

$$X_k \subseteq X_{k-1} \subseteq \cdots \subseteq X_2 \subseteq X_1,$$

where each $X_i \subseteq X$ is a 3-net of G_i.

Proof. Given a net X_i of G_i, we describe how to compute a net X_{i-1} of G_{i-1} such that $X_i \subseteq X_{i-1}$. Since G_{i-1} is a subgraph of G_i, we have that the (hop) distance in G_{i-1} between any two vertices in X_i is at least 3. We initialize X_{i-1} with X_i and assume that all vertices in G_{i-1} are initially unmarked. We repeat the following process till G_{i-1} does not contain any unmarked vertices: mark all vertices in G_{i-1} within distance 2 of X_{i-1}, and then add an arbitrary unmarked vertex from G_{i-1} to X_{i-1}.

We can construct the hierarchy of nets by starting with an arbitrary net X_k of the graph G_k, and then constructing the successive nets in the hierarchy by the process described above. To construct X_k itself, we apply the above method after initializing X_k to be the singleton set consisting of any vertex in G_k.

Our algorithm for computing the family $\mathcal{F}$ of server subsets, as claimed in Lemma 7, is described in Section 5.2. We begin by setting parameter l to be $\lceil k/2 \rceil$, just as in the statement of Lemma 7. We then use Claim 5.1 to compute a hierarchy of nets, truncating it at l: $X_k \subseteq X_{k-1} \subseteq \cdots \subseteq X_l$. Any client that belongs to $\bigcup_{i=l}^{k} X_i$ is termed as a *net client*. For each client x, we denote the l-neighborhood $N_l(x)$ as the *private servers* of x.

Algorithm 5.1 ComputeServerSubsets(X, Y, k)

```
1: For each $y \in Y$, mark $y$ as available.
2: $l \leftarrow \lceil k/2 \rceil$
3: Compute $X_k \subseteq X_{k-1} \subseteq \cdots \subseteq X_l$ using Claim 5.1.
4: **for** $i = k$ **downto** $l$ **do**
5:    Let $Y_i^s \leftarrow \varnothing, Y_i^p \leftarrow \varnothing$.
6:    **for all** $x_c \in X_i$ **do**
7:       **if** $i > l$ **then**
8:          $y_s \leftarrow$ farthest available server in $N_i(x_c)$.
9:          $Y_i^s \leftarrow Y_i^s \cup \{y_s\}$. Mark $y_s$ as not available.
10:      **if** $i > l$ or ($i = l$ and $k$ is odd) **then**
11:         $y_p \leftarrow$ any available server in $N_l(x_c)$.
12:         $Y_i^p \leftarrow Y_i^p \cup \{y_p\}$. Mark $y_p$ as not available.
13: $\mathcal{F} \leftarrow \varnothing$.
14: **for** $i = k$ **downto** $l + 1$ **do**
15:    $\mathcal{F} \leftarrow \mathcal{F} \cup \{Y_i^s, Y_i^p\}$.
16: **if** $k$ is odd **then**
17:    $\mathcal{F} \leftarrow \mathcal{F} \cup \{Y_l^p\}$.
18: **return** The family $\mathcal{F}$
```

The disjoint server subsets are computed in Lines 4 to 12 of Section 5.2 – the for loop, whose index i goes down from k to l. In each iteration $i \geq l+1$, we extract two disjoint sets of servers Y_i^p and Y_i^s, and if k is odd, we extract one server set Y_l^p in iteration l. Notice that when summed over all i from k to l, we get k disjoint server sets. The algorithm then adds all these server subsets to $\mathcal{F}$ and returns it.

Observe that in iteration i of Line 4, we go through each client in $x_c \in X_i$, and use a carefully designed rule to pick two available servers from the i-neighborhood $N_i(x_c)$ of x_c to add to Y_i^p and Y_i^s. Observe that we add the farthest available server from the i-neighborhood $N_i(x_c)$ to Y_i^s, whereas we pick an available server from $N_l(x_c) \subseteq N_i(x_c)$, i.e., a private server of x_c, to add to Y_i^p. These choices – farthest and private – are crucial to our algorithm. The two added servers are immediately made unavailable. The fact that X_i is a net of G_i is useful in controlling the impact on server availability for later iterations of the algorithm. The subsequent section is devoted to establishing the crucial fact that such available servers can be found in iteration i.

Assuming that servers are available whenever the algorithm looks for them, we can now establish Lemma 7. Fix an i such that $l+1 \leq i \leq k$, and consider any client $x \in X$. Since X_i is a net of G_i, there is a client $x' \in X_i$ that is within two hops of x in G_i. From the inner loop (Line 6) in iteration i of the outer loop (Line 4), it is evident that for each $x_c \in X_i$, there is (at least) one server in Y_i^p (resp. Y_i^s) that belongs to the i-neighborhood $N_i(x_c)$. In particular, $N_i(x') \cap Y_i^p \neq \varnothing$, and $N_i(x') \cap Y_i^s \neq \varnothing$. If k is odd, a similar argument can be made for $i = l$ and Y_l^p. This establishes Lemma 7, assuming server availability.

5.3 Server Availability

Fix an iteration i of the for loop in Line 4 in Section 5.2. In such an iteration, the algorithm considers each $x_c \in X_i$ in the inner for loop in Line 6. For each x_c, it looks for up to two available servers within $N_i(x_c)$ and uses them. In order for the algorithm to be correct, such available servers must exist when the algorithm looks for them. In this section, which is the core of our analysis, we show that this is indeed the case.

Let us begin with a roadmap of this argument. Consider a client $x \in X_k$ that belongs to the net for G_k. Since the nets form a hierarchy, the client x also belongs to the net X_i for each $i < k$. Since the i-neighborhoods of clients in X_i are disjoint, for each i, the server choices made by other net clients do not affect x at all, and so x will be able to find available servers within $N_i(x)$ for each i. Now consider a server x' that first appears in the net X_j for some $j < k$. That is, x' is not in X_i for any $i > j$ but is in X_i for every $i \leq j$. What we argue is that at the beginning of iteration j of the for loop in Line 4, the j-neighborhood of x' is, from the perspective of available servers, similar to that of the j-neighborhood of x. It is in this argument that we use the fact that a private server is chosen in Line 11.

Properties of Nets. We now state some straightforward properties concerning the hierarchy of nets $X_k \subseteq X_{k-1} \subseteq \cdots \subseteq X_l$.

Claim 5.2. Let x, x' be two distinct clients in X_i. Then $N_i(x) \cap N_i(x') = \varnothing$.

Proof. Since X_i is a 3-net of G_i, any path between x and x' in G_i has at least three edges. Recall that the condition $N_i(x) \cap N_i(x') \neq \varnothing$ is equivalent to (x, x') being an edge in G_i.

Claim 5.3. Let $x \in X \setminus X_i$. Then there is at most one $x' \in X_i$ such that $N_i(x) \cap N_i(x') \neq \varnothing$.

Proof. If there are two clients x_1 and x_2 in X_i such that $N_i(x) \cap N_i(x_1) \neq \varnothing$ and $N_i(x) \cap N_i(x_2) \neq \varnothing$, then there is a path in G_i with at most two edges connecting x_1 and x_2. Since the clients x_1 and x_2 belong to X_i, this would contradict the fact that X_i is a 3-net.

Claim 5.4. Let $x_i \in X_i$ and $x_j \in X_j$ be any two distinct clients for $l \leq i < j \leq k$. Then, $N_i(x_i) \cap N_l(x_j) = \varnothing$.

Proof. Since $i < j$, we have $X_j \subseteq X_i$ and hence the clients x_i and x_j both belong to the net X_i, implying that $N_i(x_i) \cap N_i(x_j) = \varnothing$. Since $l \leq i$, the claim follows, as $N_l(x_j)$, the l-neighborhood of x_j, is contained in $N_i(x_j)$.

We now proceed to the actual argument for server availability, beginning with some notation. For $x \in X$, let $A_i(x)$ denote the set of available servers within $N_i(x) = \{y_j(x) \mid 1 \leq j \leq i\}$ at the *beginning* of iteration i. Thus, $|A_k(x)| = k$. Furthermore, $A_{i-1}(x) \subseteq A_i(x)$ for $l+1 \leq i \leq k$. Obviously, $A_i(x) \subseteq N_i(x)$.

The *threshold level* of a net client x (denoted by $\texttt{th}(x)$) is defined as:

$$\forall x \in \bigcup_{i=l}^{k} X_i, \qquad \texttt{th}(x) = \begin{cases} k, & \text{if } x \in X_k \\ \\ j, & \text{if } x \in X_j \setminus X_{j+1}, \quad l \leq j < k \end{cases}$$

The threshold level of x denotes the iteration of the outer loop of the algorithm in which client x first enters the net. In any iteration $k \geq j \geq \texttt{th}(x) + 1$, the client x can lose neighboring servers because of the server choices made by (the algorithm for) other clients, i.e., clients in the net X_j. On the other hand, for $l + 1 \leq j \leq \texttt{th}(x)$, x is itself part of the net X_j. In these iterations, it can only lose neighboring servers because of its own server choices. The next two claims address these two phases.

We now show that any net client x has enough available servers in its $\texttt{th}(x)$ neighborhood at the iteration $i = \texttt{th}(x)$ of the outer loop of Section 5.2.

Claim 5.5. Let x be any net client, and let $i = \texttt{th}(x)$. Then

(a) $|A_i(x) \cap N_l(x)| \geq l - (k - i)$.

(b) $|A_i(x)| \geq 2i - k = k - 2(k - i)$.

Proof. We look at the servers chosen during iteration j of the outer loop, for $i < j \leq k$. Note that x didn't belong to the net X_j. Consider any client $x_j \in X_j$. By Claim 5.4, $N_i(x)$ does not intersect the l-neighborhood ball $N_l(x_j)$. Hence, during the execution of Line 11 in the inner for loop corresponding to x_j, no server is made unavailable from $N_i(x)$. This is because the server chosen in Line 11 belongs to $N_l(x_j)$.

Thus, during iteration j, servers from $N_i(x)$ can become unavailable only during the execution of Line 8 of the inner for loop. We note that by Claim 5.3, there is at most one client $x_j \in X_j$ such that $N_j(x_j) \cap N_j(x) \neq \varnothing$. Thus, at most one server from $N_i(x)$ is made unavailable in iteration j.

We conclude that across the $k - i$ iterations before iteration i, there can be at most $k - i$ servers from $N_i(x)$ that have been made unavailable. Hence, $|A_i(x)| \geq$

$i - (k - i) = 2i - k$. Since $|N_i(x) \cap N_l(x)| = l$ and at most $k - i$ servers are made unavailable from the i-neighborhood ball $N_i(x)$, $|A_i(x) \cap N_l(x)| \geq l - (k - i)$.

For any net client x, Claim 5.5 shows that in iteration $i = \texttt{th}(x)$, when x first enters the net, there are enough available servers in $N_i(x)$. The following claim aids in asserting this for subsequent iterations, by arguing that in any iteration $i \leq \texttt{th}(x)$, at most 2 available servers are made unavailable from $N_i(x)$.

Claim 5.6. Let x be any net client and $l + 1 \leq i \leq \texttt{th}(x)$. Then

(a) $|A_{i-1}(x)| \geq |A_i(x)| - 2$

(b) If $|A_{i-1}(x)| = |A_i(x)| - 2$, then one of the servers in $A_i(x) \setminus A_{i-1}(x)$ is the farthest server in $A_i(x)$ from x.

Proof. Note that $x \in X_i$ since $i \leq \texttt{th}(x)$. Consider any $x_c \in X_i \setminus \{x\}$. In the iteration of the inner for loop (Line 6) corresponding to x_c, any servers that are made unavailable belong to $N_i(x_c)$ and are therefore not in $N_i(x)$, by Claim 5.2 (since $x, x_c \in X_i$). Thus, if any servers in $A_i(x) \subseteq N_i(x)$ become unavailable in iteration i, then this can happen only in the iteration of the inner for loop corresponding to x. In this iteration of the inner for loop, the servers that become unavailable are y_s, the farthest server from x in $A_i(x)$, and y_p, a different server that is chosen from the available servers in $N_l(x)$. Note that $\{y_p, y_s\} \subseteq A_i(x)$. Thus, only the two servers y_s, y_p in $A_i(x)$ become unavailable in iteration i. Furthermore, if $y_i(x) \in A_i(x)$ then $y_i(x)$ is the farthest server in $A_i(x)$ from x, and thus $y_i(x) = y_s$. Thus $A_i(x) \setminus A_{i-1}(x) = \{y_s, y_p\}$, and Claim 5.6 (a) holds. Since y_s is the farthest server in

$A_i(x)$, Claim 5.6 (b) holds as well.

The following two claims show that our algorithm always succeeds in finding available servers.

Claim 5.7. For any $l+1 \le i \le k$, and any $x_c \in X_i$:

(a) There is an available server in $N_i(x_c)$ when the algorithm executes Line 8 in the iteration of the inner for loop (Line 6) corresponding to x_c.

(b) There is an available server in $N_l(x_c)$ when the algorithm executes Line 11 in the iteration of the inner for loop (Line 6) corresponding to x_c.

Proof. Since $x_c \in X_i$, we infer that $i \le \mathtt{th}(x_c)$. Using Claim 5.6, we have

$$\begin{aligned}
|A_i(x_c)| &\ge |A_{i+1}(x_c)| - 2 \\
&\ge |A_{i+2}(x_c)| - 2 - 2 \\
&\ge \dots \\
&\ge |A_{\mathtt{th}(x_c)}(x_c)| - 2 \cdot (\mathtt{th}(x_c) - i) \\
&\ge k - 2(k-i) && (\because |A_{\mathtt{th}(x_c)}(x_c)| \ge k - 2(k - \mathtt{th}(x_c))) \\
&\ge 2 && (\because i \ge l+1)
\end{aligned}$$

Using an argument from the proof of Claim 5.6, none of the servers in $A_i(x_c)$ are made unavailable in iteration i till x_c is considered in Line 6. Thus, there are at least two servers available when the algorithm executes Line 8 corresponding to x_c, and Claim 5.7 (a) holds.

The argument for Claim 5.7 (b) is similar but requires some case analysis. We begin by observing that when the algorithm executes Line 11 corresponding to

x_c, there is at least one available server $y \in N_i(x_c)$. Now suppose that in some iteration $i+1 \leq j \leq \texttt{th}(x_c)$, $A_j(x_c) \setminus A_{j-1}(x_c)$ consists of two servers from $N_l(x_c)$. By Claim 5.6 (b), a server from $N_l(x_c)$ is the farthest server from x_c in $A_j(x_c)$. This implies that all servers in $A_{j-1}(x_c)$ belong to $N_l(x_c)$, and thus $y \in N_l(x_c)$. This y is available when the algorithm executes Line 11 corresponding to x_c.

We are left with the case that in each iteration $i+1 \leq j \leq \texttt{th}(x_c)$, $A_j(x_c) \setminus A_{j-1}(x_c)$ consists of at most one server from $N_l(x_c)$. Using Claim 5.5 (a), and the fact that $\texttt{th}(x_c) - i$ iterations have happened since iteration $\texttt{th}(x_c)$, we have

$$|A_i(x_c) \cap N_l(x_c)| \geq |A_{\texttt{th}(x_c)}(x_c) \cap N_l(x_c)| - (\texttt{th}(x_c) - i) \geq l - (k - i) \geq 1.$$

Thus there is at least one server $y' \in A_i(x_c) \cap N_l(x_c)$. If the server chosen in Line 8 corresponding to x_c belongs to $N_l(x_c)$, then all available servers in $N_i(x_c)$ belong to $N_l(x_c)$. Thus, once again, some server in $N_l(x_c)$ is available when the algorithm executes Line 11 corresponding to x_c. If the server chosen in Line 8 corresponding to x_c does not belong to $N_l(x_c)$, then $y' \in N_l(x_c)$ is available when the algorithm executes Line 11 corresponding to x_c. We have thus shown that Claim 5.7 (b) holds.

If k is even, the algorithm does not look for available servers in iteration $i = l$. If k is odd, the algorithm will look for available servers in iteration $i = l$, in Line 11. The following claim extends the previous one to handle this. The proof is a straightforward extension of the proof of the previous claim, and is therefore omitted.

Claim 5.8. Suppose k is odd. For iteration $i = l$, and any $x_c \in X_i$, there is an available server in $N_l(x_c)$ when the algorithm executes Line 11 in the iteration of the inner for loop (Line 6) corresponding to x_c.

This completes the proof of Lemma 7.

In the next 3 chapters, we will see the application of this partitioning scheme for finding constant factor approximations for multiple variants of the multi-covering problem.

CHAPTER 6
THE UNIFORM MMC PROBLEM

6.1 Algorithm

In this section, we present a constant factor approximation for the uniform MMC problem. Recall that our input consists of two point sets Y (servers) and X (clients) in an arbitrary metric space $(X \cup Y, d)$, a positive integer k that represents the coverage demand of each client, and the constant $\alpha \geq 1$. Our algorithm (Algorithm 6.1) first computes a family $\mathcal{F}$ consisting of k pairwise disjoint subsets of Y, using Section 5.2 of Lemma 7. It then invokes Cover(X, Y', α), for each $Y' \in \mathcal{F}$, to compute a near-optimal 1-cover of X using only the servers in Y'. Since there are k server subsets in $\mathcal{F}$, we obtain k 1-covers of X. The algorithm then returns r, the union of the k covers. Because server subsets in $\mathcal{F}$ are disjoint, this union yields a k-cover of X.

Algorithm 6.1 MetricMultiCover(X, Y, k, α)

```
1: For each $y \in Y$, assign $r(y) \leftarrow 0$.
2: $\mathcal{F} \leftarrow$ ComputeServerSubsets$(X, Y, k)$.
3: for all $Y' \in \mathcal{F}$ do
4:     $\bar{r} \leftarrow$ Cover$(X, Y', \alpha)$.
5:     Let $r(y') \leftarrow \bar{r}(y')$ for each $y' \in Y'$.
6: return The assignment $r : Y \rightarrow \mathbb{R}^+$.
```

In the remainder of this chapter, we outline the ideas necessary to establish the approximation ratio of the algorithm above, using properties of the partitioning scheme as well as the outer-cover lower bound established in Chapter 3 to achieve the bound.

6.2 Approximation Guarantee

Note that Algorithm 6.1 computes the family $\mathcal{F} = \{Y_k^s, Y_k^p, Y_{k-1}^s, Y_{k-1}^p, \ldots\}$ as detailed in Lemma 7. Let $Y_i \in \mathcal{F}$ be one such subset, where Y_i may be either Y_i^p or Y_i^s. Y_i has the property that for any $x \in X$, there is an $x' \in X$ that is within two hops of x in G_i such that $N_i(x') \cap Y_i \neq \varnothing$. The following claim uses this property to argue that there is an inexpensive 1-cover of X that only uses servers from Y_i. The 1-cover is constructed by using the servers in Y_i to "host" the balls in the outer cover ρ_i.

Claim 6.1. Assume that either (a) $l + 1 \leq i \leq k$ and Y_i is either Y_i^p or Y_i^s, or (b) k is odd, $i = l$, and $Y_i = Y_i^p$. Let ρ_i be any outer cover of level i for X using servers from Y. There is a 1-cover of X that uses servers from Y_i and has cost at most $12^\alpha \cdot \texttt{cost}(\rho_i)$.

Proof. Consider the set B of balls obtained by expanding each ball in the outer cover ρ_i to 6 times its original radius. We claim

Claim 6.2. For any client $x \in X$, there is some ball in B that contains x as well as at least one server in Y_i.

Before proving Claim 6.2, we first prove Claim 6.1 using it. We construct a set B' of balls as follows. Consider any ball $b \in B$. If it does not contain a server from Y_i, we ignore it. If it does contain a server in Y_i, pick an arbitrary such server

y, translate b so that it is centered at y, double its radius, and add the resulting ball to B'.

It is possible at this stage that for a server $y \in Y_i$, there are several balls in B' centered at y. From each such concentric family, discard from B' all but the largest of the concentric balls. It follows from Claim 6.2 that B' covers each client in X. Since each ball in B' is obtained by translating and scaling some ball in the outer cover ρ_i by a factor of 12, the cost of B' is at most $12^\alpha \cdot \texttt{cost}(\rho_i)$. This establishes Claim 6.1.

We now turn to the proof of Claim 6.2. From the definition of G_i, we have that for any edge (x', x'') in G_i,

$$d(x', x'') \leq d(x', y_i(x')) + d(x'', y_i(x'')). \tag{6.1}$$

Now consider an arbitrary client $x \in X$. By Lemma 7, there is a path π in G_i with at most 2 edges (and 3 vertices) that connects x to some vertex $\bar{x}$, with $N_i(\bar{x}) \cap Y_i \neq \varnothing$.

Let $\delta(y, \rho_i(y))$ be the largest ball in outer cover ρ_i that serves at least one vertex on path π. Suppose that it serves vertex $\hat{x} \in \pi$. ($\hat{x}$ could be the same as x or $\bar{x}$.) See Figure 6.1 for an illustration. Using the definition of an outer cover of level i, and the way we pick the ball $\delta(y, \rho_i(y))$, it follows that for any vertex $x' \in \pi$,

$$d(x', y_i(x')) \leq \rho_i(y). \tag{6.2}$$

Thus,

$$d(y, x) \leq d(y, \hat{x}) + \left(\sum_{(x', x'') \in \pi[\hat{x}, x]} d(x', x'') \right) \leq 5\rho_i(y).$$

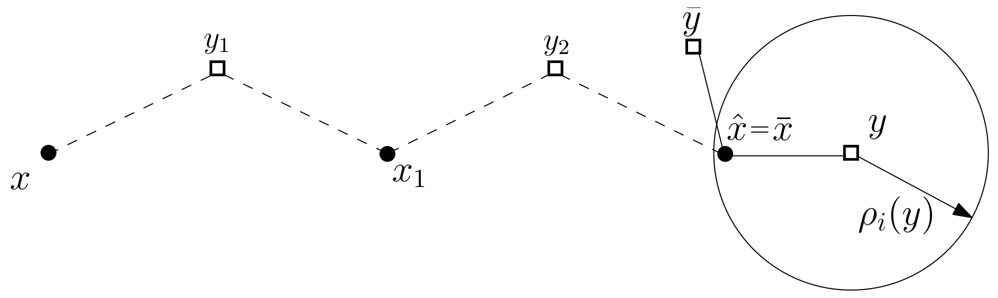

Figure 6.1: Illustration for the proof of Claim 6.2. For the client $x \in X$, the dashed edges correspond to a path π in G_i from x to $\bar{x}$. Here, $y_1 \in N_i(x) \cap N_i(x_1)$, $y_2 \in N_i(x_1) \cap N_i(\bar{x})$, and $\bar{y} \in Y_i \cap N_i(\bar{x})$. The ball $\delta(y, \rho_i(y))$ serves $\hat{x}$. Here, $\hat{x}$ happens to be $\bar{x}$. Note that we can get from y to x using 5 edges of the figure, and from y to $\bar{y}$ using 2 edges. Therefore, expanding the ball at y by a factor of 6 will cover both x and $\bar{y}$. (In this example, even a factor of 5 suffices.)

Here, we denote by $\pi[\hat{x}, x]$ the sub-path of π from $\hat{x}$ to x, and use Inequalities 6.1 and 6.2 in the second step.

Now, $N_i(\bar{x}) \cap Y_i \neq \varnothing$. Let $\bar{y} \in N_i(\bar{x}) \cap Y_i$ be chosen arbitrarily. Clearly, $d(\bar{x}, \bar{y}) \leq d(\bar{x}, y_i(\bar{x})) \leq \rho_i(y)$.

We calculate that

$$d(y, \bar{y}) \leq d(y, \hat{x}) + \left(\sum_{(x', x'') \in \pi[\hat{x}, \bar{x}]} d(x', x'') \right) + d(\bar{x}, \bar{y}) \leq 6\rho_i(y).$$

Thus, the ball $\delta(y, 6\rho_i(y))$ contains both x and $\bar{y} \in Y_i$, completing the proof of Claim 6.2.

Remark. With a more detailed argument, the factor 12^α can be improved. For instance, a bound of 11^α is almost immediate from the proof.

We can now establish the approximation guarantee for Algorithm 6.1 and our main result.

Theorem 8. *Given point sets X and Y in a metric space $(X \cup Y, d)$ and a positive integer $k \le |Y|$, Algorithm 6.1 runs in polynomial time and returns a k-cover of X with cost at most $2 \cdot (12 \cdot 9)^\alpha$ times that of an optimal k-cover.*

Proof. It is evident that the algorithm runs in polynomial time, and we have already noted that it returns a k-cover r. Let r' be any optimal assignment. By Theorem 3, there exist outer covers ρ_i, for $1 \le i \le k$, such that

$$\sum_{i=1}^{k} \texttt{cost}(\rho_i) \le 3^\alpha \cdot \texttt{cost}(r').$$

Assume that either (a) $l+1 \le i \le k$ and Y_i is either or Y_i^p or Y_i^s, or (b) k is odd, $i = l$, and $Y_i = Y_i^p$. From Claim 6.1, we conclude that there is a 1-cover for X that uses servers Y_i and has cost at most $12^\alpha \cdot \texttt{cost}(\rho_i)$. Since Cover$(X, Y_i, \alpha)$, which is invoked in Algorithm 6.1 returns a 3^α approximation, the cost of the 1-cover it returns is at most $(12 \cdot 3)^\alpha \texttt{cost}(\rho_i)$.

At most two 1-covers are computed for each i, once with server set Y_i^s and once with server set Y_i^p. Thus,

$$\begin{aligned}
\texttt{cost}(r) &\le 2 \cdot (12 \cdot 3)^\alpha \cdot \sum_{i=l}^{k} \texttt{cost}(\rho_i) \\
&\le 2 \cdot (12 \cdot 3)^\alpha \cdot \sum_{i=1}^{k} \texttt{cost}(\rho_i) \\
&\le 2 \cdot (12 \cdot 9)^\alpha \cdot \texttt{cost}(r').
\end{aligned}$$

CHAPTER 7
THE NON-UNIFORM MMC PROBLEM

In this chapter, we address the MMC problem as described in Definition 1, specifically the case where we allow each client to specify its own coverage requirement, and present an $O(1)$ approximation for it. Recall that the input consists of two point sets Y(servers) and X(clients) in an arbitrary metric space $(X \cup Y, d)$, a constant $\alpha \geq 1$, and a coverage function $\kappa : X \to \mathbb{Z}^+$.

Consider an assignment $r : Y \to \mathbb{R}^+$ of radii to each server in Y. This can be viewed as specifying a ball of radius $r(y)$ at each server $y \in Y$. If, for each client $x \in X$, at least $\kappa(x)$ of the corresponding server balls contain x, then we say that X is κ-covered by r. That is, X is κ-covered if for each $x \in X$,

$$|\{y \in Y \mid d(x, y) \leq r(y)\}| \geq \kappa(x).$$

Any assignment $r : Y \to \mathbb{R}^+$ that κ-covers X is a feasible solution to the MMC, and the goal is to find a feasible solution that minimizes the cost $\sum_{y \in Y} (r(y))^\alpha$. We assume that $\kappa(x) \leq |Y|$ for each client x, for otherwise there is no feasible solution.

To solve the non-uniform MMC problem, our plan is to partition the set of servers Y into disjoint sets and invoke a 1-covering algorithm with each server subset. Unlike the uniform case (Chapter 6), each 1-covering instance thus generated may only cover a subset of the clients, and not all clients in X. For example, a client x such that $\kappa(x) = 100$ will be involved in 100 1-covering instances, whereas a client x' with demand 50 would be in 50 1-covering instances.

7.1 Partitioning Servers

Our algorithm for partitioning Y into server subsets uses a criterion that generalizes that of Lemma 7. We adopt terminology for the non-uniform case from Chapter 5. Let k now denote $\max_{x \in X} \kappa(x)$. For client $x \in X$, let its set of *private servers* be $N_l(x) = N_{\lceil \kappa(x)/2 \rceil}(x)$. For notational convenience, we denote $N_i(x)$, the set of i nearest servers to x in Y, by $\mathtt{NN}(x, i)$.

Before stating the generalized lemma, we need some additional definitions. For $1 \leq i \leq k$, we define the coverage function $\lambda_i : X \to \mathbb{Z}^+$ by $\lambda_i(x) = \max\{0, \kappa(x) - (i-1)\}$. Thus, λ_i is obtained by decreasing the original coverage requirement of each client by $i - 1$, with the proviso that we don't decrease below 0. For each $1 \leq i \leq k$, we define an undirected graph G_{λ_i} with vertex set X. We add (x, x') as an edge in G_{λ_i} if (a) $i \leq \lceil \kappa(x)/2 \rceil$; (b) $i \leq \lceil \kappa(x')/2 \rceil$; and (c) $\mathtt{NN}(x, \kappa(x) - (i-1)) \cap \mathtt{NN}(x', \kappa(x') - (i-1)) \neq \varnothing$. Note that condition if (a) and (b) hold, condition (c) can also be written as $\mathtt{NN}(x, \lambda_i(x)) \cap \mathtt{NN}(x', \lambda_i(x')) \neq \varnothing$. The conditions (a) and (b) ensure that a client x is isolated in graph G_{λ_i} for $i > \lceil \kappa(x)/2 \rceil$.

Lemma 9. Let $l = \lceil k/2 \rceil$. We can efficiently compute a family $\mathcal{F}$ of server subsets such that

1. $\mathcal{F}$ contains two subsets $Y^s_{\lambda_i}$ and $Y^p_{\lambda_i}$ for each $1 \leq i < l$. For $i = l$, if k is even, $\mathcal{F}$ contains $Y^s_{\lambda_l}$ and $Y^p_{\lambda_l}$, else $\mathcal{F}$ contains only $Y^p_{\lambda_l}$.
2. $\mathcal{F}$ is pairwise disjoint.
3. Fix $1 \leq i \leq l$.
   (a) For any client x with $\kappa(x) \geq 2i - 1$, there is a client $x' \in X$ within 3 hops

of x in G_{λ_i} such that $Y^p_{\lambda_i} \cap \mathtt{NN}(x', \lambda_i(x')) \neq \varnothing$.

(b) If k is even or $i < l$, for any client x with $\kappa(x) \geq 2i$, there is a client $x' \in X$ within 3 hops of x in G_{λ_i} such that $Y^s_{\lambda_i} \cap \mathtt{NN}(x', \lambda_i(x')) \neq \varnothing$.

Going back to the non-uniform MMC problem, $Y^p_{\lambda_i}$ will be used to 1-cover the clients $\{x \in X \mid \kappa(x) \geq 2i - 1\}$, and $Y^s_{\lambda_i}$ will be used to 1-cover $\{x \in X \mid \kappa(x) \geq 2i\}$. Suppose that $\kappa(x_1) = 100, \kappa(x_2) = 50$ for some $x_1, x_2 \in X$. The plan is use each of the sets $Y^s_{\lambda_i}$, $Y^p_{\lambda_i}$ for $1 \leq i \leq 25$ to cover both x_1 and x_2 once. The additional demand for x_1 is met by using each of the server sets $Y^s_{\lambda_j}$, $Y^p_{\lambda_j}$ for $25 < j \leq 50$ to cover x_1 once.

In the remainder of this section, we establish Lemma 9.

7.2 Forming Nets From Filtered Clients

Roughly speaking, our approach is to extend the proof of Lemma 7, i.e. (a) compute a hierarchy of nets $X_{\lambda_1} \subseteq X_{\lambda_2} \subseteq \cdots \subseteq X_{\lambda_k}$, and (b) In each iteration $i = 1 \ldots k$, let each client in X_{λ_i} add one server to $Y^p_{\lambda_i}$ and one server to $Y^s_{\lambda_i}$. There is one obstacle that arises in this approach, and this motivates the following definition.

Definition 6. We say that client x_2 threatens client x_1 if

- $\kappa(x_1) > \kappa(x_2)$, and
- $\mathtt{NN}(x_1, \kappa(x_1) - \lfloor \kappa(x_2)/2 \rfloor) \cap \mathtt{NN}(x_2, \kappa(x_2) - \lfloor \kappa(x_2)/2 \rfloor) \neq \varnothing$.

Observe that $\mathtt{NN}(x_2, \kappa(x_2) - \lfloor \kappa(x_2)/2 \rfloor) = \mathtt{NN}(x_2, l)$, and thus the second condition informally says that some private servers of x_2 are also “inner” servers of x_1.

To help understand the definition, consider the following example: suppose that $\kappa(x_1) = 100$, $\kappa(x_2) = 50$, and x_2 threatens x_1. Thus, $\mathtt{NN}(x_1, 75) \cap \mathtt{NN}(x_2, 25) \neq \varnothing$. The plan for our algorithm is that will provide the coverage required by x_2 in the first 25 iterations, and the coverage required by x_1 in the first 50 iterations. Now suppose that x_2 is chosen in the net in the first 25 iterations. This precludes x_1 being in the net in these first 25 iterations. However, we would like to allow x_1 to enter the net in iteration 26, since x_2 is essentially finished at this point, whereas x_1 is not.

In each of the first 25 iterations, we would choose two servers for x_2, one of which would be a server from $\mathtt{NN}(x_2, 25) = \mathtt{NN}(x_2, l)$. We would like at most one of these two servers to belong to $\mathtt{NN}(x_1, 75)$, so that x_1 has enough nearby servers when it later enters the net. However, we cannot ensure this, since the condition $\mathtt{NN}(x_1, 75) \cap \mathtt{NN}(x_2, 25) \neq \varnothing$ means that a private server of x_2 can belong to $\mathtt{NN}(x_1, 75)$.

Therefore, as a preprocessing step, we compute a representative subset $\overline{X} \subseteq X$ in which no client threatens another:

Claim 7.1. We can compute in polynomial time a subset $\overline{X} \subseteq X$ of clients such that

- For any two clients x_1, x_2 such that x_2 threatens x_1, $x_1 \in \overline{X} \implies x_2 \notin \overline{X}$;
- For any client $x \in X \setminus \overline{X}$, there is an $x' \in \overline{X}$ such that x threatens x'.

Proof. Let ϕ be any ordering of the clients X such that the $\kappa(\cdot)$ values are non-increasing. Observe that if x_2 threatens x_1 then x_2 occurs after x_1 in ϕ. We initialize $\overline{X}$ to be empty, and assume all clients are initially unmarked. We process each client in X according to the ordering ϕ as follows: for each cllient x, perform the following actions if x is unmarked: 1) add x to $\overline{X}$ 2) mark all clients of X that threaten x.

It is easily checked that the resultant set of clients $\overline{X}$ satisfies the two properties.

We compute a hierarchy of nets on $\overline{X}$, instead of X. For any client $x \in X \setminus \overline{X}$, there is a client $x' \in \overline{X}$ such that x threatens x'. Such an x' will help deal with the coverage requirements of x. For each G_{λ_i}, we define H_{λ_i} as the subgraph of G_{λ_i} induced by $\overline{X}$ i.e. $H_{\lambda_i} = G_{\lambda_i}[\overline{X}]$. Recall the definition of G_{λ_i}, and observe that for $1 \leq j < i \leq k$, if (x, x') is an edge in G_{λ_i} it is also an edge in G_{λ_j}. The same holds for edges in H_{λ_i}.

We will construct a hierarchy of 3-nets for clients $\overline{X}$, using the family of graphs H_{λ_i}, obtaining an anolog of Claim 5.1.

Claim 7.2. There is a polynomial time algorithm that computes a hierarchy

$$X_{\lambda_1} \subseteq X_{\lambda_2} \subseteq \cdots \subseteq X_{\lambda_k},$$

where each $X_{\lambda_i} \subseteq \overline{X}$ is a 3-net of H_{λ_i}.

7.3 Computing Disjoint Server Subsets

Our algorithm for computing the family $\mathcal{F}$ of server subsets, as stated in Lemma 9, is described in Algorithm 7.1. In many ways, it is analagous to Section 5.2, so we only highlight the key differences. One syntactic feature worth drawing attention to is that index i goes up from 1 in the for loop in Line 3, as opposed to the for loop in Line 3 of Section 5.2 where it decreased starting from k. Thus, iteration i in Algorithm 7.1 corresponds to iteration $k - (i - 1)$ in Section 5.2.

In iteration i, we consider each client $x_c \in X_{\lambda_i}$ in the for loop in Line 5, but we add the farthest available server in $\texttt{NN}(x_c, \kappa(x_c) - (i-1))$ to $Y^s_{\lambda_i}$ only if $\kappa(x_c) \geq 2i$, and any available server from $\texttt{NN}(x_c, l)$ to $Y^p_{\lambda_i}$ only if $\kappa(x_c) \geq 2i - 1$.

Algorithm 7.1 ComputeServerSubsets(X, Y, κ)

```
 1: $l \leftarrow \lceil k/2 \rceil$
 2: Compute $X_{\lambda_1} \subseteq X_{\lambda_2} \subseteq \cdots \subseteq X_{\lambda_k}$ using Claim 7.2.
 3: for $i = 1$ to $l$ do
 4:     Let $Y^s_{\lambda_i} \leftarrow \varnothing, Y^p_{\lambda_i} \leftarrow \varnothing$.
 5:     for all $x_c \in X_{\lambda_i}$ do
 6:         if $\kappa(x_c) \geq 2i$ then
 7:             $y_s \leftarrow$ farthest available server in $\texttt{NN}(x_c, \kappa(x_c) - (i-1))$.
 8:             $Y^s_{\lambda_i} \leftarrow Y^s_{\lambda_i} \cup \{y_s\}$. Mark $y_s$ as not available.
 9:         if $\kappa(x_c) \geq 2i - 1$ then
10:             $y_p \leftarrow$ any available server in $\texttt{NN}(x_c, l)$.
11:             $Y^p_{\lambda_i} \leftarrow Y^p_{\lambda_i} \cup \{y_p\}$. Mark $y_p$ as not available.
12: $\mathcal{F} \leftarrow \varnothing$
13: for $i = 1$ to $l$ do
14:     if $k$ is even or $i < l$ then
15:         $\mathcal{F} \leftarrow \mathcal{F} \cup \{Y^s_{\lambda_i}\}$
16:     $\mathcal{F} \leftarrow \mathcal{F} \cup \{Y^p_{\lambda_i}\}$
```

Assuming that servers are available when the algorithm looks for them, we can now establish Lemma 9. Fix an i such that $1 \leq i \leq l$, and assume that k is even. Let $Z = \{x \in X \mid \kappa(x) \geq 2i\}$.

To establish part (3) of Lemma 9, we want to show that for any client in Z, there is a client $\bar{x}$ within 3 hops of this client in G_{λ_i} such that $Y^s_{\lambda_i}$ contains a server from $\texttt{NN}(\bar{x}, \lambda_i(\bar{x}))$. Let us first consider the case of a client $x \in Z$ that also belongs to $\overline{X}$, and hence is a vertex in H_{λ_i}. Since X_{λ_i} is a 3-net in H_{λ_i}, there is a path π in H_{λ_i}

with at most 2 edges (and 3 vertices) that connects x to some vertex $\bar{x} \in X_{\lambda_i}$. Let $\delta(y, \rho_{\lambda_i}(y))$ be the biggest ball in outer cover ρ_{λ_i} that serves at least one vertex on path π. Suppose that it serves vertex $\hat{x} \in \pi$. ($\hat{x}$ could be the same as x or $\bar{x}$.) Note that vertices x' in H_{λ_i} with $i > \lceil \kappa(x')/2 \rceil$ are isolated. Thus, $\kappa(x') \geq 2i - 1$ for any vertex x' on this path. We claim that in fact $\kappa(x') \geq 2i$ for any vertex x'. Otherwise, since $\kappa(x) \geq 2i$, there is an edge (x', x'') in π such that $\kappa(x') = 2i - 1$, and $\kappa(x'') \geq 2i$. Since (x', x'') is an edge in H_{λ_i}, we have

$$\mathtt{NN}(x', \kappa(x') - (i-1)) \cap \mathtt{NN}(x'', \kappa(x'') - (i-1)) \neq \varnothing.$$

As $i - 1 = \lfloor \kappa(x')/2 \rfloor$, we see that x' threatens x'', a contradiction. We conclude that $\kappa(x') \geq 2i$ for any vertex x' on π. Thus, $\kappa(\bar{x}) \geq 2i$, and Algorithm 7.1 adds a server from $\mathtt{NN}(\bar{x}, \kappa(\bar{x}) - (i-1))$ to $Y^s_{\lambda_i}$ in Line 7.

Now consider an arbitrary client $x_1 \in Z \setminus \overline{X}$. There is a client $x \in \overline{X}$ such that x_1 threatens x. Thus, $\kappa(x) \geq \kappa(x_1)$, so $x \in Z \cap \overline{X}$. Furthermore,

$$\mathtt{NN}(x, \kappa(x) - \lfloor \kappa(x_1)/2 \rfloor) \cap \mathtt{NN}(x_1, \kappa(x_1) - \lfloor \kappa(x_1)/2 \rfloor) \neq \varnothing.$$

Since $i - 1 \leq \lfloor \kappa(x_1)/2 \rfloor$, we have

$$\mathtt{NN}(x, \kappa(x) - (i-1)) \cap \mathtt{NN}(x_1, \kappa(x_1) - (i-1)) \neq \varnothing.$$

This implies that (x_1, x) is an edge in G_{λ_i}. Using the preceding argument, we can prove there is a client $\bar{x} \in X$ that is 2 hops away from x in G_{λ_i}, such that $Y^s_{\lambda_i}$ has a server added to it from $\mathtt{NN}(\bar{x}, \lambda_i(\bar{x}))$. We can thus infer that $\bar{x}$ is 3 hops aways from x_1 in G_{λ_i}. Thus, if k is even, for any client x such that $\kappa(x) \geq 2i$ there is a client $\bar{x} \in X$

within 3 hops of x in G_{λ_i}, such that $Y^s_{\lambda_i} \cap \texttt{NN}(\bar{x}, \lambda_i(\bar{x})) \neq \varnothing$. If k is odd, a similar argument can be made for $i < l$. This completes the proof of part $(3b)$ of Lemma 9, predicated on server availability.

Part $(3a)$ of Lemma 9 is established in a similar way. The argument is actually simpler, because we do not need to argue $\kappa(\bar{x}) \geq 2i$; it suffices that $\kappa(\bar{x}) \geq 2i - 1$. Combined, this establishes Lemma 9, assuming server availability, which we prove subsequently.

7.4 Server Availability

In this section, we show that Algorithm 7.1 finds available servers when it looks for them in Line 7 and Line 10. We define the *threshold level* of a client $x \in \overline{X}$ (denoted by $\texttt{th}(x)$) as the smallest i for which x belongs to the net X_{λ_i}. (Some clients in $\overline{X}$ may not be part of any of the nets; when we refer to the threshold level of a client, we implicitly assume that it is in some net, in particular, X_{λ_k}.) For client $x \in X$ and iteration $1 \leq i \leq \lceil \kappa(x)/2 \rceil$ of the for loop in Line 3, we define $A_i(x)$ to be the set of available servers within $\texttt{NN}(x, \kappa(x) - (i-1))$ at the *beginning* of iteration i. Note that $A_1(x) = \texttt{NN}(x, \kappa(x))$.

To establish availability, it suffices to consider clients $x \in \overline{X}$ for which $\texttt{th}(x) \leq \lceil \kappa(x)/2 \rceil$. For a client $x \in \overline{X}$ for which $\texttt{th}(x) > \lceil \kappa(x)/2 \rceil$, the algorithm never looks for available servers in its neighborhood in Line 7 and Line 10.

We now show that any such client x has enough available servers at the beginning of iteration $i = \texttt{th}(x)$ of the outer loop of Algorithm 7.1. This argument is where the intricacies of the non-uniform MMC and the need for resolving "threats" show up.

Claim 7.3. Let x be any client in $\overline{X}$ such that $\mathtt{th}(x) \leq \lceil \kappa(x)/2 \rceil$, and let $i = \mathtt{th}(x)$. Then

(a) $|A_i(x) \cap \mathtt{NN}(x, l)| \geq \lceil \kappa(x)/2 \rceil - (i-1)$.

(b) $|A_i(x)| \geq \kappa(x) - 2(i-1)$.

Proof. Consider any iteration $j < i$ of the outer loop in Line 3. The client x itself is not part of the net X_{λ_j}. Any client $x' \in X_{\lambda_j}$ for which some server is chosen in Line 7 or Line 10 must satisfy $j \leq \lceil \kappa(x')/2 \rceil$. For such a client x', if $\mathtt{NN}(x, \kappa(x) - (j-1)) \cap \mathtt{NN}(x', \kappa(x') - (j-1)) \neq \varnothing$, then (x, x') is an edge in H_{λ_j}. Since X_{λ_j} is a 3-net in H_{λ_j}, we conclude that there is at most one client $x' \in X_{\lambda_j}$ such that (a) some server is chosen in Line 7 or Line 10 for x', and (b) $\mathtt{NN}(x, \kappa(x) - (j-1)) \cap \mathtt{NN}(x', \kappa(x') - (j-1)) \neq \varnothing$. If there is no such client, we can conclude that in iteration j, no server in $\mathtt{NN}(x, \kappa(x) - (j-1))$ (and thus $\mathtt{NN}(x, \kappa(x) - (i-1))$) is made unavailable.

So let us assume that there is one such client x'. Next, we argue that $\mathtt{NN}(x, \kappa(x) - (i-1)) \cap \mathtt{NN}(x', l) = \varnothing$. Since server choices are made for x' in iteration j, we have $j \leq \lceil \kappa(x')/2 \rceil$.

First consider the case $i \leq \lceil \kappa(x')/2 \rceil$. Since x and x' are both part of the net X_{λ_i}, (x, x') is not an edge in H_{λ_i}. As $i \leq \lceil \kappa(x')/2 \rceil$ and $i \leq \lceil \kappa(x)/2 \rceil$, we may conclude that $\mathtt{NN}(x, \kappa(x) - (i-1)) \cap \mathtt{NN}(x', \kappa(x') - (i-1)) = \varnothing$. Also, since $i \leq \lceil \kappa(x')/2 \rceil$, we have $\mathtt{NN}(x', l) \subseteq \mathtt{NN}(x', \kappa(x') - (i-1))$. Thus, $\mathtt{NN}(x, \kappa(x) - (i-1)) \cap \mathtt{NN}(x', l) = \varnothing$.

Next, consider the case $i > \lceil \kappa(x')/2 \rceil$. Since $\lceil \kappa(x)/2 \rceil \geq i$, we have that $\kappa(x) > \kappa(x')$. Now, since x' does not threaten x, we conclude that $\texttt{NN}(x, \kappa(x) - \lfloor \kappa(x')/2 \rfloor) \cap \texttt{NN}(x', \kappa(x') - \lfloor \kappa(x')/2 \rfloor) = \varnothing$. Since $\texttt{NN}(x, \kappa(x) - (i-1)) \subseteq \texttt{NN}(x, \kappa(x) - \lfloor \kappa(x')/2 \rfloor)$, and $\texttt{NN}(x', \kappa(x') - \lfloor \kappa(x')/2 \rfloor) = \texttt{NN}(x', l)$, we conclude that $\texttt{NN}(x, \kappa(x) - (i-1)) \cap \texttt{NN}(x', l) = \varnothing$.

Thus, in iteration j, the server choice made for x' in Line 10 is not from $\texttt{NN}(x, \kappa(x) - (i-1))$, whereas the server choice made for x' in Line 7 may be from $\texttt{NN}(x, \kappa(x) - (i-1))$.

Since at most one server from $\texttt{NN}(x, \kappa(x) - (i-1))$ is made unavailable in each of the $i-1$ iterations before iteration i, we conclude that $A_i(x) \geq \kappa(x) - (i-1) - (i-1)$. The first assertion of the lemma also follows.

The next claim says that before every iteration $\texttt{th}(x) \leq i \leq \lceil \kappa(x)/2 \rceil$, there are enough available servers in $\texttt{NN}(x, \kappa(x) - (i-1))$. These are iterations in which x itself is part of the net, and the argument is identical to that of Claim 5.6.

Claim 7.4. Let $x \in \overline{X}$, and let $\texttt{th}(x) \leq i < \lceil \kappa(x)/2 \rceil$. Then

(a) $|A_{i+1}(x)| \geq |A_i(x)| - 2$

(b) If $|A_{i+1}(x)| = |A_i(x)| - 2$, then one of the servers in $A_i(x) \setminus A_{i+1}(x)$ is the farthest server in $A_i(x)$ from x.

We can now assert our final claim about server availability. The proof follows from Claim 7.3 and Claim 7.4 using arguments very similar to Claim 5.7.

Claim 7.5. Algorithm 7.1 finds an available server whenever it executes Line 10 or Line 7.

This completes our proof of Lemma 9.

7.5 Solving the Non-uniform MMC Problem

In this section, we describe a constant factor approximation for the non-uniform MMC problem. Recall that our input consists of two point sets X (clients) and Y (servers) in an arbitrary metric space $(X \cup Y, d)$, a function κ representing the coverage demand of each client, and the constant $\alpha \geq 1$.

Algorithm 7.2 NonUniformCover(X, Y, κ, α)

```
1: $k \leftarrow \max_{x \in X} \kappa(x), l \leftarrow \lceil k/2 \rceil$.
2: $\mathcal{F} \leftarrow$ ComputeServerSubsets$(X, Y, \kappa)$. $\{$Note that $\mathcal{F} = \{Y^s_{\lambda_1}, Y^p_{\lambda_1}, Y^s_{\lambda_2}, Y^p_{\lambda_2}, \dots\}$.$\}$
3: For each $y \in Y$, assign $r(y) \leftarrow 0$.
4: for $i = 1$ to $l$ do
5:     if $k$ is even or $i < l$ then
6:         Let $r_s$ be obtained by invoking Cover$(\cdot, Y^s_{\lambda_i}, \alpha)$ for clients $\{x \in X \mid \kappa(x) \geq 2i\}$.
7:         Let $r(y) \leftarrow r_s(y)$ for each $y \in Y^s_{\lambda_i}$.
8:     Let $r_p$ be obtained by invoking Cover$(\cdot, Y^p_{\lambda_i}, \alpha)$ for clients $\{x \in X \mid \kappa(x) \geq 2i - 1\}$.
9:     Let $r(y) \leftarrow r_p(y)$ for each $y \in Y^p_{\lambda_i}$.
10: return The assignment $r : Y \to \mathbb{R}^+$
```

Our algorithm first computes a family $\mathcal{F}$ consisting of k pairwise disjoint subsets of Y, using the algorithm of Lemma 9. It then invokes Cover$(\cdot, Y', \alpha)$ using a server subset from $\mathcal{F}$ and a selected subset of clients as follows. Note that $\mathcal{F} = \{Y^s_{\lambda_1}, Y^p_{\lambda_1}, Y^s_{\lambda_2}, Y^p_{\lambda_2}, \dots\}$. In the i-th iteration of the for loop in Line 4, we use servers in $Y^s_{\lambda_i}$ to 1-cover the clients with coverage demand at least $2i$, and servers in $Y^p_{\lambda_i}$ to 1-cover the clients with coverage demand at least $2i-1$. Notice that if k is odd and $i = l$, there are no clients with coverage demand at least $2i$.

The algorithm then returns r, the union of the k covers thus formed, which satisfies the coverage demand of each client (as the server subsets in $\mathcal{F}$ are pairwise disjoint). This union can be thought of as the combined assignment $r : Y \to \mathbb{R}^+$; for a server y not belonging to any subset in $\mathcal{F}$, we simply set $r(y)$ to 0.

7.6 Approximation Guarantee

To obtain an approximation guarantee for Algorithm 7.1, we first upper bound the cost of the covers returned in iteration i of the for loop in Line 4.

Claim 7.6. Assume that either (a) k is even and $1 \leq i \leq l$, or (b) k is odd and $1 \leq i < l$. Let ρ_{λ_i} be any λ_i-outer cover. There is a 1-cover of the clients $\{x \in X \mid \kappa(x) \geq 2i\}$ that uses servers from $Y^s_{\lambda_i}$ and has cost at most $16^\alpha \cdot \texttt{cost}(\rho_{\lambda_i})$.

Proof. Let $Z = \{x \in X \mid \kappa(x) \geq 2i\}$. Consider the set B of balls obtained by expanding each ball in the outer cover ρ_{λ_i} to 8 times its original radius. It suffices, as in the proof of Claim 6.1, to show the following claim.

Claim 7.7. For any client $x \in Z$, there is some ball in B that contains x as well as at least one server in $Y^s_{\lambda_i}$.

We now turn to the proof of Claim 7.7. Consider an arbitrary client $x \in Z$. By Lemma 9, there is a path π in G_{λ_i} with at most three edges that connects x to $\bar{x}$, such that $Y^s_{\lambda_i} \cap \mathtt{NN}(\bar{x}, \lambda_i(\bar{x})) \neq \varnothing$. Let $\delta(y, \rho_{\lambda_i}(y))$ be the biggest ball in outer cover ρ_{λ_i} that serves at least one vertex on path π. Suppose it serves vertex $\hat{x}$. Using the definition of λ_i, and the way we pick the ball $\delta(y, \rho_{\lambda_i}(y))$, we have that for any $x' \in \pi$,

$$d(x', \mathtt{nn}(x', \lambda_i)) \leq \rho_{\lambda_i}(y).$$

From the definition of G_{λ_i}, we have that for any edge (x', x'') in π,

$$d(x', x'') \leq d(x', \mathtt{nn}(x', \lambda_i)) + d(x'', \mathtt{nn}(x'', \lambda_i)) \leq 2\rho_{\lambda_i}(y) \tag{7.1}$$

By Lemma 9, $\mathtt{NN}(\bar{x}, \lambda_i(\bar{x})) \cap Y^s_{\lambda_i} \neq \varnothing$. Let $\bar{y}$ be an arbitrary server in $\mathtt{NN}(\bar{x}, \lambda_i(\bar{x})) \cap Y^s_{\lambda_i}$. Clearly,

$$d(\bar{x}, \bar{y}) \leq d(\bar{x}, \mathtt{nn}(\bar{x}, \lambda_i)) \leq \rho_{\lambda_i}(y).$$

We calculate

$$d(y, x) \leq d(y, \hat{x}) + \left(\sum_{(x', x'') \in \pi[\hat{x}, x]} d(x', x'') \right) \leq 7\rho_{\lambda_i}(y),$$

and

$$d(y, \bar{y}) \leq d(y, \hat{x}) + \left(\sum_{(x', x'') \in \pi[\hat{x}, \bar{x}]} d(x', x'') \right) + d(\bar{x}, \bar{y}) \leq 8\rho_{\lambda_i}(y).$$

Thus, the ball $\delta(y, 8\rho_{\lambda_i}(y))$ contains both x and $\bar{y} \in Y^s_{\lambda_i}$, completing the proof of Claim 7.7.

The following claim addresses the cost of the cover obtained using the server set $Y^p_{\lambda_i}$. Its proof is very similar to that of Claim 7.6.

Claim 7.8. Let $1 \leq i \leq l$ and ρ_{λ_i} be any λ_i-outer cover. There is a 1-cover of the clients $\{x \in X \mid \kappa(x) \geq 2i - 1\}$ that uses servers from $Y^p_{\lambda_i}$ and has cost at most $16^\alpha \cdot \mathtt{cost}(\rho_{\lambda_i})$.

We can now establish the approxmation guarantee for Algorithm 7.1 and the main result of this section.

Theorem 10. *Given point sets X and Y in a metric space $(X \cup Y, d)$ and a coverage function κ, Algorithm 7.1 runs in polynomial time and returns a κ-cover of X with cost at most $2 \cdot (16 \cdot 9)^{\alpha}$ times that of an optimal κ-cover.*

Proof. It is evident that the algorithm runs in polynomial time. It is also easy to check that the assignment r that it returns is a κ-cover, that is, each client x is covered at least $\kappa(x)$ times. Let r' be any optimal κ-cover. By Theorem 3 (in Chapter 3), there exists a λ_i-outer cover ρ_{λ_i}, for $1 \leq i \leq k$ such that

$$\sum_{i=1}^{k} \texttt{cost}(\rho_{\lambda_i}) \leq 3^{\alpha} \texttt{cost}(r').$$

From Claim 7.6 and Claim 7.8, and the fact that Cover$(\cdot, \cdot, \alpha)$ returns a 3^{α} approximation, we conclude that the cost of a 1-cover that is computed in iteration i of the for loop in Line 4 is at most $(16 \cdot 3)^{\alpha} \texttt{cost}(\rho_{\lambda_i})$. At most two 1-covers are computed in iteration i. Thus,

$$\begin{aligned}\texttt{cost}(r) &\leq 2 \cdot (16 \cdot 3)^{\alpha} \cdot \sum_{i=1}^{l} \texttt{cost}(\rho_{\lambda_i}) \\ &\leq 2 \cdot (16 \cdot 3)^{\alpha} \cdot \sum_{i=1}^{k} \texttt{cost}(\rho_{\lambda_i}) \\ &\leq 2 \cdot (16 \cdot 9)^{\alpha} \cdot \texttt{cost}(r').\end{aligned}$$

CHAPTER 8
THE t-MMC PROBLEM

In this chapter, we describe a natural generalization of the MMC problem, called the t-MMC problem. The input to this problem is similar to the MMC problem – the two point sets Y (servers) and X (clients) in an arbitrary metric space $(X \cup Y, d)$, a positive integer k that represents the coverage demand of each client, a constant α. There is an additional input, an integer t, that represents the upper bound on the number servers that can be opened or *used* in the solution.

A *k-cover using at most t servers* is a subset $Y' \subseteq Y$ such that $|Y'| \leq t$, together with an assignment $r : Y' \to \mathbb{R}^+$ that k-covers X. Here, the cost of the solution is defined as $\texttt{cost}(r) = \sum_{y \in Y'} (r(y))^\alpha$. Intuitively, the restriction of t is analogous to the cardinality restrictions imposed on the solutions in problems like t-center, t-median and so on.

Now, the goal of the t-MMC problem is to compute a minimum cost k-cover using at most t servers. In comparison to the MMC problem, the additional complexity arises from having to decide which t servers to use for k-covering X. We give an $O(1)$ approximation for this problem. Here, we assume that $k \leq |Y|$ and $k \leq t$, so that the given instance is feasible.

8.1 Algorithm

The $O(1)$ approximation algorithm for the t-MMC problem consists of the following steps.

1. We first compute a family $\mathcal{F} = \{Y^s_k, Y^p_k, Y^s_{k-1}, Y^p_{k-1}, \ldots\}$ consisting of k pairwise disjoint subsets of Y, using the algorithm of Lemma 7. For convenience, let us rename this family of servers as $\mathcal{F} = \{V_1, V_2, \cdots, V_k\}$ respectively.

2. For each $1 \leq i \leq k$, and for each $1 \leq t_i \leq t$, we compute a 1-cover of X, using at most t_i servers from V_i. Here, we use the polynomial time approximation algorithm of Charikar and Panigrahy [23] for computing 1-cover using at most t_i servers. Let us denote the solution returned by their algorithm by $S(V_i, t_i)$. Even though their algorithm is stated for the case of $\alpha = 1$, it generalizes to any $\alpha \geq 1$. It can be shown that the approximation guarantee of their algorithm is 5^α.

3. Let us call a k-tuple $(t_1, t_2, \ldots, t_k)$ a *valid k-tuple* if $1 \leq t_i \leq t$ for each i, and $\sum_{i=1}^k t_i \leq t$.

   We compute a valid k-tuple $(t^*_1, t^*_2, \ldots, t^*_k)$ that minimizes $\sum_{i=1}^k \texttt{cost}(S(V_i, t_i))$, over all valid k-tuples $(t_1, t_2, \cdots, t_k)$. Such a valid k-tuple can be computed in polynomial time using dynamic programming. We return $\bigcup_{i=1}^k S(V_i, t^*_i)$ as our solution.

8.2 Approximation Guarantee

It is easy to see that the algorithm described above runs in polynomial time. Also, for each $1 \leq i \leq k$, $S(V_i, t^*_i)$ 1-covers X using disjoint servers. Since the final solution $\bigcup_{i=1}^k S(V_i, t^*_i)$ obtained using dynamic programming is a valid k-tuple, the algorithm computes a k-cover of X that uses at most t servers.

For proving the approximation guarantee, we extract from the optimal solution to the t-MMC problem, the outer covers ρ_i for each $1 \leq i \leq k$ with some special properties. The following is an analogue of Theorem 3, however some new ideas are needed to handle the restriction on the number of servers that can be used in the resultant outer covers. The proof of the following theorem is given in Section 8.3.

Theorem 11. *Let $r' : Y' \to \mathbb{R}^+$ be an assignment that constitutes an optimal solution to the t-MMC problem, where $Y' \subseteq Y$ with $|Y'| \leq t$. For each $l \leq i \leq k$, we can find level i outer cover ρ_i that uses t'_i servers, such that*

- *If k is even, then*
  1. *$\sum_{i=l+1}^{k} \texttt{cost}(\rho_i) \leq 2 \cdot (3 \cdot 3)^{\alpha} \cdot \texttt{cost}(r')$, and*
  2. *$\sum_{i=l+1}^{k} 2 \cdot t'_i \leq t$.*
- *If k is odd, then*
  1. *$\sum_{i=l}^{k} \texttt{cost}(\rho_i) \leq 2 \cdot (3 \cdot 3)^{\alpha} \cdot \texttt{cost}(r')$, and*
  2. *$t'_l + \sum_{i=l+1}^{k} 2 \cdot t'_i \leq t$.*

Given an outer cover ρ_i that uses at most t'_i servers, the following claim constructs an inexpensive 1-cover of X using at most t'_i servers from Y_i. This will help us bound the cost of the solution returned by the algorithm from Section 8.1. This claim strengthens Claim 6.1, but the proof generalizes easily.

Claim 8.1. Assume that either (a) $l + 1 \leq i \leq k$ and Y_i is either Y_i^p or Y_i^s, or (b) k is odd, $i = l$ and $Y_i = Y_i^p$. Let ρ_i be an outer cover of level i using at most t'_i servers from servers from Y. Then there is a 1-cover of X that uses at most t'_i servers from

Y_i, and has cost at most $12^\alpha \cdot \texttt{cost}(\rho_i)$.

Now, we establish the approximation guarantee for the algorithm described in Section 8.1.

Theorem 12. *Given point sets X and Y is a metric space $(X \cup Y, d)$, and positive integers k and t such that $k \leq |Y|$ and $k \leq t$, the algorithm described in Section 8.1 runs in polynomial time, and returns a k-cover of X using at most t servers from Y, and with cost at most $4 \cdot (540)^\alpha$ times that of an optimal k-cover that uses at most t servers from Y.*

Proof. We focus on the case where k is even. The case where k is odd is similar, and is therefore omitted.

We have already argued that the algorithm runs in polynomial time, and the solution produced by the algorithm k-covers X using at most t servers.

Let $r' : Y' \to \mathbb{R}^+$ be any optimal assignment that k-covers X, where $Y' \subseteq Y$, with $|Y'| \leq t$. By Theorem 11, there exist outer covers ρ_i that use t'_i servers such that $\sum_{i=l+1}^{k} \texttt{cost}(\rho_i) \leq 2 \cdot (3 \cdot 3)^\alpha \cdot \texttt{cost}(r')$, with $\sum_{i=l+1}^{k} 2 \cdot t'_i \leq t$.

For each of (Y_i^p, t'_i, ρ_i) and (Y_i^s, t'_i, ρ_i), we use Claim 8.1, to argue that there exist two 1-covers from Y_i^p and Y_i^s respectively. These 1-covers have cost at most $(12)^\alpha \cdot \texttt{cost}(\rho_i)$ each, and each uses at most t'_i servers. Since the 5^α approximation of Charikar and Panigrahy [23] is used to get two 1-covers $S(Y_i^p, t'_i)$ and $S(Y_i^s, t'_i)$, we have that $\texttt{cost}(S(Y_i^p, t'_i)) \leq (12 \cdot 5)^\alpha \cdot \texttt{cost}(\rho_i)$ and $\texttt{cost}(S(Y_i^s, t'_i)) \leq (12 \cdot 5)^\alpha \cdot \texttt{cost}(\rho_i)$.

Note however that $\sum_{i=l+1}^{k} 2 \cdot t'_i \leq t$, so $(t'_k, t'_k, \cdots, t'_l, t'_l)$ is a valid k-tuple, and so the dynamic program of step 3 must have considered the solution

$$\left(\bigcup_{i=l+1}^{k} S(Y_i^p, t_i')\right) \cup \left(\bigcup_{i=l+1}^{k} S(Y_i^s, t_i')\right).$$

Since the cost of the solution output by the dynamic program is at most the cost of solution corresponding to this tuple, we have that,

$$\begin{aligned}\sum_{i=1}^{k} \texttt{cost}(S(V_i, t_i^*)) &\leq \sum_{i=l+1}^{k} \left(\texttt{cost}(S(Y_i^p, t_i')) + \texttt{cost}(S(Y_i^s, t_i'))\ \right) \\ &\leq 2 \cdot (12 \cdot 5)^\alpha \cdot \sum_{i=l+1}^{k} \texttt{cost}(\rho_i) \\ &\leq 2 \cdot (12 \cdot 5)^\alpha \cdot 2 \cdot 9^\alpha \cdot \texttt{cost}(r') = 4 \cdot (540)^\alpha \cdot \texttt{cost}(r').\end{aligned}$$

8.3 The Outer Cover Lower Bound for the t-MMC Problem

In this section, we prove Theorem 11, which generalizes Theorem 3 in the case when the size of the outer covers are restricted to satisfy certain properties. For convenience, we restate the theorem.

Theorem 11. *Let $r' : Y' \to \mathbb{R}^+$ be an assignment that constitutes an optimal solution to the t-MMC problem, where $Y' \subseteq Y$ with $|Y'| \leq t$. For each $l \leq i \leq k$, we can find level i outer cover ρ_i that uses t_i' servers, such that*

- *If k is even, then*
  1. *$\sum_{i=l+1}^{k} \texttt{cost}(\rho_i) \leq 2 \cdot (3 \cdot 3)^\alpha \cdot \texttt{cost}(r')$, and*
  2. *$\sum_{i=l+1}^{k} 2 \cdot t_i' \leq t$.*
- *If k is odd, then*
  1. *$\sum_{i=l}^{k} \texttt{cost}(\rho_i) \leq 2 \cdot (3 \cdot 3)^\alpha \cdot \texttt{cost}(r')$, and*

2. $t'_l + \sum_{i=l+1}^{k} 2 \cdot t'_i \leq t$.

Proof. For simplicity, we prove the theorem only for the case where k is even. The proof of the case where k is odd is similar, and is therefore omitted.

Note that any feasible solution to the t-MMC problem is also feasible for the MMC problem. Therefore, we can use Theorem 3 to extract from the assignment r', the outer covers $\bar{\rho}_i$ for each $1 \leq i \leq k$, such that $\sum_{i=1}^{k} \texttt{cost}(\bar{\rho}_i) \leq 3^\alpha \cdot \texttt{cost}(r')$.

These outer covers satisfy the first property, but they may not satisfy the second property. However, if the outer cover $\bar{\rho}_i$ uses $\bar{t}_i$ servers, then it is easy to verify that the proof of Theorem 3 ensures that $\sum_{i=1}^{k} \bar{t}_i \leq t$.

Let us order the above outer covers $\bar{\rho}_i$ in a nondecreasing order of the number of servers used, and rename them according to this ordering as $r_k, r_{k-1}, \ldots, r_1$. Let t'_i denote the number of servers used by the outer cover r_i. To transform the outer covers $\bar{\rho}_i$ into the outer covers ρ_i that satisfy the second property of the theorem, we need the following claim.

Claim 8.2. Let $\bar{\rho}_i$ be an outer cover of level i, $l \leq i \leq k$, and r be any 1-cover that uses at most t' servers. Then there is an outer cover ρ_i of level i that uses at most t' servers, and $\texttt{cost}(\rho_i) \leq 3^\alpha \cdot (\texttt{cost}(\bar{\rho}_i) + \texttt{cost}(r))$.

Proof. Let B and B' be the set of balls corresponding to the outer cover $\bar{\rho}_i$ and the 1-cover r respectively. We describe an iterative procedure to compute a set R_i of balls, which is initially empty. Each ball in B and B' is initially "unmarked". Pick any unmarked ball $b'_j \in B'$, and suppose $b'_j = \delta(y'_j, r_{b'_j})$. Let B_j denote the set unmarked of balls from B that serve a client $x \in X$ that is also covered by b'_j. Let r'_j be the maximum radius from the set of balls $B_j \cup \{b'_j\}$. Add the ball $\delta(y'_j, 3r'_j)$ to the set R_i. Mark the ball b'_j from B' and the balls B_j from B, and repeat the above process until all balls from B' are marked.

We argue that at the end of this process, the radius assignment ρ_i corresponding to R_i is an outer cover of level i with the claimed properties. Without loss of generality, we assume that each ball $b \in B$ serves some client $x \in X$. Consider a client $x \in X$, and a ball $b \in B$ that serves it. Since r is a valid 1-cover, there exists a ball $b' \in B'$ that also covers x. So, if b was not considered in any iteration before, it will be considered in the iteration when b' is marked, and both b and b' will be marked by the end of that iteration. Thus, at the end of the above process, all balls $b \in B$ will be marked. Also, for each ball $b' \in B'$, we add exactly one ball to R_i. Therefore, $|R_i| = |B'| \leq t'$.

Now we argue that ρ_i is an outer cover of level i. Consider any client $x \in X$. Since $\bar{\rho}_i$ is an outer cover of level i, there exists a ball $b \in B$ centered at some $y \in Y$ with radius $r_b \geq d(x, y_i(x))$ that covers x. Using the argument from the above paragraph, such a ball b was marked in some iteration. Suppose the ball from the set B' that was marked in that iteration was $b'_j = \delta(y'_j, r'_{b_j})$, and the ball $\delta(y'_j, 3r'_j)$ was added to R_i. Now,

$$d(x, y'_j) \leq \; d(x, y) + d(y, x') + d(x', y'_j) \leq \; r_b + r_b + r_{b'_j} \leq 3r'_j$$

Here, $x' \in X$ is a common client served by b in $\bar{\rho}_i$ and covered by b' in r. Note that x' may or may not be same as x. The last inequality follows because of the choice of

r'_j. Therefore, the ball $\delta(y'_j, 3r'_j)$ covers x. Also since $d(x, y_i(x)) \le r_b \le r'_j \le 3r'_j$, the ball $\delta(y'_j, 3r'_j)$ also serves x. Thus, it follows that ρ_i is an outer cover of level i.

The ball added to R_i in a particular iteration has radius $3r'_j$, where r'_j is the maximum radius from the set $B_j \cup \{b'_j\}$. Now, considering all such balls in R_i, the bound on the cost of ρ_i follows.

Now, we use Claim 8.2 for pairs of outer covers $(\bar{\rho}_i, r_i)$ to get an outer cover ρ_i of level i, for each $l+1 \le i \le k$ with the desired upper bounds on the cost and the number of servers used. Now,

$$\begin{aligned}\sum_{i=l+1}^{k} \texttt{cost}(\rho_i) &\le \sum_{i=l+1}^{k} 3^{\alpha} \cdot (\texttt{cost}(\bar{\rho}_i) + \texttt{cost}(r_i)) \\ &\le \sum_{i=1}^{k} 3^{\alpha} \cdot (\texttt{cost}(\bar{\rho}_i) + \texttt{cost}(r_i)) \\ &= 2 \cdot 3^{\alpha} \sum_{i=1}^{k} \texttt{cost}(\bar{\rho}_i) \\ &\le 2 \cdot (3 \cdot 3)^{\alpha} \texttt{cost}(r')\end{aligned}$$

The third inequality follows due to the fact that the set of outer covers $\{r_1, r_2, \cdots, r_k\}$ is same as the set of original outer covers $\{\bar{\rho}_1, \bar{\rho}_2, \cdots, \bar{\rho}_k\}$. Note that for each $l+1 \le i \le k$, ρ_i uses at most t'_i servers. Since the outer covers r_i are ordered in a non-decreasing order of the number of servers used, $\sum_{i=l+1}^{k} 2 \cdot t'_i \le \sum_{i=1}^{k} t'_i \le t$, and the second property follows.

CHAPTER 9
CONCLUSION AND OPEN PROBLEMS

In this document, we have presented several algorithms for covering problems, and how we can exploit combinatorial and geometric properties of the problem to get constant factor approximations. That being said, there is much work that is left to be done in this area. We conclude this thesis with a discussion of open problems related to our work.

9.1 APX-Hardness of the MMC Problem

We consider the hardness of approximation for the MMC problem in general metric space. For concreteness, we focus on resolving the question for the uniform MMC problem, in which the coverage demand for each client is k.

We first look at the hardness of the MMC problem for $k = 1$ i.e. when each client $x \in X$ needs to be covered by at least 1 ball centered at any of the servers in Y. We refer to this problem as the *Minimum Cost Covering problem* (MCC). The MCC problem for $\alpha = 1$ is exactly solvable in polynomial time in a geometric setting i.e. for $l_1, l_{\inf}$ metrics and has a PTAS for the l_2 metric. For $\alpha > 1$, the MCC problem is $\mathbb{NP}$-hard as shown by Alt et al. [3]. Charikar and Panigrahy [23] had described a primal-dual based polynomial time approximation algorithm for the MCC problem with an approximation ratio of 3^α. Recently, it was shown by Bandyapadhyay and Varadarajan [8] that the MCC problem admits a QPTAS i.e. a $(1+\varepsilon)$ approximation algorithm that runs in $O(2^{(\log(|X|\cdot|Y|)/\varepsilon)^c})$ time, where $c > 0$ is a constant. Thus, it is unlikely that the MCC problem is $\mathbb{APX}$-hard.

For general values of k, not much is known about the hardness of approximation for the MMC problem. For $\alpha \geq \log(|X|)$, Bandyapadhyay and Varadarajan [8] showed that the MCC problem cannot be approximated beyond $O(\log(|X|))$ under standard complexity assumptions. The same hardness result obviously holds for the MMC problem as well.

Open Problem 1. Is the MMC problem $\mathbb{APX}$-hard for constant values of α?

9.2 An $O(1)$ Approximation for the Capacitated MCC Problem

In the MMC problem, we assume that the balls of radius $r(y)$ centered on each server $y \in Y$ can serve every element it covers (i.e. the clients in X). Realistically, that may not be possible as each server may have the capability of serving a limited number of clients, irrespective of the number of clients contained in the ball centered on that server. Hence, *capacitated* versions of the set cover problem are considered, in which each subset can only serve a limited number of clients amongst the clients it covers. We note that any approximation algorithm for a capacitated covering problem also holds for the uncapacitated version, by setting all capacities to infinity. Furthermore, a capacitated covering problem can either be *soft-capacitated* (one is allowed to make several copies of a subset) or *hard-capacitated* (one can only use the subsets provided as input). Unless otherwise mentioned, we would only be considering hard-capacitated problems in the remainder of this section.

In non-geometric settings, many approximation algorithms have been developed for capacitated covering problems. Wolsey [65] described a logarithmic approximation for the capacitated set cover problem using a greedy algorithm. A different greedy algorithm that gives the same approximation guarantee was presented by Bar-Ilan et al. [11], who also extended the classical approximation algorithm for the metric k-center problem to derive a 10-approximation algorithm for the capacitated metric k-center problem. This guarantee was later improved to 6 by Khuller and Sussmann [50]. For the capacitated vertex cover problem, Chuzhoy and Naor [26] showed that while weighted vertex cover is as hard as set cover to approximate, the unweighted vertex cover with capacities admits a 3-approximation. The latter result was improved to a 2-approximation by Gandhi et al. [35] using randomized rounding. For the capacitated metric facility location (CFL) problem, there are multiple algorithms that give a constant factor approximation. Using local search, one can get a 3-approximation for CFL with uniform capacities [2] and a 5-approximation for the case when the capacities are different [9]. For the latter version, a LP-based approach also gives a constant factor approximation [4] - the factor being at least 5, matching the local search bound.

In contrast, relatively little progress has been made for capacitated covering problems in a geometric setting. A QPTAS for the capacitated k-median problem in the Euclidean plane was derived by Arora et al. [6] by a modification of the dynamic programming formulation of PTAS for the uncapacitated version of the problem. Hochbaum and Maass [45] gave a PTAS for the problem of covering points in the plane using minimum number of unit balls, but the capacitated version of the same problem (where each unit ball could cover at most c points) had remained open for about 30 years, till recently Ghasemi and Razzazi [36] gave a PTAS for it.

In the previous chapters, we have considered the MMC problem without capacity constraints i.e. each server disk is able to serve all clients that are contained within it. In such a setting, the MMC problem admits a constant factor approximation. We would like to know if a similar result could be obtained if the servers were constrained to serve a limited number of clients. We formally define the capacitated version of the MMC problem as follows.

Definition 7. Given are a set of points X (clients), a set of points Y (servers), a metric d on the set $X \cup Y$, a coverage function $\kappa : X \to \mathbb{Z}_+ \cup \{0\}$, a capacity function $\lambda : Y \to \mathbb{Z}_+ \cup \{0\}$ and a constant $\alpha \geq 1$. An assignment $r : Y \to \mathbb{R}^+$ corresponds to "building" a disk of radius r_y centered at each $y \in Y$. Capacity constraints of servers are represented by the assignment $z : Y \times X \to \{0, 1\}$. For any integer j, a client x is defined to be j-covered if it is covered by at least j server disks without violating capacity constraints of the servers i.e.

$$|\{y \in Y \mid z(y,x) = 1, d(y,x) \leq r_y\}| \geq j.$$

A feasible solution is an assignment (r, z) such that

1. Each client $x \in X$ is $\kappa(x)$ covered.
2. Each server $y \in Y$ serves most $\lambda(y)$ clients i.e. $\sum\limits_{x \in X} z(y,x) \leq \lambda(y)$.

The goal is to compute a feasible solution that minimizes $\sum\limits_{y \in Y} r_y^\alpha$. This is defined as the non-uniform capacitated MMC problem.

For simplicity, we look at the uniform capacitated MMC problem where $\kappa(x) = 1, \forall x \in X$ i.e. the capacitated version of the MCC problem. We note that planar MCC problem can be reduced to the problem of covering points with weighted disks. This is possible by considering all disks centered on each server as part of the input to the weighted disk set cover problem, and then discarding all but the largest disk centered at each server from the computed solution. While attempting to reduce the capacitated MCC problem to the weighted capacitated set cover problem, this straightforward strategy does not work for the following reason. Consider a solution in which two disks are centered on the same server. Discarding the smaller of the two disks and reassigning all the clients served by it to the larger disk may violate the capacity constraints of the latter, making the solution infeasible.

The standard LP formulation of the capacitated MCC problem has an unbounded integrality gap, as can be seen by the following example. Consider two servers a and b located unit distance apart, with capacity of a being $n-1$ and that of b being 1. We also have n clients co-located with a, each needing to be served by a disk that covers it and is centered at one of the servers, without violating the server capacity constraints. The only integral solution that satisfies all constraints is as follows: a disk of radius 1 centered at b that serves one client co-located with y, and a disk of radius 0 centered on a that serves the remaining $n-1$ clients. The cost of this solution is 1 (the radius of the disk centered on b). We now look at the optimal LP solution. An optimal LP solution that can open disks fractionally would open the disk of radius 1 centered at b by the fraction $\frac{1}{n}$, sufficient to serve one client completely, and open the other disk of radius 0 fully, thereby incurring a total cost of $1/n$. Thus, the ratio of the costs between the integral solution and the fractional LP solution is at least n, rendering an approach via the standard LP relaxation unviable. The question then arises - can we extend the combinatiorial technique used in Chapter 2 to get a constant factor approximation for the capacitated MCC problem?

Open Problem 2. For the uniform capacitated MCC problem, can there be a constant factor approximation when the input point sets are in:

1. Euclidean plane
2. Arbitrary metric space

REFERENCES

[1] A. K. Abu-Affash, P. Carmi, M. J. Katz, and G. Morgenstern. Multi cover of a polygon minimizing the sum of areas. *Int. J. Comput. Geometry Appl.*, 21(6): 685–698, 2011.

[2] A. Aggarwal, A. Louis, M. Bansal, N. Garg, N. Gupta, S. Gupta, and S. Jain. A 3-approximation algorithm for the facility location problem with uniform capacities. *Math. Program.*, 141(1-2):527–547, 2013.

[3] H. Alt, E. M. Arkin, H. Brönnimann, J. Erickson, S. P. Fekete, C. Knauer, J. Lenchner, J. S. B. Mitchell, and K. Whittlesey. Minimum-cost coverage of point sets by disks. In *Proceedings of the Symposium on Computational Geometry, (SoCG)*, pages 449–458, 2006.

[4] H. An, M. Singh, and O. Svensson. Lp-based algorithms for capacitated facility location. In *FOCS*, pages 256–265, 2014.

[5] B. Aronov, E. Ezra, and M. Sharir. Small-size epsilon-nets for axis-parallel rectangles and boxes. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC*, pages 639–648, 2009.

[6] S. Arora, P. Raghavan, and S. Rao. Approximation schemes for euclidean k-medians and related problems. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing*, pages 106–113, 1998.

[7] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for k-median and facility location problems. *SIAM J. Comput.*, 33(3):544–562, 2004.

[8] S. Bandyapadhyay and K. R. Varadarajan. Approximate clustering via metric partitioning. In *27th International Symposium on Algorithms and Computation, ISAAC*, pages 15:1–15:13, 2016.

[9] M. Bansal, N. Garg, and N. Gupta. A 5-approximation for capacitated facility location. In *Algorithms - ESA 2012 - 20th Annual European Symposium*, pages 133–144, 2012.

[10] N. Bansal and K. Pruhs. Weighted geometric set multi-cover via quasi-uniform sampling. In *Algorithms - ESA 2012 - 20th Annual European Symposium*, pages 145–156, 2012.

[11] J. Bar-Ilan, G. Kortsarz, and D. Peleg. How to allocate network centers. *J. Algorithms*, 15(3):385–415, 1993.

[12] R. Bar-Yehuda and D. Rawitz. On the equivalence between the primal-dual schema and the local ratio technique. *SIAM J. Discrete Math.*, 19(3):762–797, 2005.

[13] R. Bar-Yehuda and D. Rawitz. A note on multicovering with disks. *Comput. Geom.*, 46(3):394–399, 2013.

[14] M. Bellare, S. Goldwasser, C. Lund, and A. Russeli. Efficient probabilistically checkable proofs and applications to approximations. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing STOC*, pages 294–304, 1993.

[15] B. Ben-Moshe, M. J. Katz, and J. S. B. Mitchell. A constant-factor approximation algorithm for optimal 1.5d terrain guarding. *SIAM J. Comput.*, 36(6): 1631–1647, 2007.

[16] S. Bhowmick, K. Varadarajan, and S. Xue. A constant-factor approximation for multi-covering with disks. In *Proceedings of the Symposium on Computational Geometry (SoCG)*, pages 243–248, 2013.

[17] S. Bhowmick, K. R. Varadarajan, and S. Xue. A constant-factor approximation for multi-covering with disks. *JoCG*, 6(1):220–234, 2015.

[18] S. Bhowmick, T. Inamdar, and K. R. Varadarajan. Improved approximation for metric multi-cover. *CoRR*, abs/1602.04152, 2016.

[19] V. Bilò, I. Caragiannis, C. Kaklamanis, and P. Kanellopoulos. Geometric clustering to minimize the sum of cluster sizes. In *Proceedings of the European Symposium on Algorithms (ESA)*, pages 460–471, 2005.

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

[21] J. Byrka, A. Srinivasan, and C. Swamy. Fault-tolerant facility location: a randomized dependent lp-rounding algorithm. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 244–257. Springer, 2010.

[22] T. M. Chan, E. Grant, J. Könemann, and M. Sharpe. Weighted capacitated, priority, and geometric set cover via improved quasi-uniform sampling. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1576–1585, 2012.

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

[24] C. Chekuri, K. L. Clarkson, and S. Har-Peled. On the set multi-cover problem in geometric settings. In *Proceedings of the 25th ACM Symposium on Computational Geometry*, pages 341–350, 2009.

[25] F. A. Chudak and D. B. Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. *SIAM J. Comput.*, 33(1):1–25, 2003.

[26] J. Chuzhoy and J. Naor. Covering problems with hard capacities. *SIAM J. Comput.*, 36(2):498–515, 2006.

[27] K. L. Clarkson. New applications of random sampling in computational geometry. *Discrete & Computational Geometry*, 2:195–222, 1987.

[28] K. L. Clarkson and K. R. Varadarajan. Improved approximation algorithms for geometric set cover. *Discrete & Computational Geometry*, 37(1):43–58, 2007.

[29] S. Doddi, M. V. Marathe, S. S. Ravi, D. S. Taylor, and P. Widmayer. Approximation algorithms for clustering to minimize the sum of diameters. *Nord. J. Comput.*, 7(3):185–203, 2000.

[30] S. Eidenbenz, C. Stamm, and P. Widmayer. Inapproximability results for guarding polygons and terrains. *Algorithmica*, 31(1):79–113, 2001.

[31] G. Even, D. Rawitz, and S. Shahar. Hitting sets when the vc-dimension is small. *Inf. Process. Lett.*, 95(2):358–362, 2005.

[32] E. Ezra, B. Aronov, and M. Sharir. Improved bound for the union of fat triangles. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1778–1785, 2011.

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

[34] A. Freund and D. Rawitz. Combinatorial interpretations of dual fitting and primal fitting. In *Approximation and Online Algorithms, First International Workshop, WAOA*, pages 137–150, 2003.

[35] R. Gandhi, E. Halperin, S. Khuller, G. Kortsarz, and A. Srinivasan. An improved approximation algorithm for vertex cover with hard capacities. *J. Comput. Syst. Sci.*, 72(1):16–33, 2006.

[36] T. Ghasemi and M. Razzazi. A PTAS for the cardinality constrained covering with unit balls. *Theor. Comput. Sci.*, 527:50–60, 2014.

[37] M. Gibson, G. Kanade, E. Krohn, I. A. Pirwani, and K. R. Varadarajan. On metric clustering to minimize the sum of radii. *Algorithmica*, 57(3):484–498, 2010.

[38] M. Gibson, G. Kanade, E. Krohn, I. A. Pirwani, and K. R. Varadarajan. On clustering to minimize the sum of radii. *SIAM J. Comput.*, 41(1):47–60, 2012.

[39] S. Guha and S. Khuller. Greedy strikes back: Improved facility location algorithms. *J. Algorithms*, 31(1):228–248, 1999.

[40] S. Guha, A. Meyerson, and K. Munagala. A constant factor approximation algorithm for the fault-tolerant facility location problem. *Journal of Algorithms*, 48(2):429–440, 2003.

[41] M. Hajiaghayi, W. Hu, J. Li, S. Li, and B. Saha. A constant factor approximation algorithm for fault-tolerant k-median. *ACM Transactions on Algorithms (TALG)*, 12(3):36, 2016.

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

[43] S. Har-Peled and M. Lee. Weighted geometric set cover problems revisited. *JoCG*, 3(1):65–85, 2012.

[44] D. Haussler and E. Welzl. epsilon-nets and simplex range queries. *Discrete & Computational Geometry*, 2:127–151, 1987.

[45] D. S. Hochbaum and W. Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *J. ACM*, 32(1):130–136, 1985.

[46] K. Jain and V. 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.

[47] K. Jain, M. Mahdian, and A. Saberi. A new greedy approach for facility location problems. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing*, pages 731–740, 2002.

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

[49] R. M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations*, pages 85–103, 1972.

[50] S. Khuller and Y. J. Sussmann. The capacitated K-center problem. *SIAM J. Discrete Math.*, 13(3):403–418, 2000.

[51] S. Khuller, R. Pless, and Y. J. Sussmann. Fault tolerant k-center problems. *Theoretical Computer Science*, 242(1):237–245, 2000.

[52] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman. Analysis of a local search heuristic for facility location problems. *J. Algorithms*, 37(1):146–188, 2000.

[53] V. S. A. Kumar and H. Ramesh. Covering rectilinear polygons with axis-parallel rectangles. *SIAM J. Comput.*, 32(6):1509–1541, 2003.

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

[55] S. Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Inf. Comput.*, 222:45–58, 2013.

[56] J. Lin and J. S. Vitter. Approximation algorithms for geometric median problems. *Inf. Process. Lett.*, 44(5):245–249, 1992.

[57] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *J. ACM*, 41(5):960–981, 1994.

[58] N. H. Mustafa and S. Ray. Improved results on geometric hitting set problems. *Discrete & Computational Geometry*, 44(4):883–895, 2010.

[59] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing STOC*, pages 475–484, 1997.

[60] D. B. Shmoys, É. Tardos, and K. Aardal. Approximation algorithms for facility location problems (extended abstract). In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, pages 265–274, 1997.

[61] C. Swamy and D. B. Shmoys. Fault-tolerant facility location. *ACM Transactions on Algorithms (TALG)*, 4(4):51, 2008.

[62] K. R. Varadarajan. Epsilon nets and union complexity. In *Proceedings of the 25th ACM Symposium on Computational Geometry, Aarhus, Denmark, June 8-10, 2009*, pages 11–16, 2009.

[63] K. R. Varadarajan. Weighted geometric set cover via quasi-uniform sampling. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC*, pages 641–648, 2010.

[64] V. V. Vazirani. *Approximation algorithms*. Springer, 2001.

[65] L. A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982.