Theses and Dissertations

Spring 2019

Digging deeper into clustering and covering problems

Sayan Bandyapadhyay

*University of Iowa*

Follow this and additional works at: [https://ir.uiowa.edu/etd](https://ir.uiowa.edu/etd)

Copyright © 2019 Sayan Bandyapadhyay

This dissertation is available at Iowa Research Online: [https://ir.uiowa.edu/etd/6700](https://ir.uiowa.edu/etd/6700)

**Recommended Citation**


[https://doi.org/10.17077/etd.0cx5-cr9b](https://doi.org/10.17077/etd.0cx5-cr9b)

Follow this and additional works at: [https://ir.uiowa.edu/etd](https://ir.uiowa.edu/etd)
DIGGING DEEPER INTO CLUSTERING AND COVERING PROBLEMS

by

Sayan Bandyapadhyay

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 2019

Thesis Supervisor: Professor Kasturi Varadarajan
This thesis records my research works of past five and half years, which are made possible due to generous support of a number of individuals. I use this opportunity to acknowledge their effort.

First of all, I would like to express my sincere gratitude to my advisor, Prof. Kasturi Varadarajan, without whom this thesis would not have been possible. I would like to thank him for giving me the opportunity of working with him and for being my thesis supervisor. I am extremely fortunate to have such an erudite and quick-witted person like him as my mentor. For the past five and half years, his relentless support and encouragement have helped me to reach at this stage. He has always been easily accessible to me in spite of his busy schedule. Above all, he has always encouraged me to be an independent researcher. I am indebted to him for several discussions and suggestions on my works in which he was not involved. Due to his generous funding it was possible for me to attend a conference to present my work in which he was not even involved. It is so rare nowadays to find such a selfless and kind person like him.

Then, I would like to thank all the other members of my thesis committee: Prof. Sriram Pemmaraju, Prof. Suely Oliveira, Prof. Sukumar Ghosh, and Prof. Xiaodong Wu. Their cooperation and coordination have made my life much easier while scheduling my exams. I appreciate the time and effort they spent on me in spite of being busy with their work. I specially thank Prof. Pemmaraju for being an excellent teacher and for giving me the opportunity of doing research with him.
Also, I really appreciate his work as the Director of graduate studies. I thank Prof. Oliveira for her support and guidance while I was a teaching assistant under her. I thank Prof. Ghosh for being so nice and friendly to me. It is always fun chatting with him. Lastly, I thank Prof. Wu for serving on my committee.

Thereafter, I would like to thank my past and present officemates Vivek, James, Santanu, Rahil, Talal, Tanmay and Shreyas. Doing research would not have been this fun without them. I specially thank Santanu, Tanmay and Shreyas for their time and effort on our collaborative projects. By discussing and sharing ideas with them, I have learnt a lot, which had had a great impact on my research work. Also, I thank all of them for helping me by giving comments on my write ups whenever needed. I also thank all the past and present members of the Algorithms Reading Group for organizing meetings and giving talks from which I have gained ample knowledge.

I am indebted to Prof. Subhash Suri of University of California, Santa Barbara for trusting me and for giving me the opportunity of doing a research internship with him. I had spent a great time in his lab doing exciting research and learning new things. I am thankful to my UCSB colleague Neeraj for being so nice and friendly to me, and also for the brainstorming discussion sessions. I also thank my other collaborators Aritra, Aniket, Saeed, Anil and Sujoy for their time and effort.

Next, I thank the University of Iowa authorities for providing such a vibrant environment for doing research. I am thankful to all the faculty members in or out of our department with whom I had the opportunity of taking courses. These courses
had helped me jump-start my research. I also thank our department secretaries Sheryl, Catherine and Matthieu for their help and suggestions whenever needed.

Words cannot completely express my love and gratitude to my family, who have supported and encouraged me through this journey. I would like to thank my parents for their life-long support, and sacrifices, which sustained my interest in research and motivated me towards the successful completion of this study. My heartiest thanks to my sisters and brothers-in-law, who have supported me through my difficult times with their inspiration and continued encouragement. I also thank my girlfriend Manaswita for her constant support and care, and for always believing in my capability.

I will always remember the true friends who have walked with me on my path. There are too many to mention by name, yet, too few to ever lose room for in my heart. Many thanks to my dear friends, especially of Sriram, Satya, Sairam, Dan, Srijit, Indrani, Sanchita, Subin, Avijit, Santanu and Arani for times of laughter when I needed a reprieve from the intensity of work.
ABSTRACT

Clustering problems often arise in the fields like data mining, machine learning and computational biology to group a collection of objects into similar groups with respect to a similarity measure. For example, clustering can be used to group genes with related expression patterns. Covering problems are another important class of problems, where the task is to select a subset of objects from a larger set, such that the objects in the subset “cover” (or contain) a given set of elements. Covering problems have found applications in various fields including wireless and sensor networks, VLSI, and image processing. For example, covering can be used to find placement locations of the minimum number of mobile towers to serve all the customers of a region. In this dissertation, we consider an interesting collection of geometric clustering and covering problems, which are modeled as optimization problems. These problems are known to be NP-hard, i.e. no efficient algorithms are expected to be found for these problems that return optimal solutions. Thus, we focus our effort in designing efficient approximation algorithms for these problems that yield near-optimal solutions. In this work, we study three clustering problems: $k$-means, $k$-clustering and Non-Uniform-$k$-center and one covering problem: Metric Capacitated Covering.

$k$-means is one of the most studied clustering problems and probably the most frequently used clustering problem in practical applications. In this problem, we are given a set of points in an Euclidean space and we want to choose $k$ center points from the same Euclidean space. Each input point is assigned to its nearest chosen center,
and points assigned to a center form a cluster. The cost per input point is the square of its distance from its nearest center. The total cost is the sum of the costs of the points. The goal is to choose \(k\) center points so that the total cost is minimized. We give a local search based algorithm for this problem that always returns a solution of cost within \((1 + \varepsilon)\)-factor of the optimal cost for any \(\varepsilon > 0\). However, our algorithm uses \((1 + \varepsilon)k\) center points. The best known approximation before our work was about 9 that uses exactly \(k\) centers. The result appears in Chapter 2.

\(k\)-clustering is another popular clustering problem studied mainly by the theory community. In this problem, each cluster is represented by a ball in the input metric space. We would like to choose \(k\) balls whose union contains all the input points. The cost of each ball is its radius to the power \(\alpha\) for some given parameter \(\alpha \geq 1\). The total cost is the sum of the costs of the chosen \(k\) balls. The goal is to find \(k\) balls such that the total cost is minimized. We give a probabilistic metric partitioning based algorithm for this problem that always returns a solution of cost within \((1 + \varepsilon)\)-factor of the optimal cost for any \(\varepsilon > 0\). However, our algorithm uses \((1 + \varepsilon)k\) balls, and the running time is quasi-polynomial. The best known approximation in polynomial time is \(c^\alpha\) that uses exactly \(k\) balls, where \(c\) is a constant. The result appears in Chapter 3.

Non-Uniform-\(k\)-center is another clustering problem, which was posed very recently. Like in \(k\)-clustering here also each cluster is represented by a ball. Additionally, we are given \(k\) integers \(r_1, \ldots, r_k\), and we want to find the minimum dilation \(\alpha\) and choose \(k\) balls with radius \(\alpha \cdot r_i\) for \(1 \leq i \leq k\) whose union contains all the input
points. This problem is known to be notoriously hard. No approximation is known even in the special case when \( r_i \)'s belong to a set of three integers. We give an LP rounding based algorithm for this special case that always returns a solution of cost within a constant factor of the optimal cost. However, our algorithm uses \((2 + \varepsilon)k\) balls for some constant \(\varepsilon\). We also show that this special case can be solved in polynomial time under a practical assumption. Moreover, we prove that the Euclidean version of the problem is also as hard as the general version. These results appear in Chapter 4.

Capacitated Covering is a generalization of the classical set cover problem. In the Metric Capacitated Covering problem, we are given a set of balls and a set of points in a metric space. Additionally, we are given an integer that is referred to as the capacity. The goal is to find a minimum subset of the input set of balls, such that each point can be assigned to the chosen balls in a manner so that the number of points assigned to each ball is bounded by the capacity. We give an LP rounding based algorithm for this problem that always returns a solution of cost within a constant factor of the optimal cost. However, we assume that we are allowed to expand the balls by a fairly small constant. If no expansion is allowed, then the problem is known to not admit any constant approximation. We discuss our findings in Chapter 5.

As mentioned above, for many of the problems we consider, we obtain results that improve the best known approximation bounds. Our findings make significant progress towards better understanding the internals of these problems, which have impact across the disciplines. Also, during the course of our work, we have designed
tools and techniques, which might be of independent interest for solving similar optimization problems. Finally, in Chapter 6, we conclude our discussion and pose some open questions, which we consider as our potential future work.
Consider the following two fictitious scenarios. (i) Suppose an organization wants to start a web news portal that will list all the online news articles of the day classified by their categories. However, the articles are assumed to have no category information, and thus here we want to guess the categories of the articles. (ii) Suppose a company has obtained the tender for placing mobile towers in a new city. In the initial plan, it has identified a large set of placement locations such that the towers can serve all the clients in the city. We want to select a “small” number of towers from the plan that still can serve all the clients in the city. We note that the two above mentioned problems belong to the class of clustering problems and the class of covering problems, respectively. Clustering is the task of grouping a set of items into similar groups (or clusters) with respect to a similarity measure. Ideally, items in different groups should have low similarity. Often the items are represented by points in a high dimensional space. The distance between two items represents their similarity. On the other hand, covering is the task of selecting a subset of objects from a larger set, such that the objects in the subset “cover” (or contain) a given set of elements. Ideally, one wants to find such a subset of the minimum size. Usually, the objects and the elements are modeled using geometric shapes and points, respectively. We have considered several clustering and covering problems in this thesis, and have designed improved algorithms for solving them. Our main contribution is to prove theoretical guarantees on the quality of the solutions produced by our algorithms.
TABLE OF CONTENTS

LIST OF TABLES . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . xiii

LIST OF FIGURES . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . xiv

CHAPTER

1 INTRODUCTION . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1

1.1 \(k\)-means Clustering . . . . . . . . . . . . . . . . . . . . . . . 9
  1.1.1 Previous Work . . . . . . . . . . . . . . . . . . . . . . . . 10
  1.1.2 Our Results and Contributions . . . . . . . . . . . . . . . 12

1.2 \(k\)-clustering . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
  1.2.1 Previous Work . . . . . . . . . . . . . . . . . . . . . . . . 15
  1.2.2 Our Results and Contributions . . . . . . . . . . . . . . . 16

1.3 Non-Uniform \(k\)-center . . . . . . . . . . . . . . . . . . . . . . . 19
  1.3.1 Previous Work . . . . . . . . . . . . . . . . . . . . . . . . 22
  1.3.2 Our Results and Contributions . . . . . . . . . . . . . . . 24

1.4 Capacitated Covering . . . . . . . . . . . . . . . . . . . . . . . . 26
  1.4.1 Previous Work . . . . . . . . . . . . . . . . . . . . . . . . 27
  1.4.2 Our Results and Contributions . . . . . . . . . . . . . . . 29

2 \(K\)-MEANS CLUSTERING . . . . . . . . . . . . . . . . . . . . . . . . 33

2.1 Preliminaries . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33

2.2 Sum of Squares Facility Location Problem . . . . . . . . . . . . . 36
  2.2.1 The Local Search Algorithm . . . . . . . . . . . . . . . . . 37
  2.2.2 Analysis of the Local Search Algorithm . . . . . . . . . . . 38
  2.2.3 Separator Theorem . . . . . . . . . . . . . . . . . . . . . . 39
  2.2.4 The Partitioning Algorithm . . . . . . . . . . . . . . . . . . 40
  2.2.5 Approximation Bound . . . . . . . . . . . . . . . . . . . . . 45

2.3 Local Search Algorithm for \(k\)-means . . . . . . . . . . . . . . . . 48
  2.3.1 The Local Search Algorithm . . . . . . . . . . . . . . . . . 48
  2.3.2 Analysis of the Local Search Algorithm . . . . . . . . . . . 49
  2.3.3 Proof of Lemma 11 . . . . . . . . . . . . . . . . . . . . . . . 52

3 \(K\)-CLUSTERING . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57

3.1 Preliminaries . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57

3.2 A QPTAS for Minimum Cost Covering Problem . . . . . . . . . . . 58
  3.2.1 The Partitioning Scheme . . . . . . . . . . . . . . . . . . . . 59
6.3 Capacitated Covering .............................................. 157
6.4 Non-Uniform-$k$-center ........................................... 159

REFERENCES ............................................................. 161
# LIST OF TABLES

Table

1.1 Approximation algorithms for $k$-means ........................................... 11
1.2 Approximation algorithms for $k$-clustering ........................................ 16
6.1 $k$-clustering: known results and open questions .................................... 157
6.2 ECC: known results and open questions ................................................. 158
6.3 NUkC: known results and open questions ............................................... 160
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Visualization of three clusters: points in the same cluster are represented by the same shape and color.</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>A solution of 2-means where the cluster centers are the centroids.</td>
<td>4</td>
</tr>
<tr>
<td>1.3</td>
<td>A solution of 3-clustering. A point is arbitrarily assigned to a ball, which contains it.</td>
<td>5</td>
</tr>
<tr>
<td>1.4</td>
<td>(a) The dataset. (b) $k$-center clustering. (c) Non-Uniform $k$-center clustering.</td>
<td>7</td>
</tr>
<tr>
<td>1.5</td>
<td>The optimal clusters are contained in the two disks on the Euclidean plane. The centers of balls are shown by boxes. The Voronoi partition w.r.t. the centers contains two subsets of points lying on the different sides of the vertical bisector line.</td>
<td>21</td>
</tr>
<tr>
<td>1.6</td>
<td>Examples demonstrating the definition of perturbation resilience. The top-right (resp. bottom-right) instance is a 2-perturbed instance of the top-left (resp. bottom-left) instance. The points in same optimal cluster are shown by same shape and color.</td>
<td>21</td>
</tr>
<tr>
<td>4.1</td>
<td>Example showing a feasible fractional solution. But, no feasible integral solution exists. The $y$-value of each node is shown either explicitly or by the value on the line above it.</td>
<td>90</td>
</tr>
<tr>
<td>4.2</td>
<td>An example tree showing a feasible fractional solution.</td>
<td>92</td>
</tr>
<tr>
<td>4.3</td>
<td>An example tree showing a feasible fractional solution.</td>
<td>94</td>
</tr>
<tr>
<td>5.1</td>
<td>The element $x$ belongs to three triples, whose gadgets are attached to the element gadget of $x$ at three locations. The triple gadgets are also attached to other element gadgets. Note that each ball contains 4 points, but has the capacity of only 3.</td>
<td>145</td>
</tr>
</tbody>
</table>
CHAPTER 1
INTRODUCTION

In this dissertation, we study several geometric clustering and covering problems. Clustering is the task of grouping a set of elements into a number of groups, such that the items in the same group have high similarity and items in different groups have low similarity. Each such group is referred to as a cluster. On the other hand, covering refers to the task of selecting a subfamily of sets from a relatively larger family of sets, such that the union of the selected sets contain all elements of a pre-specified ground set. We say that such a subfamily covers the ground set. Clustering problems typically arise in the fields such as data mining, machine learning, pattern recognition, image processing, biology and so on [29, 46, 49, 51, 52, 76, 77, 83, 93, 111]. Similarly, covering problems have applications in various fields including wireless and sensor networks, VLSI, and image processing [2, 24, 32, 34, 65, 72, 74].

In a typical clustering problem, the items are represented by points in a metric space, i.e., in a space of points where the distance between the points satisfy the metric property. For example, one can consider the problem in $\mathbb{R}^d$ w.r.t. the Euclidean distance where each dimension of the space $\mathbb{R}^d$ represents a feature of the items (see Figure 1.1). The distance function is used to capture the similarity of the items. The lesser the distance between two points, the more similar they are. Typically, each cluster is represented by a point or by a ball (in the metric space) that contains all points of the cluster. The representative point is called the cluster center. Then, if one
can find all the representatives of the clusters, the clusters can also be identified. To retrieve good quality clusters, the problem is modeled as an optimization problem, where one needs to minimize (or maximize) certain objective function. Then, we expect to find the natural clusters if we can compute a clustering that optimizes the objective function. Hence, the objective function is just a proxy for finding the natural groups of the data. We note that, for different applications, one might need to use different objective functions. The objective function usually depends on the distances between the points and the centers or on the radii of the balls. For example, in $k$-center clustering [61] the goal is to find $k$ balls, such that the union of the balls contains all the points and the radius of the maximum radius ball is minimized.

In a typical covering problem, the goal is to cover a set of entities. One can think of these entities as clients, which require a particular service. The service is to be provided by some number of servers. Note that, for cost-effectiveness, one might
want to select these servers judiciously. Typically, the planner comes up with an initial set of servers. Each such server has a set of clients that it can potentially serve. Also there is an estimated cost associated with each such server that one needs to incur if that server is finally selected for providing service. However, in this initial plan there might be some redundant servers in the sense that even if these servers are not finally selected, all the clients can get service from the remaining servers. Thus, one would like to select a set of servers that minimizes the total cost. We note that the basic covering problem can be modeled in several ways based on different aspects of the concerned application. For example, the cost corresponding to the servers might be same or different. Also each server might have a bound on the number of clients that it can serve. Moreover, in some applications a client might want to get service from multiple servers instead of a single one. In the geometric version of the problem, the entities are represented by points in a space, and each server is represented by a geometric object. The points lying inside an object are the ones that the object can serve or cover. Due to their applicability to a wide range of real life problems, the geometric version of covering has gained the maximum attention from researchers.

In this dissertation, we consider several interesting clustering and covering problems. In particular, we study three areas pertaining to clustering: \textit{k-means}, \textit{k-clustering} and Non-Uniform \textit{k}-center clustering and one area pertaining to covering: capacitated set cover. We start with \textit{k-means}, which is one of the most studied clustering problems in the literature. In \textit{k}-means clustering, given a set \( P \) of \( n \) points in \( \mathbb{R}^d \) and an integer \( k > 0 \), the goal is to find a set \( K \) of \( k \) centers in \( \mathbb{R}^d \), such that
Figure 1.2. A solution of 2-means where the cluster centers are the centroids.

the quantity

$$\text{cost}(K) = \sum_{p \in P} \min_{q \in K} ||p - q||^2$$

is minimized, where $||p - q||$ denotes the Euclidean distance between $p$ and $q$. Once
the centers are selected, each input point is assigned to its nearest center. For each
center, all the points assigned to it form a corresponding cluster. One can show that
for any set of points $P'$, the geometric mean of $P'$ minimizes the following function
in $\mathbb{R}^d$

$$\lambda(x) = \sum_{p \in P'} ||p - x||^2$$

Hence, in $k$-means, for each cluster of points, it is beneficial to choose the geometric
mean (or centroid) of the points as the center (see Figure 1.2). As one can compute
the geometric mean of a set of points efficiently, given a cluster of points, the cluster
center can also be computed efficiently. We note that one can define the problem w.r.t.
any metric. However, the above mentioned property of the cluster centers might not
hold for arbitrary metric. We describe a “bi-criteria” approximation algorithm for $k$-
means, which uses \((1+\varepsilon)k\) centers and yields a solution whose cost is at most \((1+\varepsilon)\) times the cost of an optimal \(k\)-means solution. The algorithm runs in polynomial time for any fixed dimension.

Next, we consider the \(k\)-clustering problem. In this problem, we are given a set \(X\) of \(n\) points, and a metric \(d\) on \(X\). For \(z \in X\) and \(r \geq 0\), the ball \(B(z,r)\) centered at \(z\) and having radius \(r\) is the set \(\{y \in X | d(z,y) \leq r\}\). A cover for a subset \(Q \subseteq X\) is a set of balls, each centered at a point of \(X\), whose union contains \(Q\). The cost of a set \(B = \{B_1, \ldots, B_t\}\) of balls, denoted by \(\text{cost}(B)\), is \(\sum_{i=1}^{t} r(B_i)^{\alpha}\), where \(r(B_i)\) is the radius of \(B_i\), and \(\alpha \geq 1\) is a parameter of the problem (but not of a problem instance). The \(k\)-clustering problem is to find a set \(B\) of \(k\) balls that is a cover for \(X\) such that \(\text{cost}(B)\) is minimized. Note that here each cluster is defined by a ball and a point can be assigned to a ball that contains it. This assignment gives a partition of \(X\) into \(k\) clusters (see Figure 1.3). Clustering problems where the goal is

Figure 1.3. A solution of 3-clustering. A point is arbitrarily assigned to a ball, which contains it.
to minimize the maximum radius (or diameter) of the clusters have been well studied [6, 53, 54, 73, 61, 98]. But in many applications, clustering algorithms that try to minimize the maximum radius of the clusters suffer from a crucial problem called the dissection effect [37, 48, 66, 101]. Due to this effect similar items having relatively large distances from each other are assigned to different clusters, as otherwise the maximum radius of the clusters would be too large. This undesirable effect was the main motivation behind studying the $k$-clustering objective function. As in $k$-clustering the cost depends on all balls rather than just the maximum radius ball, it is less prone to the dissection effect [66, 101]. Thus, for all those applications, $k$-clustering objective function is more suitable. We describe an algorithm for $k$-clustering that uses at most $(1 + \varepsilon)k$ balls and yields a solution whose cost is at most $(1 + \varepsilon)$ times the cost of an optimal $k$-clustering solution. The algorithm runs in quasi-polynomial time, i.e., the running time is of the form $n^{O((\log n)^c)}$ for input size $n$, where $c$ is a constant.

Thereafter, the Non-Uniform $k$-center (NUkC) problem is considered. In NUkC, we are given a set of $n$ points $P$ in a metric space, non-negative integers $r_1, r_2, \ldots, r_k$, and the goal is to find the minimum dilation $\alpha$ and to choose $k$ balls centered at the points of $P$ with radius $\alpha \cdot r_i$ for $1 \leq i \leq k$, such that all points of $P$ are contained in the union of the chosen balls. The problem was formulated by Chakrabarty et al. [33] as a generalization of the well-studied $k$-center clustering problem where all $r_i$'s are the same. Due to the discrepancy in the radii of the balls, NUkC might be able to find hidden patterns in the data, which would not be possible
for $k$-center (see Figure 1.4). Hence, for many applications NUkC clustering might be more suitable compared to $k$-center clustering. Apart from clustering, NUkC has several applications in vehicle routing, sensor placement, and so on. For example, in vehicle routing, we need to find $k$ depot locations corresponding to $k$ vehicles having different speeds, such that any customer can be served by some vehicle as quickly as possible. We design a “bi-criteria” algorithm for a special case of the problem that achieves a constant approximation. We also obtain an inapproximability result for the Euclidean version of the problem. Moreover, we consider the problem under a practical assumption on the inputs. We show that, under this assumption, a version of NUkC can be solved exactly in polynomial time.

Lastly, we consider the Metric Capacitated Covering problem. In this problem, we are given a set $P$ of $n$ points in a metric space, a set $\mathcal{B}$ of balls from the same metric space, and a positive integer $U$ that represents the capacity of each of the balls in $\mathcal{B}$. We would like to compute a subset $\mathcal{B}' \subseteq \mathcal{B}$ of balls and assign each
point in $P$ to some ball in $B'$ that contains it, such that the number of points assigned to any ball is at most $U$. The objective function that we would like to minimize is the cardinality of $B'$. We refer to the problem as the \textit{Euclidean Capacitated Covering} problem when the underlying metric is Euclidean. Note that it is natural to consider capacity constraints that appear in many applications, for instance, an upper bound on the number of clients that can be served by an antenna. Such constraints lead to the natural formulation of Metric Capacitated Covering. We present a “bi-criteria” $(O(1), 6.47)$-approximation algorithm for the Metric Capacitated Covering problem, i.e., the algorithm returns a solution in which the balls may be expanded by a factor of 6.47 (for any ball $B_i$, and any point $p_j \in P$ that is assigned to $B_i$, $d(c_i, p_j) \leq 6.47 \cdot r_i$), and the cost of the solution is at most $O(1)$ times that of an optimal solution (which does not expand the balls). We also present an $(O(\epsilon^{-4d} \log(1/\epsilon)), 1+\epsilon)$-approximation for the Euclidean Capacitated Covering problem in $\mathbb{R}^d$ for any $\epsilon > 0$ and fixed dimension $d$. Thus, assuming we are allowed to expand the input balls by an arbitrarily small constant factor, we can obtain a solution with at most a corresponding constant times the number of balls used by any optimal solution. We note that it is a common practice in the wireless network setting to expand the radii of antennas at the planning stage to improve the quality of service. For example, Bose \textit{et al.} [28] proposed a scheme for replacing omni-directional antennas by directional antennas that expands the antennas by a constant factor. To complement our algorithmic results, we also describe two hardness of approximation results that give a better understanding of the problems we consider.
1.1 \( k \)-means Clustering

Among the great variety of clustering problems, specifically \( k \)-means has got much attention from the researchers in the community. The problem is known to be \( \text{NP} \)-hard even in the plane [95] and no inapproximability result is known for this planar version of the problem. On the other hand, the best known polynomial time approximation factor for the metric version of \( k \)-means is close to 9 [81]. No better approximation is known even in the plane, despite the fact that \( k \)-means is a well studied problem. An insight about the difficulty of \( k \)-means can be found by comparing it to the \( k \)-median clustering. \( k \)-median is similar to \( k \)-means except the goal is to minimize the sum of distances instead of the sum of squares of distances. We note that, in a real line, for any finite set \( P' \) of points, the median of the points minimizes the following function

\[
\beta(x) = \sum_{p \in P'} ||p - x||
\]

Hence, the name \( k \)-median is justified when we are interested in choosing \( k \) centers in \( \mathbb{R}^d \) that minimizes the sum of distances cost function. Like \( k \)-means, \( k \)-median is also an \( \text{NP} \)-hard problem. Arora \textit{et al.} [13] presented a \((1 + \varepsilon)\)-factor approximation algorithm for \( k \)-median in \( \mathbb{R}^d \) that runs in polynomial time. The algorithm is based on a novel technique due to Arora [12]. This result for \( k \)-median might lead to the conclusion that a reason behind the difficulty of \( k \)-means is its objective function. One reason that squares of distances are harder to handle compared to the distances is they do not follow triangle inequality in general. However, the important question in this context is, “Is the objective function of \( k \)-means by itself good enough to make
this problem harder?"

Motivated by the question mentioned in the previous paragraph we also consider the following problem, which will be referred to as the *Sum of Squares Facility Location* (SOS-FL) problem. Given a set $C$ of points (clients) in $\mathbb{R}^d$ and a real $f > 0$, find a finite set $F$ of points (facilities) in $\mathbb{R}^d$, such that the quantity

$$cost(F) = f \cdot |F| + \sum_{p \in C} \min_{q \in F} ||p - q||^2$$

is minimized. Note that SOS-FL is similar to $k$-means except the global constraint on the number of facilities (or centers) is absent here. In the following, we mention some selected work from the literature of $k$-means.

### 1.1.1 Previous Work

The most common heuristic for $k$-means is an algorithm due to Lloyd [92], which is based on an iterative refinement technique. Though he proposed this algorithm in 1957, he did not publish it until 1982. For historical reasons this algorithm is referred to as the $k$-means algorithm. Though this algorithm was proposed 50 years ago, it remains one of the most popular algorithms for clustering even today. The reasons behind its overwhelming popularity are its simplicity, ease of implementation, efficiency, and empirical success. As mentioned before, the $k$-means problem is known to be NP-hard. Thus, as expected Lloyd’s algorithm converges to a local optimal, whose quality might not be “good” compared to the global optimal. However, a recent study has shown that with high probability the algorithm can reach to the global optimal when clusters are “well separated” [99]. Considering the fact that Lloyd’s algorithm is very successful in practice, researchers have attempted to study
it from theoretical perspective as well. There are many variants of this basic algorithm that ensure theoretical guarantee on the quality of the solution. For example, Arthur and Vassilvitski [14] used a simple trick to modify the basic algorithm and have achieved a $\Theta(\log k)$ approximation.

<table>
<thead>
<tr>
<th>Approx. ratio</th>
<th>Time complexity</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$O(n^{kd+1})$</td>
<td>Inaba et al. [75]</td>
</tr>
<tr>
<td>1+\varepsilon</td>
<td>$O(2^{(k/\varepsilon)}d n)$</td>
<td>Kumar et al. [88]</td>
</tr>
<tr>
<td>1+\varepsilon</td>
<td>$O(n^{2-k^2d \log k} n)$</td>
<td>Matousek [97]</td>
</tr>
<tr>
<td>1+\varepsilon</td>
<td>$O(n + \exp(k + 1))$</td>
<td>Har-Peled and Mazumdar [71]</td>
</tr>
<tr>
<td>1+\varepsilon</td>
<td>$\alpha(n, k) + \exp(1/\varepsilon, d)$</td>
<td>Har-Peled and Kushal [69]</td>
</tr>
<tr>
<td>9+\varepsilon</td>
<td>$\text{poly}(n, k)$</td>
<td>Kanungo et al. [81]</td>
</tr>
<tr>
<td>1 + $\varepsilon$</td>
<td>$\text{poly}(n, k, d)$</td>
<td>Makarychev et al. [96]</td>
</tr>
</tbody>
</table>

$\text{poly}(n, k, d)$ (with $\beta k$ centers)

Table 1.1. Approximation algorithms for $k$-means

There are many $(1 + \varepsilon)$-factor approximation algorithms for $k$-means, whose time complexity depend linearly on $n$ [69, 71, 88, 97] (see Table 1.1). Unfortunately, the time complexity of these algorithms depend exponentially on $k$ and hence they are not suitable in practice when $k$ is sufficiently large. For arbitrary $k$ and $d$, there is a polynomial time approximation algorithm that achieves an approximation ratio $9 + \varepsilon$, which is currently the best known bound [81]. On the other hand, Makarychev et al. [96] have designed three “bi-criteria” approximation algorithms for $k$-means that use at most $\beta k$ ($> k$) centers and achieve an approximation factor $\alpha(\beta)$
(< 9 + ε) that depends on β. Moreover, α(β) decreases rapidly with β (for instance α(2) < 2.59, α(3) < 1.4). These algorithms have only polynomial dependence on the dimension of input points. Recently, Awasthi et al. [16] have attempted to study the inapproximability of k-means. They have shown that there exists a constant ε > 0, such that (1 + ε)-approximation is not achievable in polynomial time for k-means in sufficiently high (Ω(log n)) dimensions, assuming some complexity theoretic conjectures to be true.

1.1.2 Our Results and Contributions

The main contribution of our work is that we are being able to handle the square of distances in an elegant way, which yields near optimal approximation bounds. This is in particular interesting, as it gives a better understanding of the classical k-means problem, whose status in the plane has remained open for a long time. We design polynomial time approximation algorithms based on local search technique for both SOS-FL and k-means.

- For any ε > 0, we present a polynomial time algorithm for SOS-FL that yields a (1 + ε)-approximation.

- For any ε > 0, we present a polynomial time algorithm for k-means – the algorithm uses at most (1 + ε)k centers and yields a solution whose cost is at most (1 + ε) times the cost of an optimal k-means solution.

The algorithm and the analysis for both of the problems are similar. However, in case of k-means there are more subtleties, which arise due to the limitation on the
number of centers. In general, both of the algorithms are based on a local search method that allows swapping in and out of constant number of facilities or centers. Like the approaches in [25, 36, 43, 103], we also use separators to prove the quality of the approximation. To be precise we use the separator from [25], which is most suitable for our purpose. We note that this separator itself gives a lot of ease in handling the square of distances. The separator is used repeatedly to partition the local and global optimal facilities simultaneously into constant size “parts”. The rest of the analysis involves assignment of clients corresponding to the local facilities of each “part” to the global facilities corresponding to that “part” only or to some “auxiliary” points. Also one should be careful that a client should not be assigned to a point “far” away from it compared to its nearest local and global facility. The choice of the separator plays a crucial role to give a bound on this cost. From a high level our approach is similar to the approach of Cohen-Addad and Mathieu [43] for clustering problems. But the details of the analysis are significantly different in places. For example, they use the dissection technique from [87] as their separator and thus the assignment in their case is completely different and more complicated than ours. In this regard, we would like to mention that the dissection technique from [87] or the quadtree based approach of Arora [12] are not flexible enough to handle the square of distances. The local search algorithms for SOS-FL and \(k\)-means are described in Chapter 2.

**Remark.** Following our work, Cohen-Addad et al. [42] and Friggstad et al. [55] have independently obtained PTASes for \(k\)-means.
1.2 $k$-clustering

$k$-clustering is known to be NP-hard in general metric for any $\alpha \geq 1$ and in the Euclidean plane for $\alpha > 1$ [8, 59], and no inapproximability result is known even for the general $k$-clustering. On the other hand, the best known approximation guarantee is $c^\alpha$ even in the Euclidean plane for some constant $c$. To get a better understanding of $k$-clustering we define a facility location version of this problem, like in the case of $k$-means. We are given two point sets $X$ and $Y$, and a metric on $Z = X \cup Y$. The goal is to select a set $B$ of balls, such that the center of each ball belongs to $Y$, $B$ is a cover for $X$, and $\text{cost}(B) = \sum_{B \in B} r(B)^\alpha$ is minimized. We refer to this problem as the Minimum Cost Covering problem. Note that Minimum Cost Covering is similar to $k$-clustering except in Minimum Cost Covering the balls are centered at points of $Y$ and Minimum Cost Covering does not have any global constraint associated with it.

The Minimum Cost Covering problem models the famous base station placement problem [89]. One can consider the points of $X$ as clients and the points of $Y$ as the potential locations for base stations (or servers). A base station located at a point $y$ of $Y$ having transmission range $r$ can serve any clients in the ball $B(y, r)$. The cost of a collection of servers is the sum of the $\alpha^{th}$ power of the transmission ranges of the servers. The goal is to select a subset of locations from $Y$ to place servers and assign transmission range to each server, so that all the clients are being served and the total cost of the servers is minimized. For modeling the energy needed for wireless transmission, it is common to consider the value of $\alpha$ to be greater than 1.
The idea is that the cost of placing a server is defined as the energy required for wireless transmission. Hence, it is common to assume a super-linear dependence on the radius. However, the case of $\alpha = 1$ is also interesting for several reasons. Firstly, the results for the case $\alpha > 1$ can be built upon the results for this simpler case. Secondly, there are many settings where a linear dependence is more appropriate [41, 106]. The special case of Minimum Cost Covering in the plane where the transmission radius of each server is a fixed constant has also been studied, which is called the fixed radius disk-covering problem [89].

As one might think, Minimum Cost Covering is relatively “simpler” than $k$-clustering. Thus, all the algorithms for $k$-clustering can be modified accordingly to make them work for Minimum Cost Covering. In addition, for any fixed $d$, there is a PTAS for Minimum Cost Covering in $\mathbb{R}^d$ [26]. In the following, we mention some selected work from the literature of $k$-clustering.

\subsection{Previous Work}

At first we discuss the literature of $k$-clustering for the case $\alpha = 1$ (see Table 1.2). The algorithm due to Lev-Tov and Peleg in [89] can be modified to get an algorithm for $k$-clustering in the Euclidean plane that uses at most $(1 + \varepsilon)k$ balls and yields a solution whose cost is at most $(1 + \varepsilon)$ times the cost of an optimal $k$-clustering solution. Gibson et. al [60] designed a polynomial time exact algorithm for this problem in $\mathbb{R}^d$ when the underlying distance function is either the $l_1$ or $l_\infty$ metric. For the $l_2$ metric, they also get an exact algorithm if one assumes two candidate solutions can be compared efficiently; without this assumption, they get
a \((1 + \varepsilon)\)-approximation. In a different work, the authors have extended the exact algorithm to arbitrary metric spaces [59]. The running time is quasi-polynomial if the aspect ratio of the metric (ratio of maximum to minimum interpoint distance) is bounded by a polynomial in the number of points. When \(\alpha > 1\), the structure that holds for \(\alpha = 1\) breaks down (see the last row of Table 1.2). In this case, there is an algorithm due to Charikar and Panigrahy [37] that leads to an approximation guarantee of \(c^\alpha\) for some constant \(c\). Also there is a polynomial time approximation algorithm for the problem in \(\mathbb{R}^d\) that uses at most \((1 + \varepsilon)k\) balls and yields a solution whose cost is at most \((1 + \varepsilon)\) times the cost of an optimal \(k\)-clustering solution.

<table>
<thead>
<tr>
<th>(\alpha)</th>
<th>Metric space</th>
<th>Approx. ratio</th>
<th>Time complexity</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(\mathbb{R}^2)</td>
<td>(1 + \varepsilon) (with ((1 + \varepsilon)k) balls)</td>
<td>(\text{poly}(n, k))</td>
<td>Lev-Tov and Peleg [89]</td>
</tr>
<tr>
<td>1</td>
<td>(\mathbb{R}^d (l_1/l_\infty))</td>
<td>1</td>
<td>(\text{poly}(n, k))</td>
<td>Gibson et al [60]</td>
</tr>
<tr>
<td>1</td>
<td>(\mathbb{R}^d (l_2))</td>
<td>(1 + \varepsilon)</td>
<td>(\text{quasi-poly}(n) + \text{poly}(k))</td>
<td>Gibson et al [59]</td>
</tr>
<tr>
<td>&gt; 1</td>
<td>General</td>
<td>((O(1))^{\alpha})</td>
<td>(\text{poly}(n, k))</td>
<td>Charikar and Panigrahy [37]</td>
</tr>
</tbody>
</table>

Table 1.2. Approximation algorithms for \(k\)-clustering

### 1.2.2 Our Results and Contributions

We consider metric Minimum Cost Covering and \(k\)-clustering with \(\alpha \geq 1\).

- For any \(\varepsilon > 0\), we design a \((1 + \varepsilon)\)-factor approximation algorithm for Minimum
Cost Covering that runs in quasi-polynomial time, that is, in $2^{\left(\log^{mn/\epsilon}c\right)}$ time, where $c > 0$ is a constant, $m = |Y|$, and $n = |X|$.

- For any $\epsilon > 0$, we design a quasi-polynomial time algorithm for $k$-clustering that uses at most $(1 + \epsilon)k$ balls and yields a solution whose cost is at most $(1 + \epsilon)$ times the cost of an optimal $k$-clustering solution.

The Minimum Cost Covering is thus an interesting example of a metric covering/clustering problem that admits a $(1 + \epsilon)$-approximation, if one is willing to settle for quasi-polynomial time. It is also an example where the techniques used in fixed dimensional Euclidean spaces generalize nicely to metric spaces. This is in contrast to the facility location problem [13]. The result for Minimum Cost Covering should be compared with the polynomial time algorithm [37] that guarantees a $3^\alpha$ approximation.

One key property of the Minimum Cost Covering problem is that in an optimal cover, there are only a small number of balls whose radius is large. We can therefore afford to guess these balls by an explicit enumeration. However, there can be a large number of balls with small radius. To help 'find' these, we partition the metric space into blocks with at most half the original diameter, and recurse on each block. We have to pay a price for this recursion in the approximation guarantee. This price depends on the number of blocks in the partition that a small radius ball can intersect.

We are led to the following problem: is there a way to probabilistically partition a metric space into blocks of at most half the diameter, so that for any ball of radius $r$, the expected number of blocks that intersect the ball can be nicely bounded?
The celebrated partitioning algorithms of Bartal [21] and Fakcharoenphol, Rao, and Talwar [50] guarantee that the probability that such a ball is intersected by two or more blocks is nicely bounded. We would like to bound the expected number of blocks intersected by such a ball. To our knowledge, our work is the first one that highlights and studies the expected number of blocks.

If one employs the partitioning algorithm of [50], the expected number of blocks intersected by such a ball can be quite large, as we explain in Section 3.2.1. However, we argue that using a different partitioning algorithm, the expected number of blocks intersected by a small radius ball is suitably small. Our partitioning algorithm resembles the earlier algorithm of Bartal [21]. Furthermore, our analysis of the expected number of blocks intersected has elements that parallel elements from other works [22, 1, 79] that use probabilistic partitions in different contexts. Our main contribution in Section 3.2.1 is in employing probabilistic partitioning knowhow to study a new quantity, the expected number of blocks, that is of interest to us.

The algorithms for both Minimum Cost Covering and $k$-clustering use the partitions in Section 3.2.1. We also consider the inapproximability of a variant of Minimum Cost Covering where we allow $\alpha$ to be part of the input. For $\alpha \ge \log |X|$, we show, under standard complexity theoretic assumptions, that no polynomial (or quasi-polynomial) time algorithm for Minimum Cost Covering can achieve an approximation factor better than $O(\log |X|)$. The result is obtained using a reduction from the dominating set problem to Minimum Cost Covering. This partly explains the dependence on $\alpha$ of the running time of our algorithms. All these results appear in
Chapter 3.

1.3 Non-Uniform $k$-center

As mentioned before, $k$-center is a special case of NUkC, where all the input radii are equal. We call this version of the problem as NUkC with one radius class. In general, all the radii might not be equal. But, we can consider only distinct radii from the input and associate a multiplicity parameter $k_i$, with each such radius $r_i$, which denotes the number of balls of radius $r_i$ that can be opened. Then, the problem can be formulated in the following manner. The input includes $t \leq k$ distinct radii $r_1 > r_2 > \ldots > r_t$ and integers $k_1, \ldots, k_t$ whose sum equals $k$. The goal is to place $k_i$ balls of radius $r_i$ for each $1 \leq i \leq t$ and to expand the balls by the minimum factor such that the union of those balls contains all the input points. We will refer to this version as NUkC with $t$ radii classes. We refer to any feasible solution of this problem composed of the chosen balls as a feasible placement. We note that, as in $k$-clustering, in NUkC also the clusters are represented by balls. However, in case of $k$-clustering there is no hard bound on the number of balls of any particular radius that can be chosen in the solution. Intuitively, this makes the NUkC problem much more difficult compared to $k$-clustering.

We also consider a “stable” version of the NUkC problem. Stability is a fairly popular notion, which has been used in the literature in the context of beyond worst case analysis. The general idea is to impose extra constraints on the inputs such that the (stable) instances that satisfy those constraints can capture the instances that appear in real life applications. In other words, we would like to exclude the “unre-
alistic” instances from consideration and obtain optimistic bounds for algorithms on the remaining inputs. For example, a major chunk of the work along this line have focused on designing polynomial time algorithms for NP-complete problems under different stability conditions. Bilu and Linial [27] introduced a notion of stability, which they termed as $\psi$-perturbation resilience for some $\psi > 1$. Informally, an instance is called $\psi$-perturbation resilient if the optimal solution remains same even after the instance is perturbed by a factor of $\psi$.

Recently, researchers have shown overwhelming interest in studying geometric clustering problems under perturbation resilience. An instance of a clustering problem is said to satisfy $\psi$-perturbation-resilience if the optimal clustering is unique and the optimal clustering remains unchanged under $\psi$-factor perturbation of the input distances. The increasing interest in studying perturbation resilient clustering has given rise to several open directions. One such interesting direction is to study clustering problems, where the clustering is not necessarily induced by Voronoi partition. NUkC is an example of such a clustering problem. In NUkC, from a feasible placement of the balls, a clustering is retrieved in the following way – each point is assigned to a fixed ball that contains the point and then for each ball, the points that are assigned to that ball form a cluster. Figure 1.5 shows that the optimal clustering for an instance of NUkC is not the same as the Voronoi partition w.r.t. the centers of the balls in the optimal placement.

For more clarity, we describe the notion of $\psi$-perturbation-resilience in the context of NUkC using two examples in Figure 1.6 (top-left and bottom-left). In all
Figure 1.5. The optimal clusters are contained in the two disks on the Euclidean plane. The centers of balls are shown by boxes. The Voronoi partition w.r.t. the centers contains two subsets of points lying on the different sides of the vertical bisector line.

Figure 1.6. Examples demonstrating the definition of perturbation resilience. The top-right (resp. bottom-right) instance is a 2-perturbed instance of the top-left (resp. bottom-left) instance. The points in same optimal cluster are shown by same shape and color.

For the instance shown at the top-left figure, let $r_1 = 10$. We claim that this instance is 2-perturbation-resilient. To see this note that here the optimal dilation is 1, and the optimal clusters are $\{a, c\}$ and $\{b, d\}$. Moreover, even if all the distances are perturbed by a factor of 2, the distance between $a$ and $c$ (resp. $b$ and $d$) can be at most 20. Hence, the dilation of the previous clustering for the perturbed instance would be at most 2. But, as all the distances between $a$ and $b, a$ and $d, c$ and $b,$
and $c$ and $d$ are 50, in any 2-perturbation of the distances, the distance between the two points in any of these four pairs would be at least 25. Thus if both the points in such a pair remain in same cluster, the dilation must be at least 2.5. As there is a clustering of dilation at most 2, in optimal clustering, both the points cannot lie in the same cluster. Hence, the optimal clustering is unique and same as the one before. The top-right figure shows a 2-perturbed instance with the same optimal clustering. Now, consider the instance in the bottom-left figure. Let $r_1 = 15$. We claim that this instance is not 2-perturbation-resilient. To prove this we show a 2-perturbed instance where the optimal clustering is different. Note that in the original instance, the optimal dilation is 1, and the optimal clusters are $\{a, c\}$ and $\{b, d\}$. The 2-perturbed instance we consider is shown in the bottom-right figure. Note that in the perturbed instance the optimal clustering is $\{\{a, b\}, \{c, d\}\}$ with dilation $25/15 = 5/3$. This is because any other clustering has a dilation at least $30/15 = 2$.

### 1.3.1 Previous Work

Using a reduction from the Firefighters problem [5], Chakrabarty et al. [33] showed that the problem is $\textbf{NP}$-hard to approximate within any constant factor even in tree metric. On the other hand, they have designed a $(c_1, c_2)$ “bi-criteria” approximation for the problem for large constants $c_1$ and $c_2$, i.e., if the algorithm is allowed to use $c_1$ balls of each type (thus $c_1 \cdot k$ in total), it can produce a solution with dilation at most $c_2$ times the optimal dilation. Their algorithm is based on the techniques used in a recent breakthrough result of [5] where a constant approximation is given for the Firefighters problem on trees. They also give a $(1 + \sqrt{5})$-approximation for
NUkC with two radii classes. Note that, for the special case of $k$-center, several 2-approximation algorithms were already known. However, even when the number of distinct radii is 3, no approximation is known other than the bi-criteria one for the general problem. To the best of our knowledge, the problem has not been studied specifically in the Euclidean setting.

Next, we discuss the literature of clustering problems under perturbation resilience. Balcan and Liang [18] showed that the standard center based clustering problems (e.g. $k$-center, $k$-median) can be solved in polynomial time under $\psi$-perturbation-resilience for $\psi \geq 1 + \sqrt{2}$. Later, Balcan et al. [17] improved the bound for $k$-center to 2. They also showed that $k$-center under $\psi$-perturbation-resilience cannot be solved in polynomial time for $\psi < 2$, unless $\textsf{NP} = \textsf{RP}$. Angelidakis et al. [10] gave a generic polynomial time algorithm for clustering problems with center based objectives (e.g. $k$-center, $k$-median, $k$-means) under 2-perturbation-resilience. In any such center based objective, the clustering is obtained by assigning a point to its nearest center. In other words, the clustering is induced by the Voronoi partition of the points w.r.t. the chosen centers. Recently, Cohen-Addad and Schwiegelshohn [44] proved that a simple local search scheme yields optimal solutions for problems like $k$-median and $k$-means, under $\psi$-perturbation-resilience for $\psi > 3$. Chekuri and Gupta [38] showed that an LP relaxation of a $k$-center under 2-perturbation-resilience admits an integral solution. They also proved the same result for $k$-center with outliers. Balcan and Liang [18] introduced a weaker notion called $(\psi, \epsilon)$-perturbation-resilience, where the optimal solution under $\psi$-perturbation can differ in at most $\epsilon$
fraction of the points from the original optimal clustering. Assuming that each cluster contains more than $2\epsilon n$ points, Balcan et al. [17] showed that $k$-center under $(3, \epsilon)$-perturbation-resilience can be solved in polynomial time, where $n$ is the number of input points.

### 1.3.2 Our Results and Contributions

We design an approximation algorithm for the NUkC problem with three radii classes that uses $k_1$ balls of radius $r_1$, $(2 + \epsilon)k_2$ and $(2 + \epsilon)k_3$ balls of radius $r_2$ and $r_3$, respectively, and yields a solution whose dilation is at most $4.38$-factor of the optimal dilation. Here $\epsilon$ is a very small constant. We emphasize that the number of balls of the largest radius used by our algorithm is at most the allowed value $k_1$. However, it uses $(1 + \epsilon)k_3$ (resp. $(1 + \epsilon)k_2$) more balls corresponding to the smallest (resp. second smallest) radius class than what is allowed. To obtain this result, we use the LP aware reduction in [33] from NUkC problem to Firefighters problem (formally defined in Section 4.1). The reduction gives an instance of the Firefighters problem on trees with height $3$ and a fractional LP solution of the problem to the instance. Also the reduction has the property that if one can find an integral LP solution to the Firefighters instance, then it is also possible to obtain a constant approximation for NUkC. We design a randomized rounding scheme to obtain an integral LP solution from the fractional solution that violates some constraints by at most a factor of $(2 + \epsilon)$. Consequently, we obtain the bi-criteria approximation for NUkC.

For the Euclidean version of NUkC, we prove that it is not possible to get any "good" approximation for the problem. To prove the result, first we strengthen the
result of Chakrabarty et al. [33] to show that NUkC is hard to approximate within a factor of $\gamma$ in tree metric for any $\gamma$. Then, we reduce the NUkC problem in tree metric to Euclidean NUkC. In particular, we use the metric embedding result from [63] as a core of our reduction. The result in [63] shows that one can embed a tree metric on an Euclidean metric without distorting the distances by a large factor.

For the NUkC problem under perturbation resilience, we obtain the following results. First, we give a polynomial time exact algorithm for NUkC with a constant number of radii classes under 2-perturbation-resilience/(3, $\epsilon$)-perturbation-resilience. Our algorithm reduces the NUkC problem to a version of Firefighters problem on trees (formally defined in Section 4.3.2). Under the stability assumptions, we can show that a feasible solution of NUkC maps to a feasible solution of Firefighters problem and vice versa. Here, in particular, we use the “well-separated” structure of the clusters in the optimal clustering that follows due to stability. The reduction has the property that if NUkC has $t$ distinct radii classes, then the height of the constructed tree instance is $t + 1$. Then, we show that using a dynamic programming based scheme, the Firefighters problem can be solved in polynomial time for constant height tree instances. Thus, we also obtain a polynomial time algorithm for NUkC with a constant number of radii classes. We note that the algorithms for center based clustering problems in [10, 18, 38] are also based on tree computation and dynamic programming. However, the structure of the tree we compute are very different and has constant height. We also note that our result under 2-perturbation-resilience is tight, as even for $k$-center it is unlikely to obtain a polynomial time
algorithm under $\psi$-perturbation-resilience for $\psi < 2$. To prove the result for $(3, \epsilon)$-perturbation-resilience, we assume that each cluster contains more than $\epsilon n + 1$ points. We note that such a lower bound is necessary, as in its absence even $k$-center is \textbf{NP}-hard [17] under $(\psi, \epsilon)$-perturbation-resilience for all $\psi \geq 1$ and $\epsilon > 0$. For the general NU$k$C problem, we show that the perturbation resilience assumption does not make the problem any easier. In particular, we show a $\gamma$-inapproximability result for NU$k$C under $\sqrt{\gamma}$-perturbation-resilience, unless \textbf{NP} = \textbf{RP}. Our result implies that, for any $\psi, \gamma > 1$, even with $\psi$-perturbation-resilience one cannot hope to find a $\gamma$-approximation for the problem. To prove the result, we use a chain of reductions starting from the satisfiability problem to the NU$k$C problem in tree metric under perturbation resilience assumption. The last reduction in the chain is from a version of Firefighters problem, which shows that NU$k$C is hard to approximate within a factor of $\gamma$ in tree metric for any $\gamma$. We note that this is a strengthening of the result due to Chakrabarty et al. [33] that proves only a constant inapproximabilty of the problem. Our reduction is however similar to the reduction in [33]. Then, we argue that the constructed tree instances of NU$k$C are $\sqrt{\gamma}$-perturbation-resilient, and hence the similar hardness follows even for NU$k$C under $\sqrt{\gamma}$-perturbation-resilience. All these results are discussed in Chapter 4.

1.4 Capacitated Covering

Metric Capacitated Covering (MCC) is a special case of the \textit{Capacitated Set Cover} (CSC) problem. In the latter problem, we are given a set system $(X, F)$ with $n = |X|$ elements and $m = |F|$ subsets of $X$. For each set $F_i \in F$, we are also given
an integer $U_i$, which is referred to as its capacity. We are required to find a minimum size subset $\mathcal{F}' \subseteq \mathcal{F}$ and assign each element in $X$ to a set in $\mathcal{F}'$ containing it, such that for each set $\mathcal{F}_i$, the number of points assigned to $\mathcal{F}_i$ is at most $U_i$. The MCC is obtained as a special case of CSC by setting $X = P$, $\mathcal{F} = \mathcal{B}$, and $U_i = U \ \forall i$. The more popular Set Cover problem is a special case of CSC where the capacity of each set is $\infty$. We will refer to the metric version of Set Cover as Metric Uncapacitated Covering (MUC). Note that MUC is a special case of MCC when $U = \infty$. In the following, we discuss some work from the literature of covering problems.

1.4.1 Previous Work

In 1974, Johnson [78] gave a greedy algorithm for Set Cover that yields an $O(\log n)$-approximation, i.e., the solution that the algorithm produces is at most $O(\log n)$ times worse than any optimal solution. Moreover, the approximation factor cannot be improved asymptotically unless $P=NP$ [47, 108]. Though MUC is a special case of Set Cover, the $\Omega(\log n)$ lower bound on the approximation factor still holds for it as one can show by a straightforward reduction from Set Cover [19]. It follows that MCC is also NP-hard to approximate within an $o(\log n)$ factor of an optimal solution. In fact, both of the problems are NP-hard to approximate within a factor of $f(n)$, for $f(n) = o(\log n)$, even if we are allowed to expand the balls by $\lambda$ factor for any $\lambda < 3$ [19].

The MUC problem in fixed dimensional Euclidean spaces has been extensively studied. One interesting variant is when the allowed set $\mathcal{B}$ of balls consists of all unit balls. Hochbaum and Maass [72] gave an approximation algorithm for this variant.
using a grid shifting strategy and proved that it is a polynomial time approximation scheme (PTAS), i.e., for any $\varepsilon > 0$, their algorithm returns a solution, which is at most $(1 + \varepsilon)$ factor worse than an optimal solution. When $\mathcal{B}$ is an arbitrary finite set of balls, the problem seems to be much harder. An $O(1)$-approximation algorithm in the 2-dimensional Euclidean plane was given by Brönnimann and Goodrich [30]. More recently, Mustafa and Ray [104] obtained a PTAS. In dimensions 3 and higher, the best known approximation guarantee is still $O(\log n)$. Motivated by this, Har-Peled and Lee [70] gave a PTAS for a “bi-criteria” version where the algorithm is allowed to expand the input balls by $(1 + \varepsilon)$ factor. Covering with geometric objects other than balls has also been extensively studied; see [89, 40, 11, 113, 35, 62] for a sample.

For the CSC problem, Wolsey [114] used a greedy algorithm to give an $O(\log n)$ approximation. For the special case of vertex cover (where each element in $X$ belongs to exactly two sets in $\mathcal{F}$), Chuzhoy and Naor [39] presented an algorithm with approximation ratio 3, which was subsequently improved to 2 by Gandhi et al. [56]. The generalization where each element belongs to at most a bounded number $f$ of sets has been studied in a sequence of works, culminating in [82, 115]. Berman et al. [24] have considered the “soft” capacitated version of the CSC problem that allows making multiple copies of input sets. Another closely related problem to the CSC problem is the so-called Replica Placement problem. For the graphs with treewidth bounded by $t$, an $O(t)$ approximation algorithm for this problem is presented in [7]. Finally, PTASes for the Capacitated Dominating Set, and Capacitated Vertex Cover
problems on the planar graphs are presented in [23], under the assumption that the demands and capacities of the vertices are upper bounded by a constant.

Compared to MUC, relatively fewer geometric versions of the MCC problem have been studied in the literature. One such version arises when $\mathcal{B}$ comprises of all possible unit balls. This problem appeared in the Sloan Digital Sky Survey project [94]. Building on the shifting strategy of Hochbaum and Maass [72], Ghasemi and Razzazi [58] obtain a PTAS for this problem. When the set $\mathcal{B}$ of balls is arbitrary, the best known approximation guarantee is $O(\log n)$, even in the plane.

1.4.2 Our Results and Contributions

In this work, we make significant progress on both the MCC and ECC problems.

- We present an $(O(1), 6.47)$-approximation for the MCC problem. Thus, if we are allowed to expand the input balls by a constant factor, we can obtain a solution that uses at most $O(1)$ times the number of balls used by the optimal solution. As noted above, if we are not allowed to expand by a factor of at least 3, we are faced with a hardness of approximation of $\Omega(\log n)$.

- We present an $(O(\epsilon^{-4d} \log(1/\epsilon)), 1 + \epsilon)$-approximation for the ECC problem in $\mathbb{R}^d$. Thus, assuming we are allowed to expand the input balls by an arbitrarily small constant factor, we can obtain a solution with at most a corresponding constant times the number of balls used by the optimal solution. Without expansion, the best known approximation guarantee for $d \geq 3$ is $O(\log n)$, even
without capacity constraints.

Both results are obtained via a unified scheme for rounding the natural LP relaxation for the problem. This scheme, which is instantiated in different ways to obtain the two results, may be of independent interest for obtaining similar results for related capacitated covering problems. Though the LP rounding technique is a standard tool that has been used in the literature of the capacitated problems, our actual rounding scheme is different from the existing ones. In fact, the standard rounding scheme for facility location, for example the one in [90], is not useful for our problems, as there a point can be assigned to any facility. But in our case, each point must be assigned to a ball that contains it (modulo constant factor expansion). This hard constraint makes the covering problems more complicated to deal with.

When the input balls have the same radius, it is easier to obtain the above guarantees for MCC and ECC using known results or techniques. For the MCC, this (in fact, even a (1, O(1))-approximation) follows from the results for capacitated $k$-center [20, 84, 45, 9]. This is because of the connection between Capacitated $k$-center and MCC as pointed out before. The novelty in our work lies in handling the challenging scenario where the input balls have widely different radii. For geometric optimization problems, inputs with objects at multiple scales are often more difficult to handle than inputs with objects at the same scale.

As a byproduct of the rounding schemes we develop, the bicriteria approximations can be extended to a more general capacity model. In this model, the capacities of the balls are not necessarily the same. In particular, suppose ball $B_i$ has capac-
ity $U_i$ and radius $r_i$. Then, for any two balls $B_i, B_j \in \mathcal{B}$, our model assumes that the following holds: $r_i > r_j \implies U_i \geq U_j$. We refer to this capacity model as the monotonic capacity model. We refer to the generalizations of the MCC and the ECC problems with the monotonic capacity model as the Metric Monotonic Capacitated Covering (MMCC) problem and the Euclidean Monotonic Capacitated Covering (EMCC) problem, respectively. We note that the monotonicity assumption on the capacities is reasonable in many applications such as wireless networks – it might be economical to invest in capacity of an antenna to serve more clients, if it covers more area.

**Hardness.** We complement our algorithmic results with some hardness of approximation results that give a better understanding of the problems we consider. Firstly, we show that for any constant $c > 1$, there exists a constant $\epsilon_c > 0$ such that it is $\mathsf{NP}$-hard to obtain a $(1 + \epsilon_c, c)$-approximation for the MCC problem, even when the capacity of all balls is 3. This shows that it is not possible to obtain a $(1, c)$ approximation even for an arbitrarily large constant $c$. In the hardness construction, not all the balls in the hard instance have the same radius. This should be contrasted with the case where the radii of all balls are equal – in this case one can use the results from capacitated $k$-center (such as [9, 45]), to obtain a $(1, O(1))$-approximation.

It is natural to wonder if our algorithmic results can be extended to weighted versions of the problems. We derive hardness results that indicate that this is not possible. In particular, we show that for any constant $c \geq 1$, there exists a constant $c' > 0$, such that it is $\mathsf{NP}$-hard to obtain a $(c' \log n, c)$-approximation for the weighted
version of MMCC with a very simple weight function (a constant power of original radius). This implies that one cannot find a $o(\log n)$-approximation for this version even if we are allowed to expand the balls by any constant factor. All these results are discussed in Chapter 5.
CHAPTER 2

K-MEANS CLUSTERING

In this chapter, we describe the bi-criteria approximation algorithm for \( k \)-means. But before moving on we discuss some background related to the techniques we use, and have some definitions and notations that we are going to use throughout this section.

2.1 Preliminaries

**Definition 1.** A \((\beta, \gamma)\) bi-criteria approximation algorithm for \( k \)-means is an algorithm that uses at most \( \beta k \) centers and yields a solution whose cost is at most \( \gamma \) times the cost of an optimal \( k \)-means solution.

We design a \((1 + \varepsilon, 1 + \varepsilon)\) bi-criteria approximation algorithm for \( k \)-means. For a point \( p \) and a set \( R \) of points, let \( d(p, R) = \min_{q \in R} ||p - q|| \). Given a point set \( R \) in \( \mathcal{R}^d \), the voronoi diagram of \( R \), denoted by \( \mathcal{V}_R \), is the partition of \( \mathcal{R}^d \) into \( |R| \) convex cells, where the voronoi cell of \( p \in R \) is

\[
\mathcal{V}_R(p) = \{ q \in \mathcal{R}^d | d(p, q) = d(q, R) \}
\]

The algorithm we design for \( k \)-means is based on local search technique. Next, we describe some applications of this technique in the context of geometric problems.

**Local Search.** Local search is a popular technique in combinatorial optimization. But the technique was not much in use for geometric problems until recently. Still
we know very little regarding this technique for geometric approximation. Arya et al. [15] gave a $3 + \frac{2}{p}$ factor approximation for $k$-median based on a local search that swaps $p$ facilities, which was later simplified by Gupta and Tangwongsan [64]. The $9 + \varepsilon$ factor approximation for $k$-means, as mentioned before, is based on the approach of Arya et al. [15]. Mustafa and Ray [103] gave a local search PTAS for the discrete hitting set problem over pseudodisks and $r$-admissible regions in the plane. Chan and Har-Peled [36] designed a local search algorithm for the independent set problem over fat objects, and for pseudodisks in the plane, which yields a PTAS. Recently, Cohen-Addad and Mathieu [43] showed the effectiveness of local search technique for geometric optimization by designing local search algorithms for many geometric problems including $k$-median and facility location. For facility location they achieved a PTAS. For $k$-median their approach yields a $1 + \varepsilon$ factor approximation by using at most $(1 + \varepsilon)k$ centers. Very recently, Bhattiprolu and Har-Peled [25] designed a local search PTAS for a geometric hitting set problem over balls in $\mathbb{R}^d$. Their PTAS also works for those infinite set of balls which can be represented implicitly in a specific way mentioned in their paper. Also it is worth it to mention that one of the bi-criteria algorithms for $k$-means in [96], as mentioned before, is based on local search method.

Like in [25, 36, 43, 103] the analysis of our algorithm extensively uses separators to prove the quality of the approximation. Many algorithms are based on divide and conquer strategy. These algorithms partition the input into several parts, solve the problem in those parts independently, and merge the solution of each subproblem to
get the solution for the original problem. For partitioning of the input, one natural choice is to use separators. There are several different notions of separators in the literature as we describe below.

**Graph Separators.** A separator for a graph $G$ is a “small” subset of vertices, whose deletion from $G$ generates at least two subgraphs, each of which contains at most a constant fraction of the vertices. For example, for any tree $T$, there is a vertex in $T$, deletion of which from $T = (V, E)$ generates subgraphs, each of which is of size at most $|V|/2$. Lipton and Tarjan [91] showed that any planar graph with $n$ vertices contains a separator of size $O(\sqrt{n})$, which can be computed in linear time. There are also proofs of the existence of “small” separators for graphs (i) with bounded treewidth, (ii) with bounded genus, (iii) that are minor free, and (iv) that are grids.

**Geometric Separators.** The circle packing theorem says, any planar graph can be realized by a set of interior disjoint disks in the plane, where a pair of disks is corresponding to an edge if they touch each other [105]. This theorem is also referred to as Koebe-Andreev-Thurston theorem and its proof can be traced in a work of Koebe in 1936 [86]. The existence of the planar graph separator also follows from this circle packing theorem [67, 100]. Miller et al. [100] showed that for any set of $n$ balls in $\mathbb{R}^d$, such that there is no point in $\mathbb{R}^d$ that is contained in more than $k$ balls, the intersection graph of the balls has a separator of size $O(k^{\frac{1}{2}}n^{1-\frac{1}{d}})$. This implies that a $k$ nearest neighbor graph of a set of points in $\mathbb{R}^d$, has a small separator [67, 100]. Recently, Adamaszek and wiese [3] proved that for any given set of $n$ axes-parallel
rectangles in the plane, there is a closed polygonal curve $C$ with $O\left(\frac{1}{\varepsilon}\right)$ edges, such that there are at most $2n/3$ rectangles both inside and outside of $C$ and $C$ intersects at most $\varepsilon n$ rectangles. Based on this observation they presented a QPTAS for independent set of weighted axes parallel rectangles [3], and subsequently extended their approach to polygons with polylogarithmic many vertices [4]. Har-peled [68] simplified and generalized their result to polygons of arbitrary complexity. Mustafa et al. [102] also describe a simplification, other generalizations, and an application to obtain a QPTAS for computing a minimum weight set cover using pseudo-disks.

**Voronoi Separators.** Given a point set $R$ in $\mathcal{R}^d$, a voronoi separator of $R$ is a "small" set $Z \subseteq \mathcal{R}^d$, such that there is a partition of $R$ into two subsets $R_1$ and $R_2$ so that the cells of $R_1$ do not touch the cells of $R_2$ in the voronoi diagram $\mathcal{V}_{R \cup Z}$. Recently, Bhattiprolu and Har-Peled [25] have shown that for any set $R$ of $n$ points in $\mathcal{R}^d$, there is a voronoi separator of size $O(n^{1-\frac{1}{d}})$. In our analysis of the local search algorithm we will extensively use the voronoi separators.

Before describing the result for $k$-means, we describe the PTAS for the Sum of Squares Facility Location problem which also uses a similar local search technique. This will be helpful to understand the result for $k$-means.

### 2.2 Sum of Squares Facility Location Problem

Recall that in SOS-FL we are given a set of clients $C$ in $\mathcal{R}^d$ and the facility opening cost $f > 0$. Let $|C| = n$. 
2.2.1 The Local Search Algorithm

Fix an $\varepsilon > 0$. The local search algorithm starts with the solution where one facility is placed at each client (see Algorithm 2.1). Note that the cost of this solution is $nf$. Denote by $OPT$ the cost of any optimal solution. As $OPT \geq f$, the initial solution has cost at most $O(n \cdot OPT)$. In each iteration the algorithm looks for local improvement. The algorithm returns the current set of facilities if there is no such improvement. Notice that in line 2, we consider swaps with at most $\frac{c}{\varepsilon^d}$ facilities, where $c$ is a constant. Next we show that this algorithm runs in polynomial time.

Algorithm 2.1 Local Search

Require: A set of clients $C \subset \mathbb{R}^d$, a constant $\varepsilon > 0$.
Ensure: A set of facilities $F$.
1: $F \leftarrow$ the set of facilities with one facility at each client
2: while $\exists$ a set $F_1$ s.t. $\text{cost}(F_1) < (1 - \frac{1}{n})\text{cost}(F)$ and $|F_1 \setminus F| + |F \setminus F_1| \leq \frac{\varepsilon}{\varepsilon^d}$ do
3: $F \leftarrow F_1$
4: end while
5: return $F$

We note that in each iteration of the while loop the cost of $F$ gets dropped by a factor of at least $(1 - \frac{1}{n})$. Let the number of iterations of the while loop is $\tau$. As we start with a solution of cost at most $O(n \cdot OPT)$, after $\tau$ iterations

$$\text{cost}(F) \leq (1 - \frac{1}{n})^{\tau} \cdot n \cdot OPT$$

As $\text{cost}(F) \geq OPT$, we have

$$OPT \geq (1 - \frac{1}{n})^{\tau} \cdot n \cdot OPT$$
Thus the number \( \tau \) of iterations of the while loop is \( O(n \log n) \). Now consider a single iteration of the while loop: we need to compute if there exists an \( F_1 \) such that 
\[
\text{cost}(F_1) < (1 - \frac{1}{n})\text{cost}(F) \quad \text{and} \quad |F_1 \setminus F| + |F \setminus F_1| \leq \frac{\epsilon}{2^r}.
\]
Assuming such an \( F_1 \) exists, fix one such \( F_1 \). Let \( A = F \setminus F_1 \). Since \( |A| \leq \frac{\epsilon}{2^r} \), we can enumerate over the possibilities for \( A \). Let us therefore assume we have \( A \). We would like to find a set \( A' \subset \mathbb{R}^d \) with 
\[
t := |A'| \leq \frac{\epsilon}{2^r} - |A| \quad \text{that minimizes} \quad \text{cost}((F \cap F_1) \cup A').
\]
Since each point in \( A' \) has \( d \) coordinates, this is an optimization problem with \( t \cdot d \) variables for any fixed \( t \).

This optimization problem can be solved in polynomial time using techniques similar to the ones used in [75]. To explain this further, let the optimal \( A' \) be \( \{a'_1, a'_2, \ldots, a'_t\} \), let \( \chi \) be the assignment that assigns each client in \( C \) to the nearest facility in \( (F \cap F_1) \cup A' \). For solving our optimization problem, it suffices to know \( \chi \), even though \( A' \) is unknown: we can set \( a'_i \) to be the centroid of the points \( \chi^{-1}(a'_i) \).

For \( c \in C \), there are \( t + 1 \) possibilities for \( \chi(c) \): the nearest facility in \( F \setminus F_1 \), or one of the \( t \) unknown facilities in \( A' \). Naively, this suggests \( (t + 1)^n \) possibilities for \( \chi \). However, if we view the problem as an optimization problem in \( t \cdot d \) dimensions, and use standard ideas about arrangements, we can see that we only need to consider \( n^{O(t \cdot d)} \) possibilities for \( \chi \), and these can be enumerated in \( n^{O(t \cdot d)} \) time. We conclude that Algorithm 2.1 can be implemented in polynomial time.

### 2.2.2 Analysis of the Local Search Algorithm

We analyze the local search algorithm using a partitioning scheme based on the separator theorem from [25]. To our surprise the analysis is simple. The idea is to partition the set of facilities in the local search solution and an optimal solution into
parts of size $O(\frac{1}{\varepsilon^d})$. Now for each such small part, we assign the clients corresponding to the local facilities of that part, either to the optimal facilities in that part or to the points belong to a special set. This yields a new solution, whose symmetric difference with the local search solution contains $O(\frac{1}{\varepsilon^d})$ facilities. Thus using the local optimality criteria the cost of this new solution is not “small” compared to the local search solution. Then we combine the new solutions corresponding to all the parts to give a bound on the cost of the local search solution. To start with we describe the separator theorem.

### 2.2.3 Separator Theorem

A ball $B$ is said to be stabbed by a point $p$ if $p \in B$. The following theorem is due to Bhattiprolu and Har-Peled [25] which shows the existence of a “small” point set (separator), that divides a given set of points into two in a “balanced” manner.

**Theorem 2.** [25] (Separator Theorem) Let $X$ be a set of points in $\mathbb{R}^d$, and $\mu > 0$ be an integer such that $|X| > \alpha \mu$, where $\alpha$ is a constant. There is an algorithm which can compute, in $O(|X|)$ expected time, a set $Z$ of $O(\mu^{1 - \frac{1}{d}})$ points and a sphere $S$ containing $\Theta(\mu)$ points of $X$ inside it, such that for any set $B$ of balls stabbed by $X$, we have that every ball of $B$ that intersects $S$ is stabbed by a point of $Z$.

The next corollary follows from Theorem 2 which will be useful for the analysis of our local search algorithm.

**Corollary 3.** Let $X$ be a set of points in $\mathbb{R}^d$, and $\mu > 0$ be an integer such that $|X| > \alpha \mu$, where $\alpha$ is a constant. There is an algorithm which can compute, in
expected time, a set $Z$ of $O(\mu^{1-\frac{1}{2}})$ points and a ball $B$ containing $\Theta(\mu)$ points of $X$ in it, such that for any point $p \in R^d$, $d(p, Z) \leq \max\{d(p, X \setminus B), d(p, B \cap X)\}$.

Proof. We use the same Algorithm in Theorem 2 to compute the sphere $S$ and the set $Z$. Let $B$ be the ball that has $S$ as its boundary. Now consider any point $p \in R^d$. Let $p_1$ (resp. $p_2$) be a point in $B \cap X$ (resp. $X \setminus B$) nearest to $p$. If $p \in B$, the ball $B_1$ centered at $p$ and having radius $|p - p_2|$ must intersect the sphere $S$, as $p_2 \notin B$. As $p_2$ stabs $B_1$, by Theorem 2 there is a point in $Z$ that also stabs $B_1$. Hence $d(p, Z) \leq ||p - p_2|| = d(p, X \setminus B)$. Similarly, if $p \notin B$, the ball $B_2$ centered at $p$ and having radius $||p - p_1||$ intersects the sphere $S$, as $p_1 \in B$. As $B_2$ is stabbed by $p_1$, by Theorem 2 there is a point in $Z$ that also stabs $B_2$. Hence $d(p, Z) \leq ||p - p_1|| = d(p, B \cap X)$ and the corollary follows. 

The algorithm in Corollary 3 will be referred to as the Separator algorithm.

2.2.4 The Partitioning Algorithm

For the sake of analysis fix an optimal solution $O$. Let $L$ be the solution computed by the local search algorithm. We design a procedure $\text{PARTITION}(L, O, \epsilon)$ which divide the set $L \cup O$ into disjoint subsets of small size using the Separator algorithm (see Algorithm 2.2). The procedure iteratively removes points from the set until the size of the set becomes less than or equal to $\alpha \mu$, where $\alpha$ is the constant in Corollary 3, and $\mu = \frac{\gamma}{d}$ for some constant $\gamma$. This procedure is similar to the ApproximateSeparator algorithm in [25]. Next, we describe some important properties of this procedure which will be helpful to give an approximation bound on the cost of the local search solution. But before proceeding further we define some notation.
Algorithm 2.2 PARTITION($\mathcal{L}, \mathcal{O}, \varepsilon$)

1: $\mu = \frac{\varepsilon}{\mathcal{N} \mathcal{L} \mathcal{O} \varepsilon}$, $i = 1$, $L_1 = \mathcal{L}$, $O_1 = \mathcal{O}$, $Z_1 = \emptyset$
2: while $|L_i \cup O_i \cup Z_i| > \alpha \mu$ do
3: Let $B_i, T_i$ be the ball and the point set computed by applying the Separator algorithm on the set $L_i \cup O_i \cup Z_i$ with parameter $\mu$
4: $L_{i+1} = L_i \setminus B_i$, $O_{i+1} = O_i \setminus B_i$
5: $Z_{i+1} = (Z_i \setminus B_i) \cup T_i$
6: $i = i + 1$
7: end while
8: $T_I = \emptyset$
9: Let $B_I$ be any ball that contains all the points in $L_I \cup O_I \cup Z_I$
10: $T_I = \emptyset$

Let $T = \bigcup_{i=1}^{I} T_i$ be the union of the point sets computed by the Separator algorithm in PARTITION($\mathcal{L}, \mathcal{O}, \varepsilon$). Also let $C_l = \{p \in C | d(p, \mathcal{L}) \leq d(p, \mathcal{O})\}$ and $C_o = C \setminus C_l$. Consider a point set $R \subset \mathcal{R}^d$. We denote the nearest neighbor voronoi diagram of $R$ by $\mathcal{V}_R$. For $p \in R$, let $\mathcal{V}_R(p)$ be the voronoi cell of $p$ in $\mathcal{V}_R$. Also let $C_R(p) = \mathcal{V}_R(p) \cap C$, that is $C_R(p)$ is the set of clients that are contained in the voronoi cell of $p$ in $\mathcal{V}_R$. For $Q \subseteq R$, define $C_R(Q) = \cup_{q \in Q} C_R(q)$.

Now consider a client $c$. Denote its nearest neighbor in $\mathcal{O}$ (resp. $\mathcal{L}$) by $c(\mathcal{O})$ (resp. $c(\mathcal{L})$). Also let $c_O = ||c - c(\mathcal{O})||^2$ and $c_L = ||c - c(\mathcal{L})||^2$.

**Definition 4.** An assignment is a function that maps a set of clients to the set $\mathcal{L} \cup \mathcal{O} \cup T$.

Now with all these definitions we move on towards the analysis. We begin with the following observation.

**Observation 5.** Consider the procedure PARTITION($\mathcal{L}, \mathcal{O}, \varepsilon$). The following state-
ments are true.

1. \(|L_i \cup O_i \cup T_i \cup (Z_i \cap B_i)| \leq \frac{\beta}{\varepsilon^2} \text{ for a constant } \beta \text{ and } 1 \leq i \leq I\)

2. \(I \leq \varepsilon(|L| + |O|)/10\)

3. \(|T| \leq \varepsilon(|L| + |O|)/10\)

4. \(\sum_{i=1}^{I} |T_i \cup (Z_i \cap B_i)| \leq \varepsilon(|L| + |O|)/5\)

Proof. 1. Note that \(|T_i| = O(\mu^{1-\frac{1}{2}}) = O\left(\frac{1}{\varepsilon^{2d}}\right)\) and \(|L_i \cup O_i \cup Z_i| = \theta(\mu) = \theta\left(\frac{1}{\varepsilon^d}\right)\) for \(1 \leq i \leq I\). Thus there is a constant \(\beta\) such that \(|L_i \cup O_i \cup T_i \cup (Z_i \cap B_i)| \leq \frac{\beta}{\varepsilon^2}\).

2. As in each iteration we add \(O(\mu^{1-\frac{1}{2}})\) points and remove \(\theta(\mu)\) points, the number of iterations \(I = O\left(\frac{|L| + |O|}{\mu}\right) = O\left(\varepsilon^d\left(\frac{|L| + |O|}{\gamma}\right)\right)\). By choosing the constant \(\gamma\) sufficiently large one can ensure that \(I \leq \varepsilon(|L| + |O|)/10\).

3. \(|T| = \sum_{i=1}^{I} |T_i| = O\left(\frac{|L| + |O|}{\mu}\cdot \mu^{1-\frac{1}{2}}\right) = O\left(\varepsilon(\varepsilon(|L| + |O|)/\gamma^2)\right) \leq \varepsilon(|L| + |O|)/10\), by choosing the value of \(\gamma\) sufficiently large.

4. Consider any point \(p \in T_i\) for some \(1 \leq i \leq I\). If \(p \in Z_j \cap B_j\) for some \(j > i\), then \(p\) is removed in iteration \(j\) and hence cannot appear in any other \(Z_t \cap B_t\) for \(j+1 \leq t \leq I\). Thus \(p\) can appear in at most two sets of the collection \(\{T_1 \cup (Z_1 \cap B_1), \ldots, T_I \cup (Z_I \cap B_I)\}\). It follows that \(\sum_{i=1}^{I} |T_i \cup (Z_i \cap B_i)| \leq 2 \sum_{i=1}^{I} |T_i| = 2|T| \leq \varepsilon(|L| + |O|)/5\). \(\square\)

The next lemma states the existence of a “cheap” assignment for any client \(c\), such that its nearest neighbor \(c(O)\) in \(O\) is in \(O_i\), its nearest neighbor \(c(L)\) in \(L\) is in \(L_j\) with \(i < j\).
Lemma 6. Consider any client \(c\), such that \(c(\mathcal{O}) \in \mathcal{O}_i, c(\mathcal{L}) \in \mathcal{L}_j\) with \(1 \leq i < j \leq I\). Also consider the sets \(T_j\), \(\mathcal{Z}_j\), and the ball \(B_j\) computed by PARTITION(\(\mathcal{L}, \mathcal{O}, \varepsilon\)). There exists a point \(p \in (\mathcal{Z}_j \cap B_j) \cup T_j\) such that \(\|c - p\| \leq \max\{\|c - c(\mathcal{O})\|, \|c - c(\mathcal{L})\|\}\).

Proof. To prove this lemma, at first we prove the following claim.

Claim 7. For any \(i + 1 \leq t \leq j\), there exists a point \(p \in \mathcal{Z}_t\) such that \(\|c - p\| \leq \max\{\|c - c(\mathcal{O})\|, \|c - c(\mathcal{L})\|\}\).

Proof. We prove this claim using induction on the iteration number. In base case consider the iteration \(i\). Let \(X = L_i \cup O_i \cup Z_i\). As \(c(\mathcal{O}) \in B_i\) and \(c(\mathcal{L}) \in B_j\), \(c(\mathcal{O}), c(\mathcal{L}) \in X\). Now by Corollary 3, \(d(c, T_i) \leq \max\{d(c, X \setminus B_i), d(c, B_i \cap X)\}\). As \(c(\mathcal{O})\) is the nearest neighbor of \(c\) in \(\mathcal{O}\) and \(c(\mathcal{O}) \in B_i\), \(d(c, B_i \cap X) \leq \|c - c(\mathcal{O})\|\). Also \(c(\mathcal{L})\) is the nearest neighbor of \(c\) in \(\mathcal{L}\) and \(c(\mathcal{L}) \notin B_i\). Thus \(d(c, X \setminus B_i) \leq \|c - c(\mathcal{L})\|\).

Hence \(d(c, T_i) \leq \max\{\|c - c(\mathcal{O})\|, \|c - c(\mathcal{L})\|\}\). Let \(p\) be the point in \(T_i\) nearest to \(c\). As \(T_i \subseteq Z_{i+1}\), \(p \in Z_{i+1}\) and the base case holds.

Now suppose the claim is true for any iteration \(t < j - 1 \leq I - 1\). We show that the claim is also true for iteration \(t + 1\). By induction, there is a point \(p \in Z_{t+1}\) such that \(\|c - p\| \leq \max\{\|c - c(\mathcal{O})\|, \|c - c(\mathcal{L})\|\}\). Now there can be two cases: (i) \(p \notin B_{t+1}\), and (ii) \(p \in B_{t+1}\). Consider the first case. In this case, by definition of \(Z_{t+2}\), \(p \in Z_{t+2}\) and the claim holds. Thus consider the second case. Let \(X = L_{t+1} \cup O_{t+1} \cup Z_{t+1}\).

By Corollary 3, \(d(c, T_{t+1}) \leq \max\{d(c, X \setminus B_{t+1}), d(c, B_{t+1} \cap X)\}\). As \(p \in B_{t+1} \cap X\), \(d(c, B_{t+1} \cap X) \leq \|c - p\| \leq \max\{\|c - c(\mathcal{O})\|, \|c - c(\mathcal{L})\|\}\). Now \(c(\mathcal{L}) \notin B_{t+1}\), as \(t < j - 1\). Also \(c(\mathcal{L}) \in L_{t+1} \subseteq X\). Thus \(d(c, X \setminus B_{t+1}) \leq \|c - c(\mathcal{L})\|\). It follows that
\(d(c, T_{t+1}) \leq \max\{\|c - c(O)\|, \|c - c(L)\|\}\). Let \(q\) be the point in \(T_{t+1}\) nearest to \(c\).

As \(T_{t+1} \subseteq \mathcal{Z}_{t+2}\), \(q \in \mathcal{Z}_{t+2}\) and the claim holds also for this case.

Consider the iteration \(j\). From Claim 7 it follows that there exists a point \(p \in \mathcal{Z}_j\) such that \(\|c - p\| \leq \max\{\|c - c(O)\|, \|c - c(L)\|\}\). Thus if \(p \in B_j\), then \(p \in \mathcal{Z}_j \cap B_j\), and we are done. Note that the way \(B_j\) is chosen, \(\mathcal{Z}_I \subseteq B_I\). Thus in case \(j = I\), \(p \in \mathcal{Z}_I \cap B_I\). Hence consider the case when \(j \neq I\) and \(p \notin B_j\). Let \(X = L_j \cup O_j \cup \mathcal{Z}_j\).

As \(p \in \mathcal{Z}_j\), \(p \in X\). Also by Corollary 3, \(d(c, T_j) \leq \max\{d(c, X \setminus B_j), d(c, B_j \cap X)\}\). As \(c(L) \in B_j \cap X\), \(d(c, B_j \cap X) \leq \|c - c(L)\|\). Now \(p \in X \setminus B_j\). Thus \(d(c, X \setminus B_j) \leq \|c - p\| \leq \max\{\|c - c(O)\|, \|c - c(L)\|\}\). Hence \(d(c, T_j) \leq \max\{\|c - c(O)\|, \|c - c(L)\|\}\) and the lemma follows.

Now we extend Lemma 6 for any client whose nearest neighbor in \(L\) is in \(L_j\), but the nearest neighbor in \(O\) is not in \(O_j\), where \(1 \leq j \leq I\).

**Lemma 8.** Consider the sets \(T_j, \mathcal{Z}_j\) and the ball \(B_j\) computed by \(\text{PARTITION}(L, O, \varepsilon)\), where \(1 \leq j \leq I\). There is an assignment \(g\) of the clients in \(C_L(L_j) \setminus C_O(O_j)\) to \(T_j \cup (\mathcal{Z}_j \cap B_j)\) with the following properties:

1. for \(c \in (C_L(L_j) \setminus C_O(O_j)) \cap C_i\), \(|c - g(c)|^2 \leq c_O\);

2. for \(c \in (C_L(L_j) \setminus C_O(O_j)) \cap C_o\), \(|c - g(c)|^2 \leq c_L\).

**Proof.** We show how to construct the assignment \(g\) for each client in \(C_L(L_j) \setminus C_O(O_j)\). Consider any client \(c \in C_L(L_j) \setminus C_O(O_j)\). Let \(O_i\) be the subset of \(O\) that contains \(c(O)\). If \(j\) is equal to \(I\), then \(i < j\). Otherwise, there could be two cases: (i) \(i < j\),
and (ii) $i > j$. Consider the case when $i < j$ for $1 \leq j \leq I$. By Lemma 6, there is a point $p \in (Z_j \cap B_j) \cup T_j$ such that $||c - p|| \leq \max\{||c - c(O)||, ||c - c(L)||\}$.

Let $g(c)$ be $p$ in this case. Now consider the case when $i > j$ such that $j < I$. Let $X = L_j \cup O_j \cup Z_j$. By Corollary 3, $d(c, T_j) \leq \max\{d(c, X \setminus B_j), d(c, B_j \cap X)\}$. As $c(L) \in B_j \cap X$, $d(c, B_j \cap X) \leq ||c - c(L)||$. Now note that $c(O) \in X$. Also $c(O) \notin B_j$, as $c(O) \in O_i$ and $i > j$. Thus $c(O) \in X \setminus B_j$ and $d(c, X \setminus B_j) \leq ||c - c(O)||$. Hence $d(c, T_j) \leq \max\{||c - c(O)||, ||c - c(L)||\}$. Let $g(c)$ be the point in $T_j$ nearest to $c$ in this case.

In both cases $||c - g(c)|| \leq \max\{||c - c(L)||, ||c - c(O)||\}$. If $c \in C_i$, $||c - g(c)||^2 \leq ||c - c(O)||^2 = c_O$. Otherwise, $c \in C_o$, and thus $||c - g(c)||^2 \leq ||c - c(L)||^2 = c_L$. Hence the lemma holds.

\[ \Box \]

2.2.5 Approximation Bound

The next lemma gives an upper bound on the quality of the local search solution.

**Lemma 9.** $\text{cost}(L) \leq (1 + O(\varepsilon))\text{cost}(O)$.

**Proof.** Fix an iteration $i$, where $1 \leq i \leq I$. Consider the solution $S_i = (L \setminus L_i) \cup O_i \cup T_i \cup (Z_i \cap B_i)$. By Observation 5, $|L_i \cup O_i \cup T_i \cup (Z_i \cap B_i)| \leq \frac{\beta}{\varepsilon^2}$. Thus $|L \setminus S_i| + |S_i \setminus L| \leq \frac{\beta}{\varepsilon^2}$. By choosing the constant $c$ in Algorithm 2.1 sufficiently large, one can ensure that $\beta \leq c$. Hence due to the local optimality condition in Algorithm 2.1 it follows that,
To argue about the cost of $S_i$ we use an assignment of the clients to the facilities in $S_i$. Consider a client $c$. There can be three cases: (i) $c$ is nearer to a facility of $O_i$ than the facilities in $(O \setminus O_i) \cup (L \setminus L_i)$, that is, $c \in C_O(O_i) \cap (C_o \cup C_L(L_i))$, (ii) $c$ is not in $C_O(O_i)$ and $c$ is nearer to a facility of $L_i$ than the facilities in $L \setminus L_i$, that is, $c \in C_L(L_i) \setminus C_O(O_i)$, (iii) $c$ does not appear in case (i) and (ii), that is, $c$ is not in the union of $C_O(O_i) \cap (C_o \cup C_L(L_i))$ and $C_L(L_i) \setminus C_O(O_i)$. Let $R_i$ be the set of the clients that appear in case (iii). Note that a client cannot appear in both cases (i) and (ii), as the sets $C_O(O_i) \cap (C_o \cup C_L(L_i))$ and $C_L(L_i) \setminus C_O(O_i)$ are disjoint. Also note, that if $c \in C_L(L_i)$, $c$ must appear in case (i) or (ii). Thus if $c$ is corresponding to case (iii), its nearest neighbor $c(L)$ in $L$ must be in $L \setminus L_i$.

Now we describe the assignment. Note that we can assign the clients only to the facilities in $S_i = (L \setminus L_i) \cup O_i \cup T_i \cup (Z_i \cap B_i)$. For a client of type (i), assign it to a facility in $O_i$ nearest to it. For a client of type (ii), use the assignment $g$ in Lemma 8 to assign it to a point in $T_i \cup (Z_i \cap B_i)$. For a client of type (iii), assign it to a facility in $L \setminus L_i$ nearest to it. Thus by Inequality 2.1,

$$
\text{cost}(S_i) \geq (1 - \frac{1}{n})\text{cost}(L) \quad (2.1)
$$

$$
\|(L \setminus L_i) \cup O_i \cup T_i \cup (Z_i \cap B_i)|f + \sum_{c \in C_O(O_i) \cap (C_o \cup C_L(L_i))} c_O + \sum_{c \in C_L(L_i) \setminus C_O(O_i)} ||c - g(c)||^2 + \sum_{c \in R_i} c_L \geq (1 - \frac{1}{n})(|L|f + \sum_{c \in C_L} c_L) \quad (2.2)
$$

By Lemma 8, for a client $c$ in $(C_L(L_i) \setminus C_O(O_i)) \cap C_o$, $||c - g(c)||^2$ is at most $c_L$; for
a client \( c \) in \( (C_L(L_i) \setminus C_O(O_i)) \cap C_i \), \( \|c - g(c)\|^2 \) is at most \( c_O \). It follows that,

\[
|O_i \cup T_i \cup (Z_i \cap B_i)|f + \sum_{c \in C_O(O_i) \cap (C_o \cup C_L(L_i))} (c_O - c_L) + \sum_{c \in (C_L(L_i) \setminus C_O(O_i)) \cap C_i} (c_O - c_L) + \sum_{c \in (C_L(L_i) \setminus C_O(O_i)) \cap C_o} (c_O - c_L) \geq -\frac{1}{n} \text{cost}(L) + |L_i|f \tag{2.3}
\]

\[
\Rightarrow |O_i \cup T_i \cup (Z_i \cap B_i)|f + \sum_{c \in C_O(L_i) \setminus (C_o \cup C_L(L_i))} (c_O - c_L) \geq -\frac{1}{n} \text{cost}(L) + |L_i|f \tag{2.4}
\]

\[
\Rightarrow |O_i \cup T_i \cup (Z_i \cap B_i)|f + \sum_{c \in C_O \cup L \setminus (O_i \cup L_i)} (c_O - c_L) \geq -\frac{1}{n} \text{cost}(L) + |L_i|f
\]

The last inequality follows by noting that the union of \( C_O(O_i) \cap (C_o \cup C_L(L_i)) \) and \( (C_L(L_i) \setminus C_O(O_i)) \cap C_i \) is equal to the set \( C_O \cup L \setminus (O_i \cup L_i) \). Summing over all \( i \) we get,

\[
\sum_{i=1}^{l} |O_i|f + \sum_{i=1}^{l} |T_i \cup (Z_i \cap B_i)|f + \sum_{i=1}^{l} \sum_{c \in C_O \cup L \setminus (O_i \cup L_i)} (c_O - c_L) \geq -O(\varepsilon) \text{cost}(L) + \sum_{i=1}^{l} |L_i|f \tag{2.5}
\]

Note that \( I = O(\varepsilon(|L| + |O|)) = O(\varepsilon n) \) by Observation 5. Now again by Observation 5, \( \sum_{i=1}^{l} |T_i \cup (Z_i \cap B_i)| = O(\varepsilon(|L| + |O|)) \). Hence we get,

\[
|O|f + O(\varepsilon(|L| + |O|))f + \sum_{c \in C} (c_O - c_L) \geq -O(\varepsilon) \text{cost}(L) + |L|f
\]

\[
\Rightarrow \text{cost}(L) \leq (1 + O(\varepsilon)) \text{cost}(O)
\]

This completes the proof of the lemma.
As mentioned before for fixed $d$, the running time of Algorithm 2.1 is polynomial and hence we have established the following theorem.

**Theorem 10.** There is a local search algorithm for SOS-FL which yields a PTAS.

### 2.3 Local Search Algorithm for $k$-means

The local search algorithm for $k$-means and its analysis are very similar to the ones for SOS-FL. Recall that in $k$-means we are given a set $P$ of $n$ points in $\mathcal{R}^d$ and an integer $k > 0$.

#### 2.3.1 The Local Search Algorithm

Fix an $\varepsilon > 0$. The local search algorithm starts with the solution computed by the $9 + \varepsilon$ factor approximation algorithm in [81] (see Algorithm 2.3). Upon termination, the locally optimal solution $K$ has exactly $(1+5\varepsilon)k$ centers. Using an argument similar to the one in case of SOS-FL one can show that this algorithm also runs in polynomial time.

---

**Algorithm 2.3** Local Search

**Require:** A set of points $P \subset \mathcal{R}^d$, an integer $k$, a constant $\varepsilon > 0$.

**Ensure:** A set of centers.

1: $K \leftarrow$ the solution returned by the algorithm in [81]
2: Add arbitrary centers to $K$ to ensure $|K| = (1+5\varepsilon)k$
3: **while** $\exists$ a set $K_1$ s.t. $|K_1| \leq (1+5\varepsilon)k$, $\text{cost}(K_1) < (1 - \frac{1}{n})\text{cost}(K)$ and $|K \setminus K_1| + |K_1 \setminus K| \leq \frac{2\varepsilon}{\varepsilon^2}$ **do**
4: $K \leftarrow K_1$
5: **if** needed, add arbitrary centers to $K$ to ensure $|K| = (1+5\varepsilon)k$
6: **end while**
7: **return** $K$
2.3.2 Analysis of the Local Search Algorithm

Let $L$ be the solution computed by Algorithm 2.3. For the sake of analysis fix an optimal solution $O$. We use the procedure $\text{PARTITION}(L, O, \varepsilon)$ to compute the sets $L_i$, $O_i$, $Z_i \cap B_i$ and $T_i$ for $1 \leq i \leq I$. We use the same $\mu = \frac{\gamma}{\varepsilon^2}$ in this procedure. We note, that Observation 5, Lemma 6, and Lemma 8 hold in this case also, as they directly follow from the $\text{PARTITION}$ procedure, which works on any two input sets of points designated by $L$ and $O$. Let $R_i = L_i \cup O_i \cup T_i \cup (Z_i \cap B_i)$ for $1 \leq i \leq I$. Note, by Observation 5, that $|R_i| \leq \frac{\beta}{\varepsilon^2}$. Also note that $Z_i \cap B_i \subseteq T$ for each $1 \leq i \leq I$, where $T = \bigcup_{i=1}^{I} T_i$. Now we use the following lemma to group the balls returned by $\text{PARTITION}$ into groups of “small” size. This lemma is similar to the Balanced Clustering Lemma in [43].

Lemma 11. Consider the collection $R = \{R_1, \ldots, R_I\}$ of sets with $R_j = L_j \cup O_j \cup T_j \cup (Z_j \cap B_j)$ and $|R_j| \leq \frac{\beta}{\varepsilon^2}$ for $1 \leq j \leq I$, where $\beta$ is the constant in Observation 5. There exists a collection $P = \{P_1, \ldots, P_p\}$, with $P_i \subseteq R$ for $1 \leq i \leq p$, $P_i \cap P_j = \emptyset$ for any $1 \leq i < j \leq p$, and $\bigcup_{i=1}^{p} P_i = R$, which satisfies the following properties:

1. $|P_i| \leq \frac{2\beta}{\varepsilon^2}$ for $1 \leq i \leq p$;

2. $\sum_{R_j \in P_i} |L \cap R_j| \geq \sum_{R_j \in P_i} |(O \cup T) \cap R_j|$ for $1 \leq i \leq p$.

We note that $L \cap R_j = L_j$ and $(O \cup T) \cap R_j = O_j \cup T_j \cup (Z_j \cap B_j)$ for $1 \leq j \leq I$. Before proving the lemma we use it to get an approximation bound on the quality of the local search solution.

Lemma 12. $\text{cost}(L) \leq (1 + O(\varepsilon))\text{cost}(O)$. 
Proof. Consider the collection $\mathcal{P}$ as mentioned in Lemma 11. Also consider any element $J$ of $\mathcal{P}$. Let $L_J = \cup_{R_i \in J}(L \cap R_i)$, $O_J = \cup_{R_i \in J}(O \cap R_i)$, and $T_J = \cup_{R_i \in J}(T \cap R_i)$. Now consider the solution $S_J = (L \setminus L_J) \cup O_J \cup T_J$. By Lemma 11, and using the fact that $L \cap R_i$ and $L \cap R_j$ are disjoint for $i \neq j$, $|L_J| \geq \sum_{R_j \in J} |L \cap R_j| \geq (O \cup T) \cap R_j \geq |O_J \cup T_J|$. Thus by definition of $S_J$, $|S_J| \leq |L| = (1+5\varepsilon)k$. Now $J$ contains at most $\frac{2\beta^2}{\varepsilon}$ sets and thus $|L \setminus S_J| + |S_J \setminus L| \leq |L_J \cup O_J \cup T_J| \leq \frac{2\beta^2}{\varepsilon^2} \leq \frac{2\beta^2}{\varepsilon^2}$. By choosing the constant $c$ in Algorithm 2.3 sufficiently large, one can ensure that $2\beta^2 \leq c$. Hence due to the local optimality condition in Algorithm 2.3 it follows that,

$$\text{cost}(S_J) \geq (1 - \frac{1}{n})\text{cost}(L) \quad (2.6)$$

To argue about the cost of $S_J$ we use an assignment of the clients to the facilities in $S_J$. Consider a client $c$. There can be three cases: (i) $c$ is nearer to a facility of $O_J$ than the facilities in $(O \setminus O_J) \cup (L \setminus L_J)$, that is, $c \in C_1 = C_O(O_J) \cap (C_O \cup C_L(L_J))$, (ii) $c$ is not in $C_O(O_J)$ and $c$ is nearer to a facility of $L_J$ than the facilities in $L \setminus L_J$, that is, $c \in C_2 = C_L(L_J) \setminus C_O(O_J)$, (iii) $c$ does not appear in case (i) and (ii), that is, $c \in C_3 = C \setminus (C_1 \cup C_2)$. Note that if $c \in C_L(L_J)$, $c$ must appear in case (i) or (ii). Thus if $c$ is corresponding to case (iii), its nearest neighbor in $L$ must be in $L \setminus L_J$. Also note that for a client $c \in C_2$, it should be the case that $c(L) \in L_i \subseteq L_J$ and $c(O) \subseteq (O \setminus O_J)$ for some $1 \leq i \neq j \leq I$. Thus Lemma 8 is applicable to $c$, and $g(c) \in T_i \cup (Z_i \cap B_i) \subseteq T_J$.

Now we describe the assignment. For a client in $C_1$, assign it to a facility in $O_J$ nearest to it. For a client $c \in C_2$, use the assignment $g$ in Lemma 8. For a client in $C_3$, assign it to a facility in $L \setminus L_J$ nearest to it. Thus by Inequality 2.6,
\[
\sum_{c \in C_1} c_O + \sum_{c \in C_2} \|c - g(c)\|^2 + \sum_{c \in C_3} c_L \geq (1 - \frac{1}{n}) \sum_{c \in C} c_L
\]

\[
\Rightarrow \sum_{c \in C_1} (c_O - c_L) + \sum_{c \in (C_2 \cap C_o)} (c_L - c_L) + \sum_{c \in (C_2 \cap C_l)} (c_O - c_L) \geq -\frac{1}{n} \text{cost}(\mathcal{L})
\]

\[
\Rightarrow \sum_{c \in C_1} (c_O - c_L) + \sum_{c \in (C_2 \cap C_l)} (c_O - c_L) \geq -\frac{1}{n} \text{cost}(\mathcal{L})
\]

\[
\Rightarrow \sum_{c \in C_{O \cup L}(O_J \cup L_J)} (c_O - c_L) \geq -\frac{1}{n} \text{cost}(\mathcal{L})
\]

The last inequality follows by noting that \(C_1 \cup (C_2 \cap C_l) = C_{O \cup L}(O_J \cup L_J)\). Also by Observation 5, it follows that \(|\mathcal{P}| \leq I = O(\varepsilon(|L| + |O|)) = O(\varepsilon k)\). Thus by summing over all \(J \in \mathcal{P}\) we get,

\[
\sum_{J \in \mathcal{P}} \sum_{c \in C_{O \cup L}(O_J \cup L_J)} (c_O - c_L) \geq -O(\varepsilon) \text{cost}(\mathcal{L})
\]

\[
\Rightarrow \sum_{c \in C} (c_O - c_L) \geq -O(\varepsilon) \text{cost}(\mathcal{L})
\]

\[
\Rightarrow \text{cost}(\mathcal{L}) \leq (1 + O(\varepsilon)) \text{cost}(\mathcal{O})
\]

\(\square\)

As mentioned before for fixed \(d\), the running time of Algorithm 2.1 is polynomial and hence we have established the following theorem.
Theorem 13. There is a polynomial time local search algorithm for \(k\)-means that uses \((1 + O(\varepsilon))k\) facilities and returns a solution with cost at most \((1 + O(\varepsilon))\) times the cost of the optimal \(k\)-means solution.

Next we complete the proof of Lemma 11.

2.3.3 Proof of Lemma 11

For simplicity of exposition we further define some notations. For a set \(r \subseteq L \cup O \cup T\), let \(u(r) = |L \cap r| - |(O \cup T) \cap r|\). For a collection \(\Psi\) of sets, let \(u(\Psi) = \sum_{r \in \Psi} |L \cap r| - |(O \cup T) \cap r|\). Using these notations we rewrite the statement of Lemma 11 as following.

Lemma 14. Consider the collection \(R = \{R_1, \ldots, R_I\}\) of sets with \(R_j = L_j \cup O_j \cup T_j \cup (Z_j \cap B_j)\) and \(|R_j| \leq \frac{\beta}{\varepsilon^2}\) for \(1 \leq j \leq I\), where \(\beta\) is the constant in Observation 5. There exists a collection \(P = \{P_1, \ldots, P_p\}\), with \(P_i \subseteq R\) for \(1 \leq i \leq p\), \(P_i \cap P_j = \emptyset\) for any \(1 \leq i < j \leq p\), and \(\bigcup_{i=1}^{p} P_i = R\), which satisfies the following properties:

1. \(|P_i| \leq \frac{2\beta}{\varepsilon^2}\) for \(1 \leq i \leq p\), where \(\beta\) is the constant in Observation 5;

2. \(u(P_i) \geq 0\) for \(1 \leq i \leq p\).

Proof. Note that for each \(j\), \(u(R_j) \in [-\frac{\beta}{\varepsilon^2}, \frac{\beta}{\varepsilon^2}]\), as \(|R_j| \leq \frac{\beta}{\varepsilon^2}\). Now by Observation 5,

\[\sum_{j=1}^{I} |T_j \cup (Z_j \cap B_j)| \leq \varepsilon(|L| + |O|)/5 \leq \varepsilon((1 + 5\varepsilon)k + k)/5 \leq 2\varepsilon k\].

Thus,

\[u(R) = |L| - |O| - \sum_{j=1}^{I} |T_j \cup (Z_j \cap B_j)| \geq (1 + 5\varepsilon)k - k - 2\varepsilon k \geq 3\varepsilon k\].
Now we show the construction of the collection $\mathcal{P}$. For any $j$, if $u(R_j)$ equals 0, we add $\{R_j\}$ to $\mathcal{P}$ as an element. Note that such an element satisfies the desired properties. Now consider all the sets $R_j \in R$ such that $|u(R_j)| \geq 1$. Denote by $R'$ the collection of such sets. Note that $u(R') = u(R)$. We process $R'$ using the following construction.

The construction is shown as Algorithm 2.4. In each iteration of the outer while loop in line 2, we remove at most $l = \frac{2\beta}{\varepsilon^d}$ sets from $R'$ and add the collection $\Psi$ of these sets to $\mathcal{P}$ as an element. These $l$ sets are chosen carefully so that $u(\Psi)$ is non-negative. To ensure this, at first $l/2$ sets are chosen to get the collection $\Psi'$ such that $-l/2 \leq u(\Psi') \leq l/2$. Then we add at most $l/2$ more sets $\{r\}$ with $u(r) > 0$ (while loop in lines 17-19) to obtain a collection $\Psi$ with $u(\Psi) \geq 0$. Assuming the loop invariant $u(R') \geq 0$ at the beginning of each iteration of the outer while loop, such $r$ must exist. Later we will argue that this loop invariant holds. This ensures that the algorithm exits the while loop in lines 17-19 with a $\Psi$ such that $u(\Psi) \geq 0$. Also we stop the addition of sets as soon as $u(\Psi)$ becomes non-negative. This ensures that $u(\Psi) \leq \frac{\beta}{\varepsilon^d}$. Note that $|\Psi| \leq l = \frac{2\beta}{\varepsilon^d}$. Thus $\Psi$ satisfies all the desired properties.

Now consider the selection of the $l/2$ sets of $\Psi'$. We select these sets sequentially, one in each iteration of the for loop in lines 4-15. Consider a particular iteration of this for loop. There can be two cases: (i) $u(\Psi') \geq 0$, and (ii) $u(\Psi') < 0$. In case (i) if there is a set $r$ with $u(r) < 0$, we choose it. If there is no such set $r$, we add $\Psi'$ to $\mathcal{P}$ and for any set $r \in R'$, $\{r\}$ is added to $\mathcal{P}$ as an element. The algorithm terminates. In case (ii) we choose a set $r$ with $u(r) > 0$. Assuming the loop invariant
Algorithm 2.4

1: $l = \frac{2\beta}{\sqrt{v}}$
2: while $|R'| > l$ do
3: \quad $\Psi' \leftarrow \{r\}$, where $r$ is any element in $R'$ with $u(r) > 0$; $R' \leftarrow R' \setminus \{r\}$
4: \quad for $i = 1$ to $l/2$ do
5: \quad \quad if $u(\Psi') \geq 0$ then
6: \quad \quad \quad if $u(r) > 0$ for each $r \in R'$ then
7: \quad \quad \quad \quad Add $\Psi'$ to $\mathcal{P}$
8: \quad \quad \quad \quad for each $r \in R'$ do
9: \quad \quad \quad \quad \quad Add $\{r\}$ to $\mathcal{P}$; $R' \leftarrow R' \setminus \{r\}$
10: \quad \quad \quad end for
11: \quad \quad return
12: \quad \quad end if
13: \quad r \leftarrow$ any element in $R'$ with $u(r) < 0$
14: \quad $\Psi' \leftarrow \Psi' \cup \{r\}$; $R' \leftarrow R' \setminus \{r\}$
15: \quad else if $u(\Psi') < 0$ then
16: \quad \quad $r \leftarrow$ any element in $R'$ with $u(r) > 0$
17: \quad \quad $\Psi' \leftarrow \Psi' \cup \{r\}$; $R' \leftarrow R' \setminus \{r\}$
18: \quad \quad end if
19: \quad end for
20: \quad $\Psi \leftarrow \Psi'$
21: \quad while $u(\Psi) < 0$ do
22: \quad \quad $r \leftarrow$ any element in $R'$ with $u(r) > 0$
23: \quad \quad $\Psi \leftarrow \Psi \cup \{r\}$; $R' \leftarrow R' \setminus \{r\}$
24: \quad \quad end while
25: \quad Add $\Psi$ to $\mathcal{P}$
26: \quad end while
27: \quad Add $R'$ to $\mathcal{P}$
$u(R') \geq 0$ at the beginning of each iteration of the outer while loop, such an $r$ must exist. Hence in both cases we can ensure that at the end of each iteration of the for loop in lines 4-15 $u(\Psi') \in [-\frac{\beta}{\varepsilon^d}, \frac{\beta}{\varepsilon^d}]$.

Let $M$ denote the number of iterations of the outer while loop. Then in each step except the last, we remove at least $l/2$ sets from $R'$. Since $R'$ has at most $I$ sets initially, $(M - 1) \frac{l}{2} \leq I \Rightarrow M \leq \frac{2I}{l} + 1$.

Now we argue that after iteration $0 \leq j \leq M$,

$$u(R') \geq (\frac{2I}{l} + 1) \frac{l}{2} - j \frac{l}{2}$$  \hspace{1cm} (2.7)

Since $(\frac{2I}{l} + 1) \frac{l}{2} - j \frac{l}{2} \geq (\frac{2I}{l} + 1) \frac{l}{2} - M \frac{l}{2} \geq 0$, this would imply $u(R') \geq 0$ after iteration $j \leq M$. This establishes the loop invariant and also shows that the set $R'$ added to $P$ in line 21 has $u(R') \geq 0$, completing the proof of the lemma.

We now show (2.7) by induction. The inequality is true for $j = 0$, since before iteration 1,

$$u(R') = u(R) \geq 3\varepsilon k \geq I + \frac{\beta}{\varepsilon^d} = (\frac{2I}{l} + 1) \frac{l}{2}$$

Consider a $j$ such that $1 \leq j \leq M$ and suppose (2.7) is true after iteration $j - 1$. Then at the beginning of iteration $j$, we have $u(R') \geq (\frac{2I}{l} + 1) \frac{l}{2} - (j - 1) \frac{l}{2}$. If the condition in line 6 is true in iteration $j$, then the algorithm terminates. Since $R'$ becomes empty after this iteration, (2.7) trivially holds. If in iteration $j$ we add $\Psi$ to $P$ in line 20, then $u(\Psi) \leq \frac{\beta}{\varepsilon^d} = \frac{l}{2}$. Thus, after iteration $j$,
\[ u(R') \geq \left( \frac{2I}{l} + 1 \right) \frac{l}{2} - (j - 1) \frac{l}{2} - \frac{l}{2} = \left( \frac{2I}{l} + 1 \right) \frac{l}{2} - j \frac{l}{2}. \]
CHAPTER 3

K-CLUSTERING

In this chapter, we describe the “bi-criterion” approximation algorithm for $k$-clustering. But before moving on we discuss some background related to the techniques we use, and have some definitions and notations that we are going to use throughout this section.

3.1 Preliminaries

For any subset $Q$ of points in a metric space, the diameter of $Q$, denoted by $\text{diam}(Q)$, is the maximum interpoint distance of $Q$.

**Definition 15.** A $(\beta, \gamma)$ “bi-criterion” approximation algorithm for $k$-clustering is an algorithm that uses at most $\beta k$ balls and yields a solution whose cost is at most $\gamma$ times the cost of an optimal $k$-clustering solution.

We design a $(1+\epsilon, 1+\epsilon)$ “bi-criterion” approximation algorithm for $k$-clustering. The algorithm is based on a probabilistic partitioning scheme that partitions any point set $P$ into blocks (or subsets). Moreover, this partitioning scheme satisfies two properties: (i) for any block $Q$, $\text{diam}(Q) \leq \text{diam}(P)$, and (ii) for any ball $B$ with sufficiently “small” radius, the expected number of blocks that intersect $B$ is “small”. We note that in case the metric space is the real euclidean space, one can use a quadtree based probabilistic partitioning scheme which satisfies the above mentioned properties [26]. The celebrated metric partitioning algorithms of Bartal [21] and Fakcharoenphol, Rao, and Talwar [50] guarantee the first property. In addition,
both of the partitioning schemes ensure that the probability that a small radius ball is intersected by two or more blocks is nicely bounded. In the case of $\alpha = 1$, this bound on the probability is sufficient to get a $(1 + \varepsilon)$ approximation. Thus either of the partitioning algorithms works well [59]. But in the case of $\alpha > 1$, if one uses the partitioning algorithm of [50], the expected number of blocks intersected by such a ball can be quite large, as we explain in Section 3.2.1. However, we argue that using a different partitioning algorithm, the expected number of blocks intersected by a small radius ball is suitably small.

Before describing the result for $k$-clustering, we present a quasi-polynomial time approximation scheme (QPTAS) for the Minimum Cost Covering problem which also uses a similar partitioning scheme, i.e. we present a quasi-polynomial time $(1 + \varepsilon)$-approximation. Lastly, we prove an inapproximability bound of $c \log |X|$ for Minimum Cost Covering problem with $\alpha \geq \log |X|$ assuming $P \neq NP$.

3.2 A QPTAS for Minimum Cost Covering Problem

Recall, that in the Minimum Cost Covering problem (MCC) we are given two point sets $X$ and $Y$, and a metric $d$ on $Z = X \cup Y$. One key property of MCC that we will prove is that in an optimal cover, there are only a small number of balls whose radius is “large”. We can therefore afford to guess these balls by an explicit enumeration. However, there can be a large number of balls with small radius. To help ‘find’ these, we partition the metric space into blocks with at most half the original diameter, and recurse on each block. We have to pay a price for this recursion in the approximation guarantee. This price depends on the number of blocks in the
partition that a small radius ball can intersect.

We are led to the following problem: is there a way to probabilistically partition a metric space into blocks of at most half the diameter, so that for any ball of radius $r$, the expected number of blocks that intersect the ball can be nicely bounded? We use the following partitioning scheme to solve this problem.

### 3.2.1 The Partitioning Scheme

Let $P \subseteq \mathbb{Z}$ be a point set with at least 2 points, and $n \geq |P|$ be a parameter. For $Q \subseteq \mathbb{Z}$, denote the maximum interpoint distance (or diameter) of $Q$ by $\text{diam}(Q)$. In this section, we describe a probabilistic algorithm that partitions $P$ into subsets $\{P_1, P_2, \ldots, P_t\}$ and has the following guarantees:

1. For each $1 \leq i \leq t$, $\text{diam}(P_i) \leq \text{diam}(P)/2$.

2. For any ball $B$ (centered at some point in $\mathbb{Z}$) of radius $r \leq \frac{\text{diam}(P)}{16 \log n}$, the expected size of the set $\{i | P_i \cap B \neq \emptyset\}$ is at most $1 + c\frac{r}{\text{diam}(P)} \log n$, where $c > 0$ is a constant. In other words, the expected number of blocks in the partition that intersect $B$ is at most $1 + c\frac{r}{\text{diam}(P)} \log n$.

### 3.2.2 The Partitioning Scheme of [50]

We first explain why the probabilistic partitioning algorithm of [50] does not achieve this guarantee. In this algorithm, we first pick a $\beta$ uniformly at random from the interval $[\frac{\delta}{8}, \frac{\delta}{4}]$, where $\delta = \text{diam}(P)$. Also, let $\pi_1, \pi_2, \ldots, \pi_p$ be a permutation of $P$ chosen uniformly at random. We compute $P_1, P_2, \ldots, P_p$ in order as follows. Suppose
we already computed $P_1, \ldots, P_{i-1}$. We let

$$P_i = \{x \in P \setminus (P_1 \cup P_2 \cup \cdots \cup P_{i-1}) : d(x, \pi_i) \leq \beta\}.$$ 

We will refer to $P_i$ as $\pi_i$'s cluster. We return the partition $\{P_i \mid P_i \neq \emptyset\}$.

Consider the following weighted tree. Let vertex $u$ be connected to vertices $u_1, u_2, \ldots, u_b$ using edges of weight $\frac{\delta}{16 \log n}$. Here, $n$, $\delta$, and $b$ are parameters. Let $V_1, V_2, \ldots, V_b$ be disjoint sets with $b$ vertices each. For each $i$, $u_i$ is connected to every vertex in $V_i$ using edges of weight $\frac{\delta}{4} - \frac{\delta}{16 \log n}$. Finally, $z$ is a new vertex that is connected to $u$ using an edge of weight $\frac{3\delta}{4}$. Consider the metric induced by this weighted graph, and let $P$ denote the vertex set. That is $P = \{u\} \cup \bigcup_i \{u_i\} \cup \bigcup_i V_i \cup \{z\}$. Set $n := |P|$ and note that $b = \Theta(\sqrt{n})$. Also, $\delta = \text{diam}(P)$.

Let $B = B(u, r)$ where $r = \frac{\delta}{16 \log n}$. Notice that the ball $B$ consists of the points $\{u, u_1, u_2, \ldots, u_b\}$. Consider running the probabilistic partitioning algorithm of [50], described above, on $P$. We argue that the expected number of blocks in the output partition that intersect $B$ is $\Omega(\frac{\sqrt{n}}{\log n})$, which is asymptotically larger than $1 + c\frac{r}{\text{diam}(P)} \log n = O(1)$.

Fix $1 \leq i \leq b$. We observe that in the output partition, $u_i$ belongs to the cluster of some point in $\{u, u_1, u_2, \ldots, u_b\}$ or of some point in $V_i$. Call $u_i$ isolated if $u_i$ belongs to the cluster of some point in $V_i$. If $u_i$ is isolated, the cluster containing $u_i$ does not contain any of the other $u_j$. Thus, the number of blocks of the output partition that intersect $B$ is at least the number of vertices in $\{u_1, u_2, \ldots, u_b\}$ that are isolated.

Note that $u_i$ is isolated if the following two events occur: (a) $\beta \in [\frac{\delta}{4} - \frac{\delta}{16 \log n}, \frac{\delta}{4}]$;
(b) some vertex in \( V_i \) appears in the random permutation \( \pi_1, \pi_2, \ldots, \pi_p \) before all vertices in \( \{u, u_1, \ldots, u_b\} \). The probability of these two events occurring is \( \Omega(\frac{1}{\log n}) \).

It follows that the expected number of isolated vertices, and thus the expected number of blocks in the partition that intersect \( B \), is \( \Omega(\frac{b}{\log n}) = \Omega(\sqrt{n} \log n) \).

### 3.2.3 Probability Distribution

Before describing our partitioning algorithm, we consider a probability distribution that it uses. Given a positive real \( \delta \) and an integer \( k \geq 2 \), the distribution is denoted by \( \text{dist}(\delta, k) \). The probability density function (pdf) \( f \) of \( \text{dist}(\delta, k) \) is the following:

\[
  f(x) = \begin{cases} 
    0 & \text{if } x < \delta/8 \text{ and } x > \delta/4 \\
    \frac{8 \log k}{\delta} \frac{1}{2^i} & \text{if } \frac{\delta}{8} + (i-1)\frac{\delta}{8 \log k} \leq x < \frac{\delta}{8} + i\frac{\delta}{8 \log k} \text{ for } 1 \leq i \leq \log k - 1 \\
    \frac{8 \log k}{\delta} \frac{2^i}{k} & \text{if } \frac{\delta}{4} - \frac{\delta}{8 \log k} \leq x \leq \frac{\delta}{4} 
  \end{cases}
\]

The following observation shows that \( f \) is indeed a density function.

**Observation 16.** \( f \) is a probability density function.

**Proof.** It is sufficient to show that \( f \) satisfies the two properties of density function.

As \( \delta \) and \( k \) are nonnegative it is easy to see that \( f(x) \geq 0 \) for \( x \in (-\infty, +\infty) \). Also,

\[
  \int_{-\infty}^{\infty} f(x) dx = \int_{\frac{\delta}{8}}^{\delta} \frac{8 \log k}{\delta} \frac{1}{2} dx + \int_{\delta + \frac{\delta}{8 \log k}}^{\delta + \frac{2\delta}{8 \log k}} \frac{8 \log k}{\delta} \frac{1}{2^i} dx \\
  + \ldots + \int_{\delta + \frac{(\log k - 1)\delta}{8 \log k}}^{\delta + \frac{2(\log k - 1)\delta}{8 \log k}} \frac{8 \log k}{\delta} \frac{1}{2^{\log k - 1}} dx + \int_{\delta + \frac{\log k - 2}{8 \log k}}^{\delta + \frac{\log k}{8 \log k}} \frac{8 \log k}{\delta} \frac{2^i}{k} dx \\
  = \frac{1}{2} \sum_{i=0}^{\log k - 2} \frac{1}{2^i} + \frac{2}{k} = 1 - \left(\frac{1}{2}\right)^{\log k - 1} + \frac{2}{k} = 1
\]
Consider the interval \([\frac{\delta}{8}, \frac{\delta}{2}]\). Now divide the interval into \(\log k\) subintervals of equal length. The \(i^{th}\) interval for \(1 \leq i \leq \log k - 1\) is defined as \([\frac{\delta}{8} + (i - 1) \frac{\delta}{8\log k}, \frac{\delta}{8} + i \frac{\delta}{8\log k}]\). The last interval is \([\frac{\delta}{4} - \frac{\delta}{8\log k}, \frac{\delta}{4}]\). Denote the \(j^{th}\) interval by \(I_j\) for \(1 \leq j \leq \log k\).

To sample a \(\beta\) according \(\text{dist}(\delta, k)\), we first pick one of these intervals from the distribution that assigns a probability of \(1/2^j\) to \(I_j\) for \(1 \leq j \leq \log k - 1\), and a probability of \(2/k\) to \(I_{\log k}\). Having picked an interval \(I_j\), we generate \(\beta\) uniformly at random from it.

Now we discuss a vital property of the distribution \(\text{dist}(\delta, k)\) which we use in the analysis of the partitioning algorithm. For an event \(E\), let \(\Pr[E]\) denotes the probability that \(E\) occurs. Now consider the following random process. We sample a value \(p\) from \(\text{dist}(\delta, k)\). Let \(E_j\) denotes the event that \(p \in I_j\). Then we have the following observation.

**Observation 17.** \(\Pr[E_j] = \sum_{i=j+1}^{\log k} Pr[E_i]\) for \(1 \leq j \leq \log k - 1\).

**Proof.** The proof follows from the definition of the pdf of \(\text{dist}(\delta, k)\).

\[
\Pr[E_j] = \int_{\frac{\delta}{8} + (j-1) \frac{\delta}{8\log k}}^{\frac{\delta}{8} + j \frac{\delta}{8\log k}} \frac{8\log k}{\delta} \frac{1}{2^j} dx = \frac{1}{2^j} \left( \frac{1}{2} - \frac{2}{k} \right) = \frac{1}{2^j} \left( 1 \right) + \frac{2}{k} \]

\[
= \frac{1}{2^j} \left( 1 - 2 \left( \frac{1}{2^{j+1}} + \frac{1}{2^{j+2}} + \ldots + \frac{1}{2^{\log k-j-1}} \right) \right) + \frac{2}{k}
\]

\[
= \frac{1}{2^j} \left( 1 - 2 \left( \frac{1}{2^j} + \frac{1}{2^{j+1}} + \ldots + \frac{1}{2^{\log k-j-1}} \right) \right) + \frac{2}{k}
\]

\[
= \frac{1}{2^j} \left( 1 - 2 \left( \frac{1}{2^j} + \frac{1}{2^{j+1}} + \ldots + \frac{1}{2^{\log k-j-1}} \right) \right) + \frac{2}{k}
\]

\[
= \Pr[E_{j+1}] + \ldots + \Pr[E_{\log k-1}] + \Pr[E_{\log k}] = \sum_{i=j+1}^{\log k} Pr[E_i]
\]

\(\square\)
3.2.4 Partitioning Algorithm

Now we move on to the partitioning algorithm, which we recall, is given $Z$, a metric $d$ on $Z$, $P \subseteq Z$, and a parameter $n \geq |P|$. The procedure RAND-PARTITION($P$) described below (as Algorithm 3.1) takes a point set $P \subseteq Z$ as input and outputs a partition with at most $|P|$ subsets. Suppose that $P = \{p_1, \ldots, p_{|P|}\}$. The algorithm then generates $P_1, P_2, \ldots, P_{|P|}$ in order via the for loop. Suppose that $P_1, P_2, \ldots, P_{i-1}$ have already been constructed, and $Q = P \setminus (P_1 \cup P_2 \cup \cdots \cup P_{i-1})$. To construct $P_i$, the procedure samples a $\beta_i$ from $\text{dist}(\text{diam}(P), n)$. The choice of $\beta_i$ is done independently of the choices of the other points. Then $P_i$ is set to $\{x \in Q \mid d(x, p_i) \leq \beta_i\}$. Note that this is done in the $i$’th iteration of the for loop. (Note that $p_i$ might not be assigned to $P_i$, as it could already be assigned to some other subset.)

Algorithm 3.1 RAND-PARTITION($P$)

Require: A subset $P = \{p_1, \ldots, p_{|P|}\} \subseteq Z$
Ensure: A partition of $P$
1: $p \leftarrow |P|
2: $Q \leftarrow P$
3: for $i = 1$ to $p$ do
4: sample a $\beta_i$ from $\text{dist}(\text{diam}(P), n)$ corresponding to $p_i$
5: $P_i \leftarrow \{x \in Q \mid d(x, p_i) \leq \beta_i\}$
6: $Q \leftarrow Q \setminus P_i$
7: end for
8: return $\{P_i \mid P_i \neq \emptyset \text{ and } 1 \leq i \leq p\}$

We show that RAND-PARTITION($P$) satisfies the two guarantees mentioned
before. To see that the first guarantee, note that the $\beta$ values are chosen from the distribution $\text{dist}(\text{diam}(P), n)$ which ensures that $\beta_i \leq \text{diam}(P)/4$ for $1 \leq i \leq p$. Now each point in a subset $P_i$ is at a distance at most $\beta_i$ from $p_i$. Thus by triangle inequality $\text{diam}(P_i) \leq \text{diam}(P)/2$. In the next lemma we will show that the second guarantee also holds. Before that we have a definition.

We say that $P_i$ non-terminally (resp. terminally) intersects $B(y, r)$ if $P_i$ intersects $B(y, r)$ and it is not (resp. it is) the last set in the sequence $P_1, P_2, \ldots, P_p$ that intersects $B(y, r)$.

**Lemma 18.** *(Partitioning Lemma)* There is a constant $c$ such that for any ball $B(y, r)$ with $r \leq \frac{\text{diam}(P)}{16 \log n}$, the expected number of blocks in the output of $\text{RAND-PARTITION}(P)$ that intersect $B(y, r)$ is at most $1 + c \frac{r}{\text{diam}(P)} \log n$. Moreover, the expected number of blocks that intersect the ball non-terminally is at most $c \frac{r}{\text{diam}(P)} \log n$.

The intuition for the lemma is as follows. Consider the beginning of the $i$'th iteration of the for loop and assume that ball $B(y, r)$ is not fully contained in the union of the previously constructed blocks $P_1, \ldots, P_{i-1}$. Then, considering the choice of $\beta_i$, the probability that $B(p_i, \beta_i)$ fully contains the ball $B(y, r)$ is nearly as large as the probability that $B(p_i, \beta_i)$ intersects $B(y, r)$. If $B(p_i, \beta_i)$ fully contains the ball $B(y, r)$, then of course none of the blocks $P_{i+1}, P_{i+2}, \ldots, P_p$ intersect $B(y, r)$. We now proceed to the proof.

**Proof.** For a point $x \in Z$ and subset $Q \subseteq Z$, let $d_{\text{min}}(x, Q) = \min_{q \in Q} d(x, q)$ and $d_{\text{max}}(x, Q) = \max_{q \in Q} d(x, q)$. Fix the ball $B(y, r)$ with $r \leq \frac{\text{diam}(P)}{16 \log n}$. For each $1 \leq i \leq p$
Consider the indicator random variable $T_i$ defined as follows:

$$T_i = \begin{cases} 
1 & \text{if } P_i \text{ intersects } B(y, r) \\
0 & \text{otherwise}
\end{cases}$$

Let the random variable $T = \sum_{i=1}^{p} T_i$ be the number of subsets that the ball intersects. Then $E[T] = \sum_{i=1}^{p} E[T_i] = \sum_{i=1}^{p} Pr[P_i \text{ intersects } B(y, r)]$.

Clearly, there is at most one $P_i$ that is the last one that intersects $B(y, r)$.

Thus,

$$\sum_{i=1}^{p} Pr[P_i \text{ intersects } B(y, r)] \leq 1 + \sum_{i=1}^{p} Pr[P_i \text{ non-terminally intersects } B(y, r)].$$

Let $x_i = d_{\min}(p_i, B(y, r))$ and $y_i = d_{\max}(p_i, B(y, r))$. By the triangle inequality, $y_i - x_i \leq 2r$. Denote by $(S_i^j)$ the event that $\beta_i$ lands in the interval $[x_i, y_i]$. Note that for $P_i$ to non-terminally intersect $B(y, r)$, the event $(S_i^j)$ must occur. Thus, if the interval $[x_i, y_i]$ does not intersect the interval $[\frac{\text{diam}(P)}{8}, \frac{\text{diam}(P)}{4}]$, then $Pr[P_i \text{ non-terminally intersects } B(y, r)] = 0$.

We therefore turn to the case where $[x_i, y_i]$ does intersect the interval $[\frac{\text{diam}(P)}{8}, \frac{\text{diam}(P)}{4}]$.

Recall that in defining the probability distribution $\text{dist}(\text{diam}(P), n)$, we have divided the latter interval into $\log n$ subintervals $I_1, I_2, \ldots, I_{\log n}$ of equal length. Denote by $a_l$ the probability

$$Pr[ \text{a random sample drawn from.dist}(\text{diam}(P), n) \text{ belongs to } I_l].$$

For convenience, define $I_{\log n+1} = [\text{diam}(P)/4, \infty)$ and $a_{\log n+1} = 0$.

Let $I_{l_i}$ be the subinterval that contains $x_i$. (In case $x_i < \frac{\text{diam}(P)}{8}$, let $l_i = 1$.) The length of $[x_i, y_i]$ is at most $2r$, $2r \leq \text{diam}(P)/8 \log n$, and the length of each
of the subintervals is \( \text{diam}(P)/8 \log n \). Thus \([x_i, y_i]\) can intersect at most one more subinterval, and this is \( I_{i+1} \). Let \( r_1 \) and \( r_2 \) be the length of \( I_i \cap [x_i, y_i] \) and \( I_{i+1} \cap [x_i, y_i] \) respectively. Note that \( r_1 + r_2 \leq y_i - x_i \leq 2r \).

To bound \( Pr[P_i \text{ non-terminally intersects } B(y, r)] \), we now have two cases. We say that \( p_i \) is far (from the ball \( B(y, r) \)) if \( l_i \in \{ \log n - 1, \log n \} \). We say that \( p_i \) is near if \( 1 \leq l_i \leq \log n - 2 \).

**Case 1: \( p_i \) is far.** In this case \( a_{l_i}, a_{l_i+1} \leq \frac{2}{n} \). Thus

\[
Pr[S^i] \leq Pr[\beta_i \text{ lands in } I_i \cap [x_i, y_i]] + Pr[\beta_i \text{ lands in } I_{i+1} \cap [x_i, y_i]]
\]

\[
\leq \frac{r_1}{\text{diam}(P)/8 \log n} a_{l_i} + \frac{r_2}{\text{diam}(P)/8 \log n} a_{l_i+1}
\]

\[
\leq \frac{2r}{\text{diam}(P)/8 \log n} \cdot \frac{2}{n}
\]

\[
= \frac{32r \log n}{n \cdot \text{diam}(P)}.
\]

Thus, \( Pr[P_i \text{ non-terminally intersects } B(y, r)] \leq Pr[S^i] \leq \frac{32r \log n}{n \cdot \text{diam}(P)} \).

**Case 2: \( p_i \) is near.** For such a \( p_i \) we have the following crucial observation.

**Claim 19.** \( Pr[P_i \text{ non-terminally intersects } B(y, r)] \leq \frac{32r \log n}{\text{diam}(P)} Pr[P_i \text{ terminally intersects } B(y, r)] \).

**Proof.** Suppose that \( P_1, P_2, \ldots, P_{i-1} \) have been chosen and \( B(y, r) \subseteq P_1 \cup P_2 \cup \cdots \cup P_{i-1} \). Conditioned on such a history, we have \( Pr[P_i \text{ non-terminally intersects } B(y, r)] = Pr[P_i \text{ terminally intersects } B(y, r)] = 0 \) and the claimed inequality holds.

Now suppose that \( B(y, r) \setminus (P_1 \cup P_2 \cup \cdots \cup P_{i-1}) \neq \emptyset \). Let us condition on such a history. Then, \( P_i \) terminally intersects \( B(y, r) \) if \( \beta_i \) lands in \( I_{l_i+2} \cup I_{l_i+3} \cup \cdots \cup I_{\log n} \).
Thus, using Observation 17,

\[ Pr[P_i \text{ terminally intersects } B(y, r)] \geq a_{l_i+2} + a_{l_i+3} + \cdots + a_{\log n} = a_{l_i+1}. \]

On the other hand,

\[ Pr[P_i \text{ non-terminally intersects } B(y, r)] \leq Pr[S^i | \beta_i \in I_{l_i} \cup I_{l_i+1}] \cdot Pr[\beta_i \in I_{l_i} \cup I_{l_i+1}] \leq \left( \frac{2}{3} \frac{r_1}{\text{diam}(P)/8 \log n} + \frac{1}{3} \frac{r_2}{\text{diam}(P)/8 \log n} \right) \cdot (a_{l_i} + a_{l_i+1}) \leq \frac{32 \cdot r \log n}{3 \cdot \text{diam}(P)} \cdot 3a_{l_i+1} \leq \frac{32 \cdot r \log n}{\text{diam}(P)} \cdot Pr[P_i \text{ terminally intersects } B(y, r)]. \]

\[ \square \]

Hence the expected number of subsets that intersect \( B(y, r) \) non-terminally is

\[ \leq \sum_{i=1}^{p} Pr[P_i \text{ non-terminally intersects } B(y, r)] \]

\[ \leq \sum_{i: p_i \text{ is far}} Pr[P_i \text{ non-terminally intersects } B(y, r)] + \sum_{i: p_i \text{ is near}} Pr[P_i \text{ non-terminally intersects } B(y, r)] \]

\[ \leq \frac{32r \log n}{\text{diam}(P)} \sum_{i=1}^{p} \frac{1}{n} + \frac{32r \log n}{\text{diam}(P)} \sum_{i=1}^{p} Pr[P_i \text{ terminally intersects } B(y, r)] \]

\[ \leq \frac{c}{\text{diam}(P)} \log n. \]

For the last inequality, we used the fact that \( \sum_{i=1}^{p} Pr[P_i \text{ terminally intersects } B(y, r)] = 1 \), since there is exactly one \( P_i \) that terminally intersects \( B(y, r) \). Putting the two cases together, we have

\[ E[T] \leq 1 + c \frac{r}{\text{diam}(P)} \log n. \]
We conclude by summarizing the result of this section.

**Theorem 20.** Let $Z$ be a point set with an associated metric $d$, let $P \subseteq Z$ be a point set with at least 2 points, and $n \geq |P|$ be a parameter. The probabilistic algorithm $\text{RAND-PARTITION}(P)$ partitions $P$ into blocks $\{P_1, P_2, \ldots, P_t\}$ and has the following guarantees:

1. For each $1 \leq i \leq t$, $\text{diam}(P_i) \leq \text{diam}(P)/2$.

2. There is a constant $c > 0$ so that for any ball $B$ (centered at some point in $Z$) of radius $r \leq \frac{\text{diam}(P)}{16 \log n}$, the expected size of the set $\{i | P_i \cap B \neq \emptyset\}$ is at most $1 + c \frac{r}{\text{diam}(P)} \log n$.

Moreover, for any ball $B$, the expected number of blocks that intersect $B$ non-terminally is at most $c \frac{r}{\text{diam}(P)} \log n$.

### 3.2.5 Algorithm for MCC

We now describe our $(1 + \epsilon)$-factor approximation algorithm for the MCC problem. Recall that we are given a set $X$ of clients, a set $Y$ of servers, and a metric $d$ on $Z = X \cup Y$. We wish to compute a cover for $X$ with minimum cost. Let $m = |Y|$ and $n = |X|$.

For $P \subseteq X$, let $\text{opt}(P)$ denote some optimal cover for $P$. Denote by $\text{cost}(B)$ the cost of a ball $B$ (the $\alpha$-th power of B’s radius) and by $\text{cost}(D)$ the cost $\sum_{B \in D} \text{cost}(B)$ of a set $D$ of balls.
To compute a cover for $P$, our algorithm first guesses the set $Q \subseteq \text{opt}(P)$ consisting of all the large balls in $\text{opt}(P)$. As we note in the structure lemma below, we may assume that the number of large balls in $\text{opt}(P)$ is small. We then use the algorithm of Theorem 20 to partition $P$ into $\{P_1, P_2, \ldots, P_t\}$. For each $1 \leq i \leq t$, we recursively compute a cover for the set $P'_i \subseteq P_i$ of points not covered by $Q$.

To obtain an approximation guarantee for this algorithm, we use the guarantees of Theorem 20. With this overview, we proceed to the structure lemma and a complete description of the algorithm.

### 3.2.6 A Structure Lemma

It is not hard to show that for any $\gamma \geq 1$ and $P \subseteq X$ such that $\text{diam}(P)$ is within a constant factor of $\text{diam}(Z)$, $\text{opt}(P)$ contains at most $(c/\gamma)^\alpha$ balls of radius at least $\text{diam}(P)/\gamma$. Here $c$ is some absolute constant. The following structural lemma extends this fact.

**Lemma 21.** Let $P \subseteq X$, $0 < \lambda < 1$ and $\gamma \geq 1$, and suppose that $\text{opt}(P)$ does not contain any ball of radius greater than or equal to $2\alpha \cdot \text{diam}(P)/\lambda$. Then the number of balls in $\text{opt}(P)$ of radius greater than or equal to $\text{diam}(P)/\gamma$ is at most $c(\lambda, \gamma) := (9\alpha \gamma/\lambda)^\alpha$.

**Proof.** Suppose that $\text{opt}(P)$ does not contain any ball of radius greater than or equal to $2\alpha \cdot \text{diam}(P)/\lambda$. Note that each ball in $\text{opt}(P)$ intersects $P$ and has radius at most $2\alpha \cdot \text{diam}(P)/\lambda$. Thus the point set $\{z \in Z \mid z \in B \text{ for some } B \in \text{opt}(P)\}$ has diameter at most $\text{diam}(P) + 8\alpha \cdot \text{diam}(P)/\lambda \leq 9\alpha \cdot \text{diam}(P)/\lambda$. It follows that there is a ball centered at a point in $Y$, with radius at most $9\alpha \cdot \text{diam}(P)/\lambda$ that contains
Let $\mu$ denote the number of balls in opt($P$) of radius greater than or equal to $\text{diam}(P)/\gamma$. By optimality of opt($P$), we have $\mu(\text{diam}(P)/\gamma)^\alpha \leq (9\alpha \cdot \text{diam}(P)/\lambda)^\alpha$. Thus $\mu \leq (9\alpha \gamma/\lambda)^\alpha$. 

### 3.2.7 The Algorithm

We may assume that the minimum distance between two points in $X$ is 1. Let $L = 1 + \log(\text{diam}(X))$. As we want a $(1+\varepsilon)$-approximation, we fix a parameter $\lambda = \varepsilon/2L$. Let $\gamma = \frac{c \log n}{\lambda}$, where $c$ is the constant in Theorem 20. Denote $\mathcal{D}$ to be the set of balls such that each ball is centered at a point of $y \in Y$ and has radius $r = d(x,y)$ for some $x \in X$. We note that for any $P \subseteq X$, any ball in opt($P$) must belong to this set. Note that $|\mathcal{D}| \leq mn$. Recall that $c(\lambda, \gamma) = (9\alpha \gamma/\lambda)^\alpha$.

With this terminology, the procedure POINT-COVER($P$) described as Algorithm 3.2 returns a cover of $P \subseteq X$. If $|P|$ is smaller than some constant, then the procedure returns an optimal solution by searching all covers with a constant number of balls. In the general case, one candidate solution is the best single ball solution. For the other candidate solutions, the procedure first computes a partition $\{P_1, \ldots, P_\tau\}$ of $P$, using the RAND-PARTITION($P$) procedure. Note that RAND-PARTITION($P$) is called with $Z = X \cup Y$ and $n = |X| \geq |P|$. Then it iterates over all possible subsets of $\mathcal{D}$ of size at most $c(\lambda, \gamma)$ containing balls of radius greater than $\text{diam}(P)/\gamma$. For each such subset $Q$ and $1 \leq i \leq \tau$, it computes the set $P'_i \subseteq P_i$ of points not covered by $Q$. It then makes recursive calls and generates the candidate solution $Q \cup \bigcup_{i=1}^\tau \text{POINT-COVER}(P'_i)$. Note that all the candidate solutions are ac-
tually valid covers for $P$. Among these candidate solutions the algorithm returns the best solution.

**Algorithm 3.2 POINT-COVER($P$)**

**Require:** A subset $P \subseteq X$.

**Ensure:** A cover of the points in $P$.

1: if $|P|$ is smaller than some constant $\kappa$ then
2: return the minimum solution by checking all covers with at most $\kappa$ balls.
3: end if
4: sol $\leftarrow$ the best cover with one ball
5: cost $\leftarrow$ cost(sol)
6: Let $\{P_1, \ldots, P_\tau\}$ be the set of nonempty subsets returned by RAND-PARTITION($P$)
7: Let $\mathcal{B}$ be the set of balls in $\mathcal{D}$ having radius greater than $\frac{\text{diam}(P)}{\gamma}$
8: for each $Q \subseteq \mathcal{B}$ of size at most $c(\lambda, \gamma)$ do
9: for $i = 1$ to $\tau$ do
10: Let $P_i' = \{p \in P_i \mid p \not\in \bigcup_{B \in Q} B\}$
11: end for
12: $Q' \leftarrow Q \cup \bigcup_{i=1}^\tau$ POINT-COVER($P_i'$)
13: if cost($Q'$) < cost then
14: cost $\leftarrow$ cost($Q'$)
15: sol $\leftarrow$ $Q'$
16: end if
17: end for
18: return sol

Our overall algorithm for MCC calls the procedure POINT-COVER($X$) to get a cover of $X$.

### 3.2.8 Approximation Guarantee

For $P \subseteq X$, let level($P$) denote the smallest non-negative integer $i$ such that $\text{diam}(P) < 2^i$. As the minimum interpoint distance in $X$ is 1, level($P$) = 0 if and only if $|P| \leq 1$. Note that level($X$) $\leq L$. 
The following lemma bounds the quality of the approximation of our algorithm.

**Lemma 22.** \(\text{POINT-COVER}(P)\) returns a solution whose expected cost is at most 
\[(1 + \lambda)^l \text{cost}(\text{opt}(P)),\] where \(l = \text{level}(P)\).

**Proof.** We prove this lemma using induction on \(l\). If \(l = 0\), then \(|P| \leq 1\) and \(\text{POINT-COVER}(P)\) returns an optimal solution, whose cost is \(\text{cost}(\text{opt}(P))\). Thus assume that \(l \geq 1\) and the statement is true for subsets having level at most \(l - 1\). Let \(P \subseteq X\) be a point set with \(\text{level}(P) = l\). If \(|P|\) is smaller than the constant threshold \(\kappa\), \(\text{POINT-COVER}(P)\) returns an optimal solution. So we may assume that \(|P|\) is larger than this threshold. We have two cases.

**Case 1:** There is some ball in \(\text{opt}(P)\) whose radius is at least \(2\alpha \cdot \text{diam}(P)/\lambda\). Let \(B\) denote such a ball and \(r(B) \geq 2\alpha \cdot \text{diam}(P)/\lambda\) be its radius. Since \((1 + \lambda/2\alpha)r(B) \geq r(B) + \text{diam}(P)\), the concentric ball of radius \((1 + \lambda/2\alpha)r(B)\) contains \(P\). It follows that there is a cover for \(P\) that consists of a single ball and has cost at most
\[
(1 + \lambda/2\alpha)^\alpha r(B)^\alpha \leq (1 + \lambda)\text{cost}(\text{opt}(P)) \leq (1 + \lambda)^l \text{cost}(\text{opt}(P)).
\]

**Case 2:** There is no ball in \(\text{opt}(P)\) whose radius is at least \(2\alpha \cdot \text{diam}(P)/\lambda\). Let \(Q_0 \subseteq \text{opt}(P)\) contain those balls of radius at least \(\text{diam}(P)/\gamma\). It follows from Lemma 21 that \(|Q_0| \leq c(\lambda, \gamma)\). Thus the algorithm considers a \(Q\) with \(Q = Q_0\). Fix this iteration. Also fix the partition \(\{P_1, \ldots, P_\tau\}\) of \(P\) computed by \(\text{RAND-PARTITION}(P)\).
RAND-PARTITION ensures that $\text{diam}(P_i) \leq \text{diam}(P)/2$ for $1 \leq i \leq \tau$. Thus $\text{diam}(P'_i) \leq \text{diam}(P)/2$ and the level of each $P'_i$ is at most $l - 1$. Hence by induction the expected value of $\text{cost}(\text{POINT-COVER}(P'_i))$ is at most $(1 + \lambda)^{l-1}\text{cost}(\text{opt}(P'_i))$.

Let $S' = \text{opt}(P) \setminus Q_0$. We argue below that the expected value of $\sum_{i=1}^{\tau} \text{cost}(\text{opt}(P'_i))$ is at most $(1 + \lambda)\text{cost}(S')$. Assuming this, we have

$$E[\text{cost}(\mathcal{Q}_0 \cup \bigcup_{i=1}^{\tau} \text{POINT-COVER}(P'_i))] \leq \text{cost}(\mathcal{Q}_0) + (1 + \lambda)^{l-1}E[\sum_{i=1}^{\tau} \text{cost}(\text{opt}(P'_i))]$$

$$\leq \text{cost}(\mathcal{Q}_0) + (1 + \lambda)^l\text{cost}(S')$$

$$\leq (1 + \lambda)^l\text{cost}(\text{opt}(P)).$$

Thus $\text{POINT-COVER}(P)$ returns a solution whose expected cost is at most $(1 + \lambda)^l\text{cost}(\text{opt}(P))$, as desired.

We now argue that the expected value of $\sum_{i=1}^{\tau} \text{cost}(\text{opt}(P'_i))$ is at most $(1 + \lambda)\text{cost}(S')$. Let $B_i$ consist of those balls in $S'$ that intersect $P_i$. For $B \in S'$, let $\mu(B)$ denote the number of blocks in the partition $\{P_1, \ldots, P_{\tau}\}$ that $B$ intersects. Because $B_i$ is a cover for $P'_i$, we have $\text{cost}(\text{opt}(P'_i)) \leq \text{cost}(B_i)$. Thus

$$\sum_{i=1}^{\tau} \text{cost}(\text{opt}(P'_i)) \leq \sum_{i=1}^{\tau} \text{cost}(B_i) = \sum_{B \in S'} \mu(B)\text{cost}(B).$$

By definition of $Q_0$, any ball $B \in S' = \text{opt}(P) \setminus Q_0$ has radius at most $\frac{\text{diam}(P)}{\gamma} = \frac{\lambda \cdot \text{diam}(P)}{c \log n}$, where $c$ is the constant in Theorem 20. We may assume that $c \geq 16$ and hence $\frac{\lambda \cdot \text{diam}(P)}{c \log n} \leq \frac{\text{diam}(P)}{16 \log n}$. Theorem 20 now implies that

$$E[\mu(B)] \leq 1 + \frac{c \cdot r(B) \log n}{\text{diam}(P)} \leq 1 + \frac{c \log n}{\text{diam}(P)} \cdot \frac{\lambda \cdot \text{diam}(P)}{c \log n} = 1 + \lambda.$$
Thus the expected value of $\sum_{i=1}^{\tau} \text{cost}(\text{opt}(P'_i))$ is at most
\[
\sum_{B \in S'} E[\mu(B)] \text{cost}(B) \leq (1 + \lambda) \sum_{B \in S'} \text{cost}(B) = (1 + \lambda)\text{cost}(S'),
\]
as claimed.

We conclude that the expected cost of the cover returned by POINT-COVER($X$) is at most $(1 + \lambda)^L \text{cost}(\text{opt}(X)) \leq (1 + \varepsilon)\text{cost}(\text{opt}(X))$, since $\lambda = \varepsilon/2L$.

Now consider the time complexity of the algorithm. POINT-COVER($P$) makes $(mn)^O(c(\lambda, \gamma))$ direct recursive calls on subsets of diameter at most $\text{diam}(P)/2$. Thus the overall time complexity of POINT-COVER($X$) can be bounded by $(mn)^O(c(\lambda, \gamma)L)$.

Plugging in $\lambda = \varepsilon/2L$, $\gamma = c \log n/\lambda$, and $c(\lambda, \gamma) = (9\alpha \gamma/\lambda)^\alpha$, we conclude

**Theorem 23.** There is an algorithm for MCC that runs in time $(mn)^O((\frac{\alpha L^2 \log n}{\varepsilon})^{\alpha} L)$ and returns a cover whose expected cost is at most $(1 + \varepsilon)$ times the optimal. Here $L$ is 1 plus the logarithm of the aspect ratio of $X$, that is, the ratio of the maximum and minimum interpoint distances in the client set $X$.

Using relatively standard techniques, which we omit here, we can pre-process the input to ensure that the ratio of the maximum and minimum interpoint distances in $X$ is upper bounded by a polynomial in $\frac{mn}{\varepsilon}$. However, this affects the optimal solution by a factor of at most $(1 + \varepsilon)$. After this pre-processing, we have $L = O(\log \frac{mn}{\varepsilon})$. Using the algorithm in Theorem 23 after the pre-processing, we obtain a $(1 + \varepsilon)$ approximation with the quasi-polynomial running time $O(2^{\log^O(1) mn})$. Here the $O(1)$ hides a constant that depends on $\alpha$ and $\varepsilon$. 
3.3 Algorithm for \( k \)-clustering

Recall that in \( k \)-clustering we are given a set \( X \) of points and a metric \( d \) on \( X \). The goal is to select \( k \) balls with minimum cost centered at points of \( X \), whose union contains \( X \). As mentioned before, the constraint on the number of balls that can be used in \( k \)-clustering problem makes it relatively harder than MCC. However, we describe a \((1 + \varepsilon, 1 + \varepsilon)\) bi-criterion approximation algorithm for \( k \)-clustering, which is similar to Algorithm 2 in [59].

For \( P \subseteq X \), let \( \text{opt}(P, \kappa) \) denote an optimal solution of \( \kappa \)-clustering for \( P \). We reuse the notions of \( \text{level}(P) \), \( \text{cost}(B) \) and \( \text{cost}(D) \) from Section 3.2, for a point set \( P \), a ball \( B \), and a set \( D \) of balls, respectively. Let \(|X| = n|\).

To start with we prove a structure lemma for \( k \)-clustering which is same as Lemma 21 adapted for \( k \)-clustering.

**Lemma 24.** Let \( P \subseteq X \), \( \kappa \) is an integer, \( 0 < \lambda < 1 \) and \( \gamma \geq 1 \), and suppose that \( \text{opt}(P, \kappa) \) does not contain any ball of radius greater than or equal to \( 2\alpha \cdot \text{diam}(P)/\lambda \). Then the number of balls in \( \text{opt}(P, \kappa) \) of radius greater than or equal to \( \text{diam}(P)/\gamma \) is at most \( c(\lambda, \gamma) := (9\alpha\gamma/\lambda)^{\alpha} \).

**Proof.** Suppose that \( \text{opt}(P, \kappa) \) does not contain any ball of radius greater than or equal to \( 2\alpha \cdot \text{diam}(P)/\lambda \). Note that each ball in \( \text{opt}(P, \kappa) \) intersects \( P \) and has radius at most \( 2\alpha \cdot \text{diam}(P)/\lambda \). Thus the point set \( \{x \in X \mid x \in B \text{ for some } B \in \text{opt}(P, \kappa)\} \) has diameter at most \( \text{diam}(P) + 8\alpha \cdot \text{diam}(P)/\lambda \leq 9\alpha \cdot \text{diam}(P)/\lambda \). It follows that there is a ball centered at a point of \( X \), with radius at most \( 9\alpha \cdot \text{diam}(P)/\lambda \) that contains \( P \).
Let $\mu$ denote the number of balls in $\text{opt}(P, \kappa)$ of radius greater than or equal to $\frac{\text{diam}(P)}{\gamma}$. By optimality of $\text{opt}(P, \kappa)$, we have $\mu \left( \frac{\text{diam}(P)}{\gamma} \right)^\alpha \leq (9\alpha \cdot \frac{\text{diam}(P)}{\lambda})^\alpha$. Thus $\mu \leq (9\alpha \gamma / \lambda)^\alpha$.

Let $L = 1 + \log(\text{diam}(X))$. We fix a parameter $\lambda = \varepsilon / 6L$. Let $\gamma = \frac{c \log n}{\lambda}$, where $c$ is the constant in Theorem 20. Denote $D$ to be the set of balls such that each ball is centered at a point of $y \in X$ and has radius $r = d(x, y)$ for some $x \in X$. We note that for any $P \subseteq X$, any ball in $\text{opt}(P, \kappa)$ must belong to this set. Note that $|D| \leq n^2$. Recall that $c(\lambda, \gamma) = (9\alpha \gamma / \lambda)^\alpha$.

We design a procedure $\text{CLUSTERING}(\cdot)$ (see Algorithm 3.3) that given a subset $P$ of $X$ and an integer $\kappa \leq k$, returns a set of at most $(1 + 3\lambda)^l \kappa$ balls whose union contains $P$, where $l = \text{level}(P)$. This procedure is similar to $\text{POINT-COVER}(\cdot)$ except in $\text{CLUSTERING}(\cdot)$ we compute the subproblems on $P_i'$ and integer $\kappa_1 \leq \kappa$ using dynamic programming for $1 \leq i \leq \tau$. Next we give the approximation bound on the cost of the solution returned by $\text{CLUSTERING}(P, \kappa)$.

**Lemma 25.** For any $P \subseteq X$ and an integer $1 \leq \kappa \leq k$, the cost of the solution returned by $\text{CLUSTERING}(P, \kappa)$ is at most $(1 + 3\lambda)^l \text{cost}(\text{opt}(P, \kappa))$ with probability at least $1 - \frac{|P| - 1}{n^2}$, where $l = \text{level}(P)$.

**Proof.** We prove this lemma using induction on $l$. If $l = 0$, then $|P| \leq 1$ and $\text{CLUSTERING}(P, \kappa)$ returns an optimal solution, whose cost is $\text{cost}(\text{opt}(P, \kappa))$. Thus assume that $l \geq 1$ and the statement is true for subsets having level at most $l - 1$. If $|P|$ is smaller than the constant threshold $\beta$, $\text{CLUSTERING}(P, \kappa)$ returns an optimal
Algorithm 3.3 CLUSTERING\((P, \kappa)\)

**Require:** A subset \(P \subseteq X\), an integer \(\kappa \leq k\).

**Ensure:** A set of balls whose union contains the points in \(P\).

1: if \(|P|\) is smaller than some constant \(\beta\) then
2:  return the minimum among the solutions with at most \(\kappa\) balls.
3: end if
4: sol ← the best solution with one ball
5: cost ← cost(sol)
6: \(l \leftarrow \text{level}(P)\)
7: for all \(2 \log_3 n\) iterations do
8:  Let \(\{P_1, \ldots, P_\tau\}\) be the set of nonempty subsets returned by RAND-PARTITION\((P)\)
9:  Let \(B\) be the set of balls in \(D\) having radius greater than \(\frac{\text{diam}(P)}{\gamma}\)
10:  for each \(Q \subseteq B\) of size at most \(c(\lambda, \gamma)\) do
11:     for \(i = 1\) to \(\tau\) do
12:        Let \(P'_i = \{p \in P_i \mid p \notin \bigcup_{B \in Q} B\}\)
13:     end for
14:     for each \(1 \leq i \leq \tau\) and \(0 \leq \kappa_1 \leq \kappa\) do
15:        \(\text{cluster}(P'_i, \kappa_1) \leftarrow \text{CLUSTERING}(P'_i, \kappa_1)\)
16:     end for
17:     for \(i = 0\) to \(\tau - 1\) do
18:        \(R_i \leftarrow \bigcup_{j=i+1}^\tau P'_i\)
19:     end for
20:     for all \(i = \tau - 2\) to \(0\) and \(0 \leq \kappa_1 \leq \kappa\) do
21:        \(\text{cluster}(R_i, \kappa_1) \leftarrow \min_{\kappa':0 \leq \kappa' \leq \kappa_1} \text{cost}(\text{cluster}(P'_{i+1}, \kappa') \cup \text{cluster}(R_{i+1}, \kappa_1 - \kappa'))\)
22:     end for
23:  \(Q' \leftarrow Q \cup \text{cluster}(R_0, \kappa - |Q|)\)
24:  if \(|Q'| \leq (1+3\lambda)\kappa\) and \(\text{cost}(Q') < \text{cost}\) then
25:     cost ← \(\text{cost}(Q')\)
26:     sol ← \(Q'\)
27:  end if
28: end for
29: end for
30: return sol
solution. So we may assume that $|P|$ is larger than this threshold. We have two cases.

**Case 1:** There is some ball in $\text{opt}(P, \kappa)$ whose radius is at least $2\alpha \cdot \text{diam}(P)/\lambda$. Let $B$ denote such a ball and $r(B) \geq 2\alpha \cdot \text{diam}(P)/\lambda$ be its radius. Since $(1 + \lambda/2\alpha)r(B) \geq r(B) + \text{diam}(P)$, the concentric ball of radius $(1 + \lambda/2\alpha)r(B)$ contains $P$. It follows that there is a solution for $P$ that consists of a single ball and has cost at most

$$
(1 + \lambda/2\alpha)^{\alpha}r(B)^{\alpha} \leq (1 + \lambda)\text{cost(\text{opt}(P, \kappa))} \leq (1 + 3\lambda)^{l}\text{cost(\text{opt}(P, \kappa))}.
$$

**Case 2:** There is no ball in $\text{opt}(P, \kappa)$ whose radius is at least $2\alpha \cdot \text{diam}(P)/\lambda$. Note that it is sufficient to show that with probability at least $1 - |P| - 1/n^2$, there is a cover of $P$ with at most $(1 + 3\lambda)^{l}\kappa$ balls whose cost is at most $(1 + 3\lambda)^{l}\text{cost(\text{opt}(P, \kappa))}$.

Consider one of the $2\log_3 n$ iterations of CLUSTERING($P, \kappa$). Fix the partition $\{P_1, \ldots, P_\tau\}$ of $P$ computed by RAND-PARTITION($P$) in this iteration. Let $Q_0$ be the set containing the balls of $\text{opt}(P, \kappa)$ with radius at least $\text{diam}(P)/\gamma$. It follows from Lemma 24 that $|Q_0| \leq c(\lambda, \gamma)$. Fix the choice $Q = Q_0$.

RAND-PARTITION() ensures that $\text{diam}(P_i) \leq \text{diam}(P)/2$ for $1 \leq i \leq \tau$. Thus $\text{diam}(P'_i) \leq \text{diam}(P)/2$ and the level of each $P'_i$ is at most $l - 1$. Hence by induction, with probability at least $1 - |P'_i| - 1/n^2$, $\text{cluster}(P'_i, \kappa_1)$ contains at most $(1 + 3\lambda)^{l-1}\kappa_1$ balls and its cost is at most $(1 + 3\lambda)^{l-1}\text{cost(\text{opt}(P'_i, \kappa_1))}$ for $1 \leq \kappa_1 \leq \kappa$.

Let $S' = \text{opt}(P, \kappa) \setminus Q_0$. Thus $|S'| \leq \kappa - |Q_0|$. We note that the union of the balls in $S'$ contains the points in $\bigcup_{i=1}^\tau P'_i$. Let $S'_i \subseteq S'$ be the set of balls that intersect with $P'_i$ and thus the union of balls in $S'_i$ contains $P'_i$, where $1 \leq i \leq \tau$. We note that by induction, with probability at least $1 - |P'_i| - 1/n^2$, $\text{cluster}(P'_i, |S'_i|)$ contains
at most \((1 + 3\lambda)^{l-1}|S'_i|\) balls and its cost is at most \((1 + 3\lambda)^{l-1}\text{cost}(\text{opt}(P'_i, |S'_i|))\) for \(1 \leq i \leq \tau\). The probability that for every \(1 \leq i \leq \tau\), the cost of \text{cluster}(P'_i, |S'_i|)\) is at most \((1 + 3\lambda)^{l-1}\text{cost}(\text{opt}(P'_i, |S'_i|))\) is at least

\[
\prod_{i=1}^{\tau} \frac{|P'_i| - 1}{n^2} \geq \prod_{i=1}^{\tau} \frac{|P_i| - 1}{n^2} \geq 1 - \frac{|P| - 2}{n^2}.
\]

Assuming this first good event happens, it follows by an easy induction on \(i\) that \text{cluster}(R_i, \sum_{j=i+1}^{\tau} |S'_j|)\) covers \(R_i = \bigcup_{j=i+1}^{\tau} P'_j\) with at most \((1 + 3\lambda)^{l-1}\sum_{j=i+1}^{\tau} |S'_j|\) balls and its cost is at most \((1 + 3\lambda)^{l-1}\text{cost}(\text{opt}(R_i, \sum_{j=i+1}^{\tau} |S'_j|))\). Thus \text{cluster}(R_0, \kappa - |Q_0|) = \bigcup_{i=1}^{\tau} \text{cluster}(P'_i, |S'_i|)\) covers \(R_0 = \bigcup_{i=1}^{\tau} P'_i\) with at most \((1 + 3\lambda)^{l-1}\sum_{i=1}^{\tau} |S'_i|\) balls and its cost is at most \((1 + 3\lambda)^{l-1}\text{cost}(\text{opt}(R_0, \sum_{i=1}^{\tau} |S'_i|))\). Hence \(Q_0 \cup \bigcup_{i=1}^{\tau} \text{cluster}(P'_i, |S'_i|)\) covers \(P\). We argue below that the value of \(\sum_{i=1}^{\tau} |S'_i|\) is at most \((1 + 3\lambda)|S'|\) and the value of \(\sum_{i=1}^{\tau} \text{cost}(\text{opt}(P'_i, |S'_i|))\) is at most \((1 + 3\lambda)\text{cost}(S')\) with probability at least 1/3. Then the probability is at least \(1 - \frac{1}{n^2}\) that corresponding to one of the \(2\log_3 n\) iterations the value of \(\sum_{i=1}^{\tau} |S'_i|\) is at most \((1 + 3\lambda)|S'|\) and the value of \(\sum_{i=1}^{\tau} \text{cost}(\text{opt}(P'_i, |S'_i|))\) is at most \((1 + 3\lambda)\text{cost}(S')\). We refer to this as the second good event. Assuming the second good event also occurs, we have

\[
|Q_0 \cup \bigcup_{i=1}^{\tau} \text{cluster}(P'_i, |S'_i|)| \leq |Q_0| + \sum_{i=1}^{\tau} (1+3\lambda)^{l-1}|S'_i| \leq |Q_0| + (1+3\lambda)^{l-1} \sum_{i=1}^{\tau} |S'_i| \leq |Q_0| + (1+3\lambda)^{l}|S'| \leq |Q_0| + (1+3\lambda)^{l}(\kappa - |Q_0|) \leq (1+3\lambda)^{l}\kappa
\]
\[
\text{cost}(Q_0 \cup \bigcup_{i=1}^{\tau} \text{cluster}(P'_i, |S'_i|)) \leq \text{cost}(Q_0) + (1+3\lambda)^l \sum_{i=1}^{\tau} \text{cost}(\text{opt}(P'_i, |S'_i|)) \\
\leq \text{cost}(Q_0) + (1+3\lambda)^l \text{cost}(S') \\
\leq (1+3\lambda)^l \text{cost}(\text{opt}(P, \kappa)).
\]

These happen when both of the good events occur. The corresponding probability is at least

\[
(1 - \frac{|P| - 2}{n^2}) \times (1 - \frac{1}{n^2}) \geq 1 - \frac{|P| - 1}{n^2}.
\]

Thus \(Q_0 \cup \bigcup_{i=1}^{\tau} \text{cluster}(P'_i, |S'_i|)\) is the desired solution.

We now argue that the probability that \(\sum_{i=1}^{\tau} |S'_i|\) is greater than \((1 + 3\lambda)|S'|\) is at most \(1/3\) and the probability that \(\sum_{i=1}^{\tau} \text{cost}(\text{opt}(P'_i, |S'_i|))\) is greater than \((1 + 3\lambda)\text{cost}(S')\) is at most \(1/3\). Then using union bound, the probability that \(\sum_{i=1}^{\tau} |S'_i|\) is at most \((1 + 3\lambda)|S'|\) and \(\sum_{i=1}^{\tau} \text{cost}(\text{opt}(P'_i, |S'_i|))\) is at most \((1 + 3\lambda)\text{cost}(S')\) is at least \(1/3\).

For \(B \in S'\), let \(\mu(B)\) denote the number of blocks in the partition \(\{P_1, \ldots, P_\tau\}\) that intersects \(B\) non-terminally. Because \(S'_i\) is a solution corresponding to \(P'_i\) and \(|S'_i|\), we have \(\text{cost}(\text{opt}(P'_i, |S'_i|)) \leq \text{cost}(S'_i)\). Thus

\[
\sum_{i=1}^{\tau} \text{cost}(\text{opt}(P'_i, |S'_i|)) \leq \sum_{i=1}^{\tau} \text{cost}(S'_i) = \sum_{B \in S'} (1+\mu(B))\text{cost}(B) = \text{cost}(S') + \sum_{B \in S'} \mu(B)\text{cost}(B).
\]

and,

\[
\sum_{i=1}^{\tau} |S'_i| \leq \sum_{B \in S'} (1 + \mu(B)) = |S'| + \sum_{B \in S'} \mu(B).
\]

By definition of \(Q_0\), any ball \(B \in S' = \text{opt}(P, \kappa) \setminus Q_0\) has radius at most \(\frac{\text{diam}(P)}{\gamma} = \frac{\lambda\text{diam}(P)}{c \log n}\), where \(c\) is the constant in Theorem 20. We may assume that
\[ c \geq 16 \text{ and hence } \frac{\lambda \cdot \text{diam}(P)}{c \log n} \leq \frac{\text{diam}(P)}{16 \log n}. \] Theorem 20 now implies that
\[ E[\mu(B)] \leq \frac{c \cdot r(B) \log n}{\text{diam}(P)} \leq \frac{c \log n}{\text{diam}(P)} \cdot \frac{\lambda \cdot \text{diam}(P)}{c \log n} = \lambda. \]

Using linearity of expectation,
\[ E[\sum_{B \in S'} \mu(B)] = \sum_{B \in S'} E[\mu(B)] \leq \lambda \cdot |S'|. \]

Now by Markov’s inequality,
\[ \Pr[\sum_{B \in S'} \mu(B) > 3\lambda \cdot |S'|] \leq 1/3. \]
It follows that,
\[ \Pr[\sum_{i=1}^\tau |S'_i| > (1 + 3\lambda)|S'|] \leq \Pr[|S'| + \sum_{B \in S'} \mu(B) > (1 + 3\lambda)|S'|] \leq \Pr[\sum_{B \in S'} \mu(B) > 3\lambda \cdot |S'|] \leq 1/3. \]
Again using linearity of expectation,
\[ E[\sum_{B \in S'} \mu(B)\text{cost}(B)] = \sum_{B \in S'} E[\mu(B)]\text{cost}(B) \leq \lambda \cdot \text{cost}(S'). \]
Now by Markov’s inequality,
\[ \Pr[\sum_{B \in S'} \mu(B)\text{cost}(B) > 3\lambda \cdot \text{cost}(S')] \leq 1/3. \]
It follows that,
\[ \Pr[\sum_{i=1}^\tau \text{cost}(\text{opt}(P'_i, |S'_i|)) > (1 + 3\lambda)\text{cost}(S')] \leq \Pr[\text{cost}(S') + \sum_{B \in S'} \mu(B)\text{cost}(B) > (1 + 3\lambda)\text{cost}(S')] \leq \Pr[\sum_{B \in S'} \mu(B)\text{cost}(B) > 3\lambda \cdot \text{cost}(S')] \leq 1/3. \]
Since $\lambda = \varepsilon / 6L$, $(1 + 3\lambda)^L \leq 1 + \varepsilon$. Thus we conclude that with probability at least $1 - \frac{1}{n}$, CLUSTERING($X, k$) returns a solution with at most $(1 + \varepsilon)k$ balls whose cost is at most $(1 + \varepsilon)\text{cost}(\text{opt}(X, k))$.

Now consider the time complexity of the algorithm. CLUSTERING($P, \kappa$) makes $n^{O(c(\lambda, \gamma))}$ direct recursive calls on subsets of diameter at most $\text{diam}(P)/2$. Each recursive call takes $O(n\kappa)$ time to process. Thus the overall time complexity of CLUSTERING($X, k$) can be bounded by $n^{O((\lambda, \gamma)L)}$. Plugging in $\lambda = \varepsilon / 6L$, $\gamma = c\log n / \lambda$, and $c(\lambda, \gamma) = (9\alpha \gamma / \lambda)^{\alpha}$, we conclude

**Theorem 26.** There is a randomized algorithm for $k$-clustering that runs in time $n^{O((\frac{n L^2 \log n}{\alpha}L)}$ and with probability at least $1 - \frac{1}{n}$ returns a solution with at most $(1 + \varepsilon)k$ balls whose cost is at most $(1 + \varepsilon)$ times the optimal. Here $L$ is 1 plus the logarithm of the aspect ratio of $X$, that is, the ratio of the maximum and minimum interpoint distances in the client set $X$.

### 3.4 Inapproximability Result

In this section we present an inapproximability result which complements the result in Section 3.2. In particular here we consider the case when $\alpha$ is not a constant. The heart of this result is a reduction from the dominating set problem. Given a graph $G = (V, E)$, a dominating set for $G$ is a subset $V'$ of $V$ such that for any vertex $v \in V \setminus V'$, $v$ is connected to at least one vertex of $V'$ by an edge in $E$. The dominating set problem is defined as follows.

**Dominating Set Problem (DSP)**
INSTANCE: Graph $G = (V, E)$, positive integer $k \leq |V|$.

QUESTION: Is there a dominating set for $G$ of size at most $k$?

The following inapproximability result is proved by Kann [80].

**Theorem 27.** There is a constant $c > 0$ such that there is no polynomial-time $c \log |V|$-factor approximation algorithm for DSP assuming $P \neq NP$.

The following theorem shows an inapproximability bound for MCC when $\alpha \geq \log |X|$.

**Theorem 28.** For $\alpha \geq \log |X|$, no polynomial time algorithm for MCC can achieve an approximation factor better than $c \log |X|$ assuming $P \neq NP$.

**Proof.** To prove this theorem we show a reduction from DSP. Given an instance $(G = (V, E), k)$ of DSP we construct an instance of MCC. The instance of MCC consists of two sets of points $X$ (clients) and $Y$ (servers), and a metric $d$ defined on $X \cup Y$. Let $V = \{v_1, v_2, \ldots, v_n\}$, where $n = |V|$. For each $v_i \in V$, $Y$ contains a point $y_i$ and $X$ contains a point $x_i$. For any point $p \in X \cup Y$, $d(p, p) = 0$. For $i, j \in [n]$, $d(x_i, y_j)$ is 1 if $i = j$ or the edge $(v_i, v_j) \in E$, and $d(x_i, y_j)$ is 3 otherwise. For $i, j \in [n]$ such that $i \neq j$, we set $d(x_i, x_j) = d(y_i, y_j) = 2$.

Consider two nonadjacent vertices $v_i$ and $v_j$. For any $x_t \in X$ such that $t \neq i, j$, $d(x_i, x_t) + d(x_t, y_j) \geq 3$. Similarly, for any $y_t \in Y$ such that $t \neq i, j$, $d(x_i, y_t) + d(y_t, y_j) \geq 3$. Thus $d$ defines a metric. Next we will prove that $G$ has a dominating set of size at most $k$ iff the cost of covering the points in $X$ using the balls around
the points in $Y$ is at most $k$.

Suppose $G$ has a dominating set $J$ of size at most $k$. For each vertex $v_j \in J$, build a radius 1 ball around $y_j$. We return this set of balls $B$ as the solution of MCC. Now consider any point $x_i \in X$. If $v_i \in J$, then $x_i$ is covered by the ball around $y_i$. Otherwise, there must be a vertex $v_j \in J$ such that $(v_i, v_j) \in E$. Then $d(x_i, y_j)$ is 1 and $x_i$ is covered by the ball around $y_j$. Hence $B$ is a valid solution of MCC with cost at most $k$.

Now suppose there is a solution $B$ of MCC with cost at most $k$. If $k > |X|$, then $V$ is a dominating set for $G$ of size $|X| < k$. If $k \leq |X|$, our claim is that the radius of each ball in $B$ is 1. Suppose one of the balls $B$ has a radius more than 1. Then the way the instance of MCC is created the radius should be at least 3. Hence $k \geq 3^\alpha \geq 3^{\log |X|} > |X|$, which is a contradiction. Now consider the set of vertices $J$ corresponding to the centers of balls in $B$. It is not hard to see that $J$ is a dominating set for $G$ of size at most $k$.

Let OPT be the cost of any optimal solution of MCC for the instance $(X, Y, d)$. Then by the properties of this reduction the size of any minimum dominating set for $G$ is OPT. Thus if there is an approximation algorithm for MCC that gives a solution with cost $(c \log |X|) \cdot \text{OPT}$, then using the reduction we can produce a dominating set of size $(c \log |V|) \cdot \text{OPT}$. Then from Theorem 27 it follows that $P = \text{NP}$. This completes the proof of our theorem. \qed
CHAPTER 4
NON-UNIFORM-K-CENTER

In this chapter, we describe the approximation algorithm for NUkC with three radii classes and derive an inapproximability bound for the Euclidean version of the problem. We also discuss a polynomial time algorithm and an inapproximability result pertaining to the NUkC problem under perturbation resilience.

4.1 NUkC with Three Radii Classes

Chakrabarty et al. [33] have used an LP-aware reduction from NUkC to a similar problem in tree metric to obtain an \((O(\log^* n), 8)\) bi-criteria approximation for NUkC. We will use the same reduction for obtaining the result for NUkC with three radii classes. For the sake of completion, below we briefly discuss their reduction.

First, we consider the LP relaxation of NUkC. For each \(p \in P\) and radius \(r_i\), we have a variable \(0 \leq x_{p,i} \leq 1\), which denotes the extent to which a ball with radius \(r_i\) centered at \(p\) is chosen. WLOG we can assume that the optimal dilation is 1.

\[
\forall p \in P, \sum_{i=1}^{t} \sum_{q \in B(p, r_i)} x_{q,i} \geq 1 \quad \text{(NUkC-LP)}
\]

\[
\forall i \in 1, \ldots, t \quad \sum_{p \in P} x_{p,i} \leq k_i
\]

The reduction is to the Resource Minimization for Fire Containment on Trees problem, which is defined as follows.

**Definition 29.** (Resource Minimization for Fire Containment on Trees (RMFC-T)).
Given a rooted tree $T$ with height $t + 1$ and integers $k_1, \ldots, k_t$, the goal is to select a collection of non-root nodes $U$ and the minimum value $\alpha$, such that (a) for every leaf-root path $\pi$, $U$ contains at least one node from $\pi$, (b) $|U \cap L_i| \leq \alpha k_i$ for $1 \leq i \leq t$, where $L_i$ is the layer $i$ nodes of $T$, i.e., the nodes at distance exactly $i$ from the root.

Now, we consider the LP relaxation of RMFC-T. For each node $u$ of $T$, we have the variable $0 \leq y_u \leq 1$, which denotes the extent to which $u$ is selected. Let $L$ denote the set of leaves of $T$. For a leaf $v$, let $P_v$ denote the set of non-root nodes along the path between $v$ and the root.

\[
\begin{align*}
\text{min } \alpha & \quad \text{(RMFC-T-LP)} \\
\forall v \in L, & \quad \sum_{u \in P_v} y_u \geq 1 \\
\forall i \in 1, \ldots, t & \quad \sum_{u \in L_i} y_u \leq \alpha k_i
\end{align*}
\]

The reduction takes an instance $\mathcal{I}$ of NUkC and a feasible solution $x$ to NUkC-LP on $\mathcal{I}$, and constructs a tree instance $\mathcal{T}$ and a feasible solution $y$ to RMFC-T-LP.

The constructed tree contains $t + 1$ layers $L_1, \ldots, L_{t+1}$, which are constructed over $t + 1$ iterations. Set $L_{t+1}$ to contain all the points in $P$. We will use the terms point and node interchangeably. In iteration $i$, where $i$ goes from $t$ down to $1$, we construct $L_i$, which is a subset of $L_{i+1}$. Now, we describe how the points in $L_{i+1}$ are chosen to be included to $L_i$. We choose these points in some number of rounds. In each round, a point $p \in L_{i+1}$ is chosen whose coverage value w.r.t. the levels $i$ to $t$, i.e., $\sum_{j=i}^t \sum_{q \in B(p, r_j)} x_{q,j}$ is minimum. Such a point $p$ is included to $L_i$ and all the points
in \( L_{i+1} \) that are at a distance at most \( 2r_i \) from \( p \) are never chosen to be included to \( L_i \). Indeed, the node \( p \) at \( L_i \) becomes the parent of all these discarded nodes of \( L_{i+1} \).

We also make the node \( p \) at \( L_i \) to be the parent of the node \( p \) at \( L_{i+1} \). The \( y \)-value of \( p \) at \( L_i \) is set to be its coverage value w.r.t. level \( i \), i.e., \( \sum_{q \in B(p,r_i)} x_{q,i} \). The process is repeated until each of the points of \( L_{i+1} \) is either included to \( L_i \) or discarded. Lastly, all the nodes of \( L_1 \) are connected to a single root to complete the construction of the tree. Chakrabarty et al. [33] have shown that the following two lemmas are implied by the reduction.

**Lemma 30.** [33] The solution \( y \) is a feasible solution to RMFC-T-LP on \( T \) with dilation 1.

**Lemma 31.** [33] Suppose there exists a feasible integral solution \( U \) to \( T \) such that for all \( 1 \leq i \leq t \), \( |U \cap L_i| \leq \alpha_i k_i \). Then, there is a solution to the NUkC instance \( I \) that opens, for each \( 1 \leq i \leq t \), at most \( \alpha_i k_i \) balls of radius at most \( 2r_{\geq i} \), where \( r_{\geq i} = r_i + \ldots + r_t \).

Below we will show that given any instance \( T \) of RMFC-T with \( t = 3 \) and a feasible solution \( y \) to RMFC-T-LP on \( T \), how to round \( y \) to get a feasible integral solution \( U \) such that \( |U \cap L_1| \leq k_1 \), and for \( i \in \{2, 3\} \), \( |U \cap L_i| \leq (2 + \epsilon) k_i \), where \( \epsilon \) is a small constant. From Lemma 31, we obtain the following lemma.

**Lemma 32.** There is a polynomial time algorithm for NUkC with three radii classes that opens \( k_1 \) balls of radius at most \( 2(r_1 + r_2 + r_3) \), \( (2 + \epsilon) k_2 \) balls of radius at most \( 2(r_2 + r_3) \), and \( (2 + \epsilon) k_3 \) balls of radius at most \( 2r_3 \).
Now, to optimize the dilation, we apply the following algorithm based on the values of $r_1, r_2$ and $r_3$.

1. ($r_1 \leq 2.19 \cdot r_3$). In this case, we merge all the classes by loosing a dilation of at most 2.19, and then apply the 2-factor approximation algorithm for $k$-center. Thus, the total dilation is $\leq 4.38$.

2. ($r_1 > 2.19 \cdot r_3, r_2 \leq 1.357 \cdot r_3$). We merge the two smallest classes by loosing a dilation of 1.357. Then, we apply the $1 + \sqrt{5}$-factor approximation algorithm for NUkC with two radii classes. Thus, the total dilation is $\leq 1.357 \cdot (1 + \sqrt{5}) = 4.38$.

3. ($r_1 > 2.19 \cdot r_3, r_2 > 1.357 \cdot r_3, r_1 \leq 1.357 \cdot r_2$). We merge the two largest classes by loosing a dilation of 1.357. Then, we apply the $1 + \sqrt{5}$-factor approximation algorithm for NUkC with two radii classes. Thus, the total dilation is $\leq 1.357 \cdot (1 + \sqrt{5}) = 4.38$.

4. ($r_1 > 2.19 \cdot r_3, r_2 > 1.357 \cdot r_3, r_1 > 1.357 \cdot r_2$). We apply the LP aware reduction described above and our rounding scheme. By Lemma 32, the dilation is $\leq (2(r_1 + r_2 + r_3))/r_1 < 2 + (2/1.357) + (2/2.19) = 4.38$.

Hence, we get the following theorem.

**Theorem 33.** There is a 4.38-approximation algorithm for NUkC with three radii classes that opens $k_1$ balls of radius $r_1$ and $(2 + \epsilon)k_i$ balls of radius $r_i$ for $i \in \{2, 3\}$.

Next, we describe our rounding scheme for RMFC-T with $t = 3$. 
4.1.1 The Rounding Scheme

We are given an instance $T$ of RMFC-T with $t = 3$ and a feasible solution $y$ to RMFC-T-LP on $T$. We would like to round $y$ to find a feasible integral solution. In case of $t = 2$, Chakrabarty et al. [33] showed that any feasible fractional solution can be rounded to a feasible integral solution that uses at most $k_2$ nodes of $L_2$ and at most $k_1$ nodes of $L_1$. In particular, the following theorem follows from [33].

**Theorem 34.** Given any instance $T$ of RMFC-T with $t = 2$ and a feasible solution $y$ to RMFC-T-LP on $T$, there is a polynomial time algorithm that rounds $y$ to obtain a feasible integral solution $U$ such that for $i \in \{1, 2\}$, $|U \cap L_i| \leq k_i$.

However, for $t = 3$, it might not always be possible to round a feasible fractional solution to a feasible integral solution by using at most $k_i$ nodes of $L_i$ for all $i$. Consider the example in Figure 4.1. For any choice of one node from $L_1$ and two nodes from $L_2$, it is not possible to choose only three nodes from $L_3$ that satisfies the path constraint for all the leaves. We note that by using one more node in any of the three levels, one can get a feasible integral solution for this example. However, this might not be possible in general. Indeed, later we will show even larger integrality gap examples.

Now, we move on towards our rounding scheme. We will use the following theorem due to Srinivasan [110] to derive our result.

**Theorem 35.** [110] Given $p_1, \ldots, p_l \in [0, 1]$ such that $k = \sum_i p_i$ is an integer, there is a polynomial time algorithm that randomly rounds the $p_i$’s to $X_1, \ldots, X_l \in \{0, 1\}$ satisfying the conditions: (i) $\sum_{i=1}^{l} X_i = k$, and (ii) for $1 \leq i \leq l$, $Pr[X_i = 1] = p_i$. 
In the following, we describe our rounding scheme.

**Randomized Rounding Algorithm.** Let \( u_1, \ldots, u_l \) be the nodes in \( L_1 \). WLOG we can assume that \( \sum_{i=1}^l y_{u_i} = k_1 \). We apply the randomized algorithm in Theorem 35 with \( p_i = y_{u_i} \) to obtain the \( X_1, \ldots, X_l \in \{0,1\} \). For each \( i \) with \( X_i = 1 \), we round the value \( y_{u_i} \) to 1 and set the \( y \)-values of all of its children and grandchildren to 0. For each \( i \) with \( X_i = 0 \), we scale up the \( y \)-values of all of its children and grandchildren by a factor of \( 1/(1 - y_{u_i}) \). Also, we round the value \( y_{u_i} \) to 0. This process ensures that the sum of \( y \)-values along any leaf-root path is 1. Note that we have already rounded the \( y \)-values of the nodes in \( L_1 \) to 0 or 1. To round the values corresponding to the nodes in \( L_2 \) and \( L_3 \), we apply the algorithm in Theorem 34 to levels 2 and 3 of \( T \). Let \( U \) be the set of nodes whose \( y \)-values are 1 after rounding. We return \( U \) as the solution.
**Lemma 36.** With constant probability, the above Randomized Rounding Algorithm returns a feasible integral solution $U$ such that $|U \cap L_1| \leq k_1$, and for $i \in \{2, 3\}$, $|U \cap L_i| \leq (2 + \epsilon)k_i$, where $\epsilon$ is a small constant.

**Proof.** The path constraints are trivially satisfied for the solution $U$ by construction. Also, as $\sum_{i=1}^{t} X_i = \sum_{i=1}^{t} y_{u_i} = k_1$ and we round only those $y_{u_i}$’s to 1 for which $X_i$’s are equal to 1, $|U \cap L_1| \leq k_1$. Now, consider the sum of $y$-values of the nodes in $L_i$ for $i \in \{2, 3\}$ before the application of the algorithm in Theorem 34 for trees with height 2. Note that these sums depend on the values of $X_i$’s. Let $Y_2$ and $Y_3$ be the random variables corresponding to the sum of $y$-values of the nodes in $L_2$ and $L_3$, respectively. First, we show that $\mathbb{E}[Y_2] \leq k_2$ and $\mathbb{E}[Y_3] \leq k_3$. Consider any node $u_i \in L_1$. If $X_i = 1$, the $y$-values of the children of $u_i$ is 0. Otherwise, the sum of $y$-values of its children is $1/(1 - y_{u_i})$ times the sum of $y$-values of its children before scaling. As the $\Pr[X_i = 0] = 1 - y_{u_i}$, the expected sum of $y$-values of the children of $u_i$ is exactly the sum of $y$-values of its children before scaling. Hence, $\mathbb{E}[Y_2] \leq k_2$. Similarly, one can show that $\mathbb{E}[Y_3] \leq k_3$. Then, from Markov’s inequality we have that,

$$\Pr[Y_2 > (2 + \epsilon)k_2] \leq 1/(2 + \epsilon)$$

$$\Pr[Y_3 > (2 + \epsilon)k_3] \leq 1/(2 + \epsilon)$$

Using union bound we have,

$$\Pr[Y_2 \leq (2 + \epsilon)k_2 \text{ and } Y_3 \leq (2 + \epsilon)k_3] \geq \epsilon/(2 + \epsilon)$$

As the algorithm in Theorem 34 does not use any more number of nodes at each level
than what is required, the lemma follows.

\[\]

4.1.2 Integrality Gap Examples

\textbf{Arbitrary Gap at Level 3.} First, we show a class of examples that admit a feasible fractional solution, but any feasible integral solution that uses \(\leq k_1\) and \(\leq k_2\) nodes from \(L_1\) and \(L_2\), respectively, needs \(\geq (l - 1) \cdot k_3\) nodes from \(L_3\), where \(l \geq 2\) is an integer. The example is shown in Figure 4.2. Fix any integer \(l\). The root has \(l\) children \(u_1, \ldots, u_l\). \(u_1\) has \(l k_3\) children each of which has only one child. For \(i \geq 2\), \(u_i\) has \(k_2 + 1 = l\) children each of which has \(l k_3\) children. Now, we describe a feasible fractional solution for this tree. Set \(y\)-value of \(u_i\) to \(1 - 1/l\) for all \(i\). Set \(y\)-values of the grandchildren of \(u_1\) to \(1/l\). Set \(y\)-values of the children of \(u_i\) to \(1/l\) for all \(i\). It is not hard to verify that the path constraints are satisfied. The sum of \(y\)-values at level 1 is \((1 - 1/l) \cdot l = l - 1 = k_1\). The sum of \(y\)-values at level 2 is \((l - 1) \cdot l \cdot (1/l) = k_2\). The sum of \(y\)-values at level 3 is \(l k_3 \cdot (1/l) = k_3\).

![Figure 4.2. An example tree showing a feasible fractional solution.](image)

Now, consider any feasible integral solution that uses \(\leq k_1\) and \(\leq k_2\) nodes
from $L_2$ and $L_3$, respectively. Consider any $u_i$ for $i \geq 2$. If $u_i$ is not chosen, then even after selecting $k_2$ of its children one needs to select $\geq lk_3$ of its grandchildren to satisfy their path constraints. Otherwise, if all of those $u_i$’s are chosen, $u_1$ cannot be selected. In that case, to satisfy the path constraints for all the grandchildren of $u_1$, we need to select $\geq lk_3 - k_2$ nodes from $L_3$. By setting $k_3 \geq k_2$, we get the desired bound.

**Constant Gap at Levels 2 and 3.** Here we show an example that admits a feasible fractional solution, but any feasible integral solution that uses $\leq k_1$ and $< 2k_2$ nodes from $L_1$ and $L_2$, respectively, needs $\geq (2 - \epsilon)k_3$ nodes from $L_3$, where $\epsilon$ is a small constant. This shows that the bound we have obtained using the rounding scheme is almost tight, and thus by using the LP aware technique of [33], one cannot obtain a significantly better bound for NUkC with three radii classes.

The example is shown in Figure 4.3 and is a special case of our previous example when $l = 2$. The root has 2 children $u_1$ and $u_2$. $u_1$ has $2k_3$ children each of which has only one child. $u_2$ has 2 children each of which has $2k_3$ children. Now, we describe a feasible fractional solution for this tree. Set $y$-value of $u_i$ to $1/2$ for all $i$. Set $y$-values of the grandchildren of $u_1$ to $1/2$. Set $y$-values of the children of $u_2$ to $1/2$. It is not hard to verify that this is a feasible solution.

Now, consider any feasible integral solution that uses $\leq k_1 = 1$ and $< 2k_2 = 2$ nodes from $L_2$ and $L_3$, respectively. If $u_2$ is not chosen, then even after selecting 1 of its children one needs to select $\geq 2k_3$ of its grandchildren to satisfy their path constraints. Otherwise, if $u_1$ is not chosen, to satisfy the path constraints for all the
grandchildren of $u_1$, we need to select $\geq 2k_3 - 1 \geq (2 - \epsilon)k_3$ nodes from $L_3$, where $\epsilon$ is a small constant and $k_3 >> 1/\epsilon$. Hence, we get the desired bound.

4.2 Hardness of Approximation
4.2.1 Tree Metric

For the sake of deriving an inapproximability result in tree metric, we use the following decision version of the Resource Minimization for Fire Containment on Trees (RMFC-T) problem: given a rooted tree $T$ and integer $m$, the goal is to find a set $N$ of non-root nodes such that every root-leaf path contains a node from $N$ and for any integer $t \geq 1$, $|N \cap L_t| \leq m$, where $L_t$ is the set of nodes at distance exactly $t$ from the root.

**Theorem 37.** [85] Given a tree $T$ whose leaves are at the same distance from the root, it is \textbf{NP}-hard to distinguish between the following two cases.

- **YES**: There is a solution to the RMFC-T instance with $m = 1$.
- **NO**: There is no solution to the RMFC-T instance with $m = 1$. 

![Figure 4.3. An example tree showing a feasible fractional solution.](image)
Chakrabarty et al. [33] proved that for any constant $c$, NUkC is NP-hard to approximate within a factor of $c$ even in tree metric. We extend this result for any factor $\gamma \leq c^{n^c}$, where $c$ is a constant. In particular, we prove the following theorem.

**Theorem 38.** For any constant $c$ and any $\gamma \leq c^{n^c}$, NUkC is NP-hard to approximate within a factor of $\gamma$ in tree metric.

To prove this theorem we show a reduction from the RMFC-T problem. As mentioned before, the reduction is very similar to the reduction used by Chakrabarty et al. [33]. The construction is as follows. Let $h$ be the height of the tree. We set $P$ to be the leaves of the given tree $T$, i.e., $P = L_h$. For any edge $(u, v)$ of $T$ such that $u \in L_h$ and $v \in L_{h-1}$, assign a weight $(\gamma + 1)/2$ to $(u, v)$. For any edge $(u, v)$ of $T$ such that $u \in L_i$ and $v \in L_{i-1}$ for $i \leq h - 1$, assign a weight $((\gamma + 1)^{h-i+1} - (\gamma + 1)^{h-i})/2$ to $(u, v)$. Then, the distance function $d$ is the shortest-path metric on $P$ induced by the weights of $T$. We set $t = h$, $r_t = 0$ and for any $1 \leq j < t$, $r_j = (\gamma + 1)^{t-j}$. Also, $k_1 = \ldots = k_t = 1$. Now, we have the following observation.

**Observation 39.** For any two leaves $u, u'$ with a common ancestor $v \in L_j$, $d(u, u') \leq r_j$.

**Proof.**

\[
d(u, u') \leq d(u, v) + d(v, u') = ((\gamma + 1)/2 + ((\gamma + 1)^2 - (\gamma + 1))/2 + \ldots + ((\gamma + 1)^{h-j} - (\gamma + 1)^{h-j-1})/2) +
\]
\[
((\gamma + 1)/2 + ((\gamma + 1)^2 - (\gamma + 1))/2 + \ldots + ((\gamma + 1)^{h-j} - (\gamma + 1)^{h-j-1})/2)
\]
\[
= r_j.
\]
We note that the weight of any edge is bounded by $(\gamma + 1)^{h-1} = c^{O(n^c h)}$, and thus can be represented using $O(n^c h)$ number of bits. It follows that the construction can be done in polynomial-time. We denote the constructed instance of NUkC by $I$. For simplicity, we use the terms point and leaf interchangeably. The following lemma completes the proof of Theorem 38.

**Lemma 40.** If $T$ is the “YES” case of Lemma 59, then the optimum dilation of $I$ is 1. If $T$ is the “NO” case of Lemma 59, then the optimum dilation of $I$ is more than $\gamma$.

*Proof.* Let $T$ be a “YES” instance and $N$ be a solution for $T$. We construct a solution for $I$ from $N$ as follows. For any $v \in N$, let $j$ be the integer such that $v \in L_j$. We select a leaf $u$ from the subtree rooted at $v$, and place a ball of radius $r_j$ at $u$. We note that at most 1 ball of radius $r_i$ is selected for all $i$, as $|N \cap L_i| \leq 1$. Now, consider any point $w \in P$. Then, there must be a node $v$ in $N$ along the path between $w$ and the root. Let $v \in L_j$. Now, the way we place the balls there must be a leaf $u$ in the subtree rooted at $v$ such that a ball of radius $r_j$ is opened at $u$. As $v$ is a common ancestor of $u$ and $w$, from Observation 39, it follows that $d(u, w) \leq r_j$. Hence, the ball $B(u, r_j)$ covers $w$.

Now, let $T$ be a “NO” instance and the optimum dilation of $I$ be at most $\gamma$. Consider such a solution $S$ corresponding to the instance $I$. We construct a solution $N$ for RMFC-T on $T$ using $S$ as follows. For any $1 \leq j \leq t$, let $u$ be the point where
the ball (of radius at most $\gamma r_j$) corresponding to $r_j$ is placed. Let $v$ be the ancestor of $u$ that is in $L_j$. We add $v$ to $N$. Note that, as $S$ contains only one ball corresponding to the value $r_i$, $|N \cap L_i| \leq 1$ for all $i$. Now, consider any leaf $w$. We show that $N$ contains a node along the $w$-root path. Let $B$ be a ball in $S$ that covers $w$. Also, let $B$ be corresponding to the value $r_j$ and be centered at the point $u$. Suppose $v$ is the ancestor of $u$ that is in $L_j$. As the radius of the ball at $u$ is at most $\gamma r_j < r_{j-1}$, a point that is not contained in the subtree rooted at $v$ cannot be covered by $B$. Hence, $w$ must be contained in the subtree rooted at $v$, and thus $w$-root path contains $v \in N$. This implies that $N$ is a solution for $T$ corresponding to the “YES” case, and thus $T$ must be a “YES” instance. But, this is a contradiction, and thus the optimum dilation of $I$ must be more than $\gamma$.

4.2.2 Euclidean Metric

Let $X$ and $Y$ be two finite metric spaces with metrics $d$ and $d'$, respectively. Let $f : X \rightarrow Y$ be a map. Then, the contraction of $f$ is defined as,

$$D_c(f) = \max_{x,y \in X} \frac{d(x,y)}{d'(f(x), f(y))}. $$

The expansion of $f$ is similarly defined as,

$$D_e(f) = \max_{x,y \in X} \frac{d'(f(x), f(y))}{d(x,y)}. $$

The distortion of $f$, $D(f) = D_c(f) \cdot D_e(f)$. We need the following theorem due to Gupta [63] for proving the hardness result.

**Theorem 41.** [63] Any weighted tree $T$ with $L$ leaves can be embedded in polynomial-time into $d$-dimensional Euclidean space with $O(dL^{1/(d-1)} \min\{\log L, d\}^{1/2})$ distortion.
Now, we prove the hardness result for the Euclidean metric.

**Theorem 42.** For any constant $\kappa$ and any $\beta \leq \kappa^n$, NUkC is \NP-hard to approximate within a factor of $\beta$ in the Euclidean metric of dimension $d$ for any $d \geq 1$.

**Proof.** Suppose there is a polynomial-time $\beta$-approximation for NUkC in the Euclidean metric for any constant $\kappa$ and any $\beta \leq \kappa^n$. Then, we show that there is a polynomial-time $\gamma$-approximation for NUkC in tree metric for any $\gamma \leq c^n$, where $c$ is a constant. But, by Theorem 38 this is a contradiction, and hence the proof of the theorem follows.

Now, consider a constant $c$ and any $\gamma \leq c^n$. Also, consider any instance of NUkC in the tree metric induced by the weighted tree $T$. We show how to get a $\gamma$-approximate solution for $T$ using the approximation algorithm for the Euclidean metric. Let $\Delta = O(dn^{1/(d-1)} \log n)$. First, we embed the tree $T$ into $d$-dimensional Euclidean space $\mathbb{R}^d$ using the algorithm of Theorem 41. Let $f : T \to \mathbb{R}^d$ be the embedding. Also, let $d$ and $d_f$ denote the tree and the Euclidean metric, respectively. We fix $\beta$ such that $\beta \leq \gamma/\Delta$, and compute a $\beta$-approximate solution $S$ of NUkC for the Euclidean instance. Thereafter, we construct a solution $S'$ for the problem on $T$ from the solution $S$ in the following way. For any node $x$ of $T$, if $S$ contains a ball centered at $f(x)$ with radius $r$, then we add the ball at $x$ of radius $D_c(f) \cdot r$ to $S'$, where $D_c(f)$ is the contraction of $f$. First, we show that the solution $S'$ constructed in this way covers all the nodes of $T$. Consider any node $x$ of $T$. Then, there is a ball in $S$ centered at some point $f(y)$ that covers $f(x)$. Let $r$ be the radius of this ball.
It follows that $S'$ contains the ball $B$ centered at $y$ having radius $D_c(f) \cdot r$. Now,

$$d(x, y) \leq D_c(f) \cdot d_f(f(x), f(y)) \leq D_c(f) \cdot r.$$ 

Hence, the ball $B$ contains $x$, and thus $S'$ is a feasible solution. Now, we show that the dilation $\alpha(S')$ of the balls in $S'$ is at most $\gamma$ times the optimum dilation. To this end, let $OPT$ and $OPT_f$ be the optimum dilation for the tree and the Euclidean instance, respectively. Then, the dilation $\alpha(S')$ is at most $\beta \cdot OPT_f \cdot D_c(f)$. Now, as the distances between the points can get expanded by a factor of at most $D_e(f)$ due to the embedding, $OPT_f \leq D_e(f) \cdot OPT$. Here $D_e(f)$ is the expansion of $f$. Hence,

$$\alpha(S') \leq \beta \cdot D_e(f) \cdot OPT \leq \beta \cdot \Delta \cdot OPT \leq \gamma \cdot OPT.$$ 

This completes the proof of the theorem. \hfill \qed

### 4.3 Perturbation Resilience of NUkC

We denote an instance of NUkC with $t$ radii classes on metric $d$ by $(P, d, t)$. Note that the radii ($r_i$) and multiplicity ($k_j$) parameters remain implicit in this notation. But, references to these parameters will become clear from the context.

A ball with center $p \in P$ and radius $r$, denoted by $B(p, r)$, is the set of points $\{q \in P \mid d(p, q) \leq r\}$. A set of balls covers a set of points if the union of the balls contains all the points. Recall that a feasible placement is a feasible solution of the problem composed of the chosen balls that cover all the input points. A feasible NUkC clustering $\mathcal{C}$ of the input set of points $P$ is a partition $\{C_1, \ldots, C_k\}$, such that there is a feasible placement $\Pi$ with the property that for all $i$, $C_i$ is a subset of a ball in the placement. We say that the clustering $\mathcal{C}$ is induced by the placement $\Pi$. 
The radius of a cluster $C$ w.r.t. any distance function $d$, denoted by $\text{c-radius}(C, d)$, is $\min_{p \in P} \max_{q \in C} d(p, q)$. Note that no ball centered at a point $p \in P$ of radius smaller than $\text{c-radius}(C, d)$ can cover all the points of $C$. For a placement with dilation $\alpha$, a ball with radius $\alpha r_i$ (resp. $< \alpha r_i$ and $\geq \alpha r_i$) is called an $r_i$ (resp. $< r_i$ and $\geq r_i$) -ball.

Consider a metric space $P$ with metric $d : P \times P \to \mathbb{R}_{\geq 0}$. A metric $d_1$ is called a $\psi$-perturbation of $d$ if for any $p, q \in P$, $d(p, q) / \psi \leq d_1(p, q) \leq \psi \cdot d(p, q)$. Moreover, if the distances in $d_1$ are non-increasing, i.e., $d(p, q) / \psi \leq d_1(p, q) \leq d(p, q)$, $d_1$ is called a $\psi$-NI-perturbation. In this paper, all perturbations we consider satisfy the metric properties.

**Definition 43.** An instance $\mathcal{I} = (P, d, t)$ of NUkC is called $\psi$-perturbation-resilient ($\psi$-PR) if for any metric $\psi$-perturbation $d_1$ of $d$, the unique optimal NUkC clustering of $\mathcal{I}' = (P, d_1, t)$ is identical to the unique optimal clustering of $\mathcal{I}$.

We refer to the instance $\mathcal{I}'$ as a $\psi$-perturbed instance of $\mathcal{I}$. In the above definition, if $d_1$ is a $\psi$-NI-perturbation, then the instance $\mathcal{I}$ is called $\psi$-NI-perturbation-resilient ($\psi$-NI-PR). Note that if an instance $\mathcal{I}$ of NUkC is $\psi$-PR, then it is also $\psi$-NI-PR. However, the converse might not be true. We also consider the notion of robust perturbation resilience where the optimal clustering is allowed to be different by a few points when the distances are perturbed. Two clusterings $\mathcal{C} = \{C_1, \ldots, C_k\}$ and $\mathcal{C}' = \{C'_1, \ldots, C'_k\}$ are called $\epsilon$-close if at most $\epsilon n$ points are clustered differently in the two clusterings, i.e., $\min_{f} \sum_{i=1}^{k} |C_i \setminus C'_f(i)| \leq \epsilon n$. 
Definition 44. An instance $\mathcal{I} = (P, d, t)$ of NUkC is called $(\psi, \epsilon)$-perturbation resilient ($(\psi, \epsilon)$-PR) if for any metric $\psi$-perturbation $d_1$ of $d$, the unique optimal NUkC clustering of $\mathcal{I}' = (P, d_1, t)$ is $\epsilon$-close to the unique optimal clustering of $\mathcal{I}$.

Note that if an instance of NUkC is $\psi$-PR, then it is also $(\psi, 0)$-PR. Thus, any hardness result for NUkC under $\psi$-PR trivially follows for NUkC under $(\psi, \epsilon)$-PR.

Now, we have the following simple observations, which will be useful later in proving the properties of the PR instances.

Observation 45. Consider an NUkC instance $\mathcal{I} = (P, d, t)$ that admits a unique optimal clustering $\mathcal{O}$. Let $C$ be any cluster in $\mathcal{O}$. Also, consider an optimal placement $\Pi$ where $C$ is covered by a ball $B$. Then, the following two properties hold.

- The center $p$ of the ball $B$ must belong to $C$.

- For any two points $u, v$ that lie in two different clusters of $\mathcal{O}$, both of $u, v$ cannot be contained in $B$.

Proof. • Suppose $p$ belongs to the cluster $C'$ such that $C \neq C'$. Construct another clustering $\mathcal{O}'$ by selecting all the clusters in $\mathcal{O}$ except $C$ and $C'$, and the clusters $C \cup \{p\}$ and $C' \setminus \{p\}$. It is not hard to see that $\mathcal{O}'$ is also a feasible clustering induced by $\Pi$. As $\Pi$ is an optimal placement, $\mathcal{O}'$ is also an optimal clustering, which contradicts the uniqueness of the optimal clustering of $\mathcal{I}$. Hence, the statement follows.

• Suppose $B$ contains both $u$ and $v$. We construct a new clustering $\mathcal{O}'$, which is identical to $\mathcal{O}$ except, in $\mathcal{O}'$, we move the points $u, v$ to the cluster $C$. Note that
the clustering $\mathcal{O}'$ can be induced by the placement $\Pi$, as the ball $B$ that covers $C \in \mathcal{O}$ also contains $u, v$. Hence, $\mathcal{O}'$ is an optimal clustering for $\mathcal{I}$ different than $\mathcal{O}$, which is a contradiction, and thus the statement follows.

**Observation 46.** Consider an instance $\mathcal{I} = (P, d, t)$ of NUkC with optimal dilation $\alpha$. Suppose $d_1$ is a distance function, such that for any $p, q \in P$, $d_1(p, q) = d(p, q)/\beta$ for $\beta > 1$. Then, the optimal dilation of the instance $\mathcal{I}' = (P, d_1, t)$ is $\alpha/\beta$.

**Proof.** From the definition of $d_1$ it is easy to verify that the optimal dilation of $\mathcal{I}'$ is at most $\alpha/\beta$. Suppose the optimal dilation is $< \alpha/\beta$. Consider an optimal clustering $\mathcal{O}'$ of $\mathcal{I}'$. Let $C_i$ be any cluster in $\mathcal{O}'$ induced by an $r_i$-ball centered at $c_i$. Then, for any $p \in C_i$, $d_1(c_i, p) < (\alpha/\beta)r_i$. It follows that $d(c_i, p) < \alpha \cdot r_i$. Hence, one can construct a clustering from $\mathcal{O}'$ for $\mathcal{I}$ with dilation $< \alpha$. But, this is a contradiction, and hence the observation follows.

Next, we argue that it is correct to assume, the optimal dilation of a $\psi$-PR or a $(\psi, \epsilon)$-PR instance of NUkC is 1. In general case without perturbation resilience, this assumption can be removed by constructing suitable instances of the problem and by applying the algorithm at hand on those instances. But, here we need to be more careful, as the constructed instances need to preserve the perturbation resilience property.

**Lemma 47.** Suppose there is a polynomial time algorithm $\mathcal{A}$ for the NUkC problem with $t$ radii classes under $\psi$-PR (resp. $(\psi, \epsilon)$-PR) with the properties that (i) for an
instance, if there is a feasible placement of balls with dilation 1, \( A \) returns “yes” and a feasible clustering, and (ii) for an instance, if there is no feasible placement of balls with dilation 1, \( A \) returns “no”. Then, the NUkC problem with \( t \) radii classes under \( \psi\text{-PR} \) (resp. \((\psi, \epsilon)\text{-PR}) can be solved in polynomial time.

**Proof.** Consider any instance \( I = (P, d, t) \) of the NUkC problem with \( c \) radii classes under \( \psi\text{-PR} \) (resp. \((\psi, \epsilon)\text{-PR})). Let \( \alpha \) be the optimal dilation. Note that we do not know the value of \( \alpha \). However, as the input metric is finite, there are only polynomial number of guesses for \( \alpha \). We use the following procedure to obtain the optimal clustering for \( I \). In each step, we guess a value \( \alpha' \) for the optimal dilation in the increasing order of the values. We construct a new instance \( I' \) from \( I \) by only changing the radius \( r_i \) to \( \alpha' \cdot r_i \) for all \( i \). Then, we apply the algorithm \( A \) on the constructed instance. If \( A \) returns “no”, we repeat the process with a different guess. Otherwise, the procedure terminates. We return the same clustering returned by \( A \) as the solution for the instance \( I \).

Now, we argue about the correctness of the procedure. First, we claim that \( I' \) is a \( \psi\text{-PR} \) (resp. \((\psi, \epsilon)\text{-PR}) instance. Before proving this claim we discuss its consequences. Note that if there is no feasible solution for \( I \) with dilation \( \alpha' \), then with \( k_i \) balls of radius \( \alpha' \cdot r_i \) for all \( i \) it is not possible to cover the input points. Hence, in this case, for the constructed instance, there is no feasible solution with dilation 1. Thus, the algorithm correctly returns “no” assuming \( I' \) is a \( \psi\text{-PR} \) (resp. \((\psi, \epsilon)\text{-PR}) instance. If there is a feasible solution for \( I \) with dilation \( \alpha' \), then with \( k_i \) balls of radius \( \alpha' \cdot r_i \) for all \( i \) one can cover the input points. Thus, in that case, for the
constructed instance, there is a feasible solution with dilation 1. Hence, $A$ correctly returns “yes” assuming $I'$ is $\psi$-PR (resp. $(\psi, \epsilon)$-PR). Thus, when $\alpha' = \alpha$, $A$ returns “yes” and the returned clustering is optimal for $I$. Now, we prove the claim.

**Claim 48.** $I'$ is a $\psi$-PR (resp. $(\psi, \epsilon)$-PR) instance.

**Proof.** First, we show that the optimal clustering of $I'$ is unique. Note that the optimal dilation of $I'$ is $\alpha/\alpha'$. Suppose optimal clustering of $I'$ is not unique. Then, there are two different clusterings where the points can be covered using $k_i$ balls of radius $(\alpha/\alpha') \cdot \alpha' \cdot r_i = \alpha \cdot r_i$ from each class $i$. It follows that there are two different optimal clusterings for $I$. But, this is a contradiction, and thus the optimal clustering of $I'$ is unique. Note that the optimal clusterings of $I$ and $I'$ are identical. Let $C$ be that clustering. Now, consider any $\psi$-perturbation $d_1$ of the input metric $d$ and the $\psi$ perturbed instance $I'_1$ of $I'$. Let $I_1 = (P, d_1, t)$ be the corresponding $\psi$ perturbed instance of $I$. Also, let $C'_1$ be the optimal clustering of $I'_1$ with dilation $\alpha'_1$. For the sake of contradiction, suppose $C'_1$ is not identical (resp. $\epsilon$-close) to $C$. We argue that $C'_1$ is also an optimal clustering of $I_1$. But, this is a contradiction, as $I_1$ is a $\psi$ perturbed instance of $I$ and $I$ is a $\psi$-PR (resp. $(\psi, \epsilon)$-PR) instance. Now, note that a placement that induces the clustering $C'_1$ of $I'_1$ uses $k_i$ balls of radius $\alpha'_i \cdot \alpha' \cdot r_i$ from each class $i$. Thus, $C'_1$ is a clustering for $I_1$ with dilation $\alpha'_1 \cdot \alpha'$. It is sufficient to argue that this dilation is optimal for $I_1$. Suppose the optimal dilation is $< \alpha'_1 \cdot \alpha'$. Then, using $k_i$ balls of radius $< \alpha'_i \cdot \alpha' \cdot r_i$ from each class $i$ all the points can be covered. Hence, there is a clustering for $I'_1$ with dilation $< \alpha'_1$, which is a contradiction, and hence the claim follows. 

$\square$
Finally, as the number of guesses for $\alpha$ is a polynomial, the procedure terminates in polynomial time.

4.3.1 Properties of Perturbation Resilience

In this section, we show that perturbation resilience imposes useful structure on the optimal solution. We note that similar properties have been proved in the literature in the context of other problems. First, we consider the instances under robust perturbation resilience with $\psi = 3$ and prove a very interesting property of the optimal clustering. Intuitively, we show that the optimal clusters are “well-separated”.

Lemma 49. Consider any optimal placement $\Pi$ for a $(3, \epsilon)$-PR NUkC instance $I = (P, d, t)$ with optimal dilation 1 where the size of each optimal cluster is $> \epsilon \cdot n + 1$. Let $C_1$ and $C_2$ be two clusters induced by two balls of $\Pi$ with radii $r_i$ and $r_j$, respectively with $r_i \geq r_j$. Then, for any $p \in C_1$ and $q \in C_2$, $d(p, q) > r_i$.

Proof. Let $O$ be the optimal clustering of $I$ having dilation $\alpha$. For the sake of contradiction, suppose there are two points $p \in C_1$ and $q \in C_2$ such that $d(p, q) \leq r_i$. We show that there is a 3-perturbation $d_1$ of $d$ such that the optimal clusterings of $I$ and $I' = (P, d_1, t)$ are not $\epsilon$-close. But, this gives a contradiction to the assumption that $I$ is a $(3, \epsilon)$-PR instance, and hence the lemma follows.

We construct the 3-perturbation $d_1$ of $d$ in the following way. Let $B_1 = B(c_1, r_i)$ and $B_2 = B(c_2, r_j)$ be the ball in $\Pi$ that induces $C_1$ and $C_2$, respectively. Then, for any $s \in C_2$, $d(p, s) \leq d(p, q) + d(q, s) \leq r_i + 2r_j \leq 3r_i$. Also, for any $w \in C_1$, $d(p, w) \leq 2r_i$. First, we construct a complete graph $G$ with vertex set equal to $P$ and
for any edge \((u,v)\), its length is defined by the function \(l\) as follows.

\[
    l(u,v) = \begin{cases} 
        3r_i & \text{if } u = p, v \in (C_1 \cup C_2) \setminus \{c_1\} \text{ and } d(u,v) \geq r_i \\
        3 \cdot d(u,v) & \text{otherwise}
    \end{cases}
\]

The distance \(d_1\) is the shortest path metric on \(G\). Note that, as mentioned before, for any \(v \in (C_1 \cup C_2) \setminus \{c_1\}\), \(d(p,v) \leq 3r_i\). Thus, it is not hard to see that, for any \(u,v \in P\), \(d(u,v) \leq d_1(u,v) \leq 3 \cdot d(u,v)\). It follows that \(d_1\) is a metric 3-perturbation of \(d\). Hence, the instance \(\mathcal{I}'\) has the same optimal clustering \(\mathcal{O}\). Next, we prove a claim that the optimal dilation of \(\mathcal{I}'\) is 3.

**Claim 50.** The optimal dilation of \(\mathcal{I}'\) is 3.

**Proof.** As for any \(u,v \in V\), \(d_1(u,v) \leq 3 \cdot d(u,v)\), the optimal dilation of \(\mathcal{I}'\) is at most 3. We prove that this dilation is at least 3. Suppose the dilation is less than 3. Let \(\Pi'\) be any placement with dilation less than 3 that induces the clustering \(\mathcal{O}\) of \(I'\). Then, we show that there is a placement for \(I\) with dilation less than 1. But, this is a contradiction, and hence the claim follows. Consider any cluster \(C \in \mathcal{O}\) other than \(C_1\) and suppose it gets covered by an \(r_t\)-ball \(B = B(w,r)\) in \(\Pi'\). Let \(x\) be any point in \(C\). Now, consider the distance \(d_1\). Let \(\pi\) be any shortest path between \(w\) and \(x\). We claim that \(\pi\) cannot contain the edge \((p,v)\) for any \(v \in (C_1 \cup C_2) \setminus \{c_1\}\). For the sake of contradiction, say \(\pi\) contains \((p,v)\). But, this implies, \(d_1(w,p) \leq d_1(w,x) \leq r\), and thus \(B\) contains \(p\). But, \(p\) belongs to \(C_1\), and thus by the second property in Observation 45, we get a contradiction. Hence, \(\pi\) does not contain \((p,v)\), and thus from the definition of the metric \(d_1\), it follows that \(d_1(w,x) = 3 \cdot d(w,x)\). Thus, a
ball centered at \( w \) and having radius \( r/3 \) can cover the points of \( C \) in \( I \). Now, note that \( r < 3r_i \), and thus \( r/3 < r_i \). Hence, it is sufficient to use an \( r_i \)-ball with less than 1 factor expansion to cover the points of \( C \) in \( I \). In our new placement for \( I \), we use the \( r_i \)-ball \( B(w, r/3) \) corresponding to each such cluster \( C \neq C_1 \).

Now, consider the cluster \( C_1 \) and suppose \( C_1 \) gets covered by an \( r_i \)-ball \( B' = B(w', r') \) in \( \Pi' \). Let \( x', y' \) be any two points in \( C_1 \). Consider the distance \( d_1 \). Let \( \pi' \) be any shortest path between \( x' \) and \( y' \). Now, if \( \pi' \) contains an edge \((p, u)\), for any \( u \in (C_1 \cup C_2) \setminus \{c_1\} \), such that \( d(p, u) \geq r_i \), then \( d_1(x', y') \geq d_1(p, u) \geq 3r_i \).

Otherwise, \( d_1(x', y') = 3 \cdot d(x', y') \). Now, consider a point \( s \) such that \( d_1(w', s) = \max_{v \in C_1} d_1(w', v) \). If \( d_1(w', s) \geq 3r_i \), in the new placement, we cover the cluster \( C_1 \) by the \( r_i \)-ball \( B_1 = B(c_1, r_i) \) with appropriate expansion. As \( r' \geq 3r_i \), \( r_i \leq r'/3 \). Also, \( r' < 3r_i \). Thus, \( r_i < r_i \). Otherwise, \( d_1(w', s) = 3 \cdot d(w', s) \). In that case, in the new placement, we cover the cluster \( C_1 \) by the \( r_i \)-ball \( B(w', r'/3) \). Hence, in both the cases, it is sufficient to use an \( r_i \)-ball with at most \( < 1 \) factor expansion to cover the points of \( C_1 \) in \( I \). Clearly, the dilation of the new placement is less than 1. \( \square \)

Now, we show an optimal clustering \( \mathcal{O}' \) of \( \mathcal{T}' \) that contains exactly \( k \) clusters and is not \( \epsilon \)-close to \( \mathcal{O} \). \( \mathcal{O}' \) contains all the clusters in \( \mathcal{O} \) except \( C_1 \) and \( C_2 \), and the clusters \((C_1 \cup C_2) \setminus \{c_1\}, \{c_1\}\). Note that for any \( s \in (C_1 \cup C_2) \setminus \{c_1\} \), \( d(p, s) \leq 3r_i \). Thus, \((C_1 \cup C_2) \setminus \{c_1\}\) can be covered by a ball of radius \( 3r_i \). It follows that the dilation of \( \mathcal{O}' \) is at most 3. Clearly, the two clusterings \( \mathcal{O} \) and \( \mathcal{O}' \) differ in \( > \epsilon \cdot n \) points, as \( |C_1| > \epsilon \cdot n + 1 \) and \( |C_2| > \epsilon \cdot n + 1 \). This completes the proof of the lemma. \( \square \)
Note that, as a 3-PR instance is also a (3,0)-PR instance, the above lemma trivially follows for 3-PR instances. In the following, we will show that the above mentioned property of the optimal clustering follows even for any 2-PR instance.

**Lemma 51.** Consider any optimal placement $\Pi$ for a 2-PR NUkC instance $I = (P, d, t)$ with optimal dilation $1$. Let $C_1$ and $C_2$ be two clusters induced by two balls of $\Pi$ with radius $r_i$ and $r_j$, respectively, where $r_i \geq r_j$. Then, for any $p \in C_1$ and $q \in C_2$, $d(p, q) > r_i$.

**Proof.** Let $O$ be the optimal clustering induced by the placement $\Pi$. Also, let $B_1$ and $B_2$ be the balls that induce the clusters $C_1$ and $C_2$, respectively. For the sake of contradiction, suppose there exist two points $p \in C_1$, $q \in C_2$ such that $d(p, q) \leq r_i$. The idea is to show that there is a metric $d_1$ that is a 2-perturbation of $d$ such that $I' = (P, d_1, t)$ has a different optimal clustering than $O$. But, this is a contradiction, and thus the lemma follows.

Let $c_t$ be the center of the ball $B_t$ for $t \in \{1, 2\}$. Then, $d(c_1, q) \leq d(c_1, p) + d(p, q) \leq 2r_i$. We define the distance function $d_1$ in the following way. First, we construct the complete graph with vertex set equal to $P$, and for any edge $(u, v)$, its length is defined by the function $l$:

$$l(u, v) = \begin{cases} \min\{d(u, v), r_i\} & \text{if } u = c_1 \text{ and } v = q \\ d(u, v) & \text{otherwise} \end{cases}$$

We note that, for any $u, v$, $d(u, v)/2 \leq l(u, v) \leq d(u, v)$. The distance function $d_1$ is defined by the shortest path distance between any pair of vertices. It is not hard to verify the following observation.
Observation 52. $d_1$ is a metric 2-perturbation of $d$.

Hence, the instance $I' = (P, d_1, t)$ has the same optimal clustering $O$. Next, we prove a claim that the optimal dilation of $I'$ is also 1.

Claim 53. The optimal dilation of $I'$ is 1.

Proof. As for any $u, v \in V$, $d_1(u, v) \leq d(u, v)$, the optimal dilation of $I'$ is at most 1. We prove that this dilation is at least 1. Suppose the dilation is less than 1. Let $\Pi'$ be any placement with dilation less than 1 that induces the clustering $O$ of $I'$. Then, we show that there is a placement for $I$ with dilation less than 1. But, this is a contradiction, and hence the claim follows. Consider any cluster $C \in O$ that gets covered by an $r_t$-ball $B = B(w, r)$ in $\Pi'$. Let $x$ be any point in $C$. Now, consider the distance $d_1$. Let $\pi$ be any shortest path between $w$ and $x$. We claim that $\pi$ cannot contain the edge $(c_1, q)$. For the sake of contradiction, say $\pi$ contains $(c_1, q)$. But, this implies $d_1(w, c_1) \leq d_1(w, x) \leq r$ and $d_1(w, q) \leq d_1(w, x) \leq r$. Thus, $B$ contains both $c_1$ and $q$. Now, by the first property of Observation 45, $c_1$ belongs to $C_1$. Thus, by the second property of Observation 45, we obtain a contradiction. Hence, $\pi$ does not contain $(c_1, q)$. It follows that $d_1(w, x) \geq d(w, x)$. Thus, the radius of the ball needed to cover the points of $C$ in $I$ is at most $r$. Hence, it is sufficient to use an $r_t$-ball with at most $r/r_t < 1$ factor expansion to cover the points of $C$ in $I$. Now, we construct a placement for $I$ by selecting the same balls to cover the clusters that are used in $\Pi'$. Clearly, the dilation of this placement is less than 1. \hfill \Box

Next, we show that there is a different clustering $O'$ of $I'$ with exactly $k$
clusters that achieves the optimal dilation. This gives rise to a contradiction, and thus \( d(p, q) > r_i \). Now, there are two cases. In the first case, \( q \) is the only point in \( C_2 \), and thus \( C_2 \setminus \{q\} \) is empty. In this case, we pick a non-singleton cluster \( C \) from \( O \setminus \{C_1\} \) and choose a point \( s \in C \). Such a cluster exists WLOG. Then, we define \( O' \) to be the set of clusters in \( O \) except \( C, C_1 \) and \( C_2 \), and the clusters \( C_1 \cup \{q\}, \{s\} \) and \( C \setminus \{s\} \). In the second case, \( q \) is not the only point in \( C_2 \), and thus \( C_2 \setminus \{q\} \) is not empty. In this case, \( O' \) is defined to be the set of clusters in \( O \) except \( C_1 \) and \( C_2 \), and the clusters \( C_1 \cup \{q\}, C_2 \setminus \{q\} \). It is not hard to see that \( C_1 \cup \{q\} \) can be covered by the ball \( B(c_1, r_i) \). Also, if \( C_2 \setminus \{q\} \) is not empty, then \( B(c_2, r_j) \) covers the points in \( C_2 \setminus \{q\} \). Hence, in all the cases, it is trivial to verify that the dilation of the new clustering is 1.

Next, we have a lemma that proves that a \( \psi \)-NI-PR instance is also a \( \sqrt{\psi} \)-PR instance.

**Lemma 54.** If \( I = (P, d, t) \) is a \( \psi \)-NI-PR NUkC instance for some \( \psi > 1 \), then \( I \) is also a \( \sqrt{\psi} \)-PR NUkC instance.

**Proof.** Let \( O \) be the optimal clustering of \( I \). For the sake of contradiction, suppose \( I \) is not \( \sqrt{\psi} \)-PR. Thus, there exists a \( \sqrt{\psi} \)-perturbation \( d_1 \) of \( d \) such that the instance \( I_1 = (P, d_1, t) \) has a different optimal clustering \( O' \) from \( O \). Then, we show that there is also a \( \psi \)-NI-perturbation \( d_2 \), such that \( O' \) is the optimal clustering of the instance \( I_2 = (P, d_2, t) \). But, this is a contradiction, and hence the lemma follows. Towards this end, consider the distance \( d_2 \), such that for any \( p, q \in P \), \( d_2(p, q) = d_1(p, q)/\sqrt{\psi} \).
As \(d(p, q)/\sqrt{\psi} \leq d_1(p, q) \leq \sqrt{\psi} \cdot d(p, q), d(p, q)/\psi \leq d_2(p, q) \leq d(p, q)\). Thus, \(d_2\) is a \(\psi\)-NI-perturbation of \(d\). Also, it is not hard to see that \(d_2\) is a metric.

Let \(\alpha_1\) be the optimal dilation of \(I_1\). From Observation 46, it follows that the optimal dilation \(\alpha_2\) of \(I_2\) is \(\alpha_1/\sqrt{\psi}\). Note that it is sufficient to prove that \(O'\) is a clustering of \(I_2\) with dilation \(\alpha_1/\sqrt{\psi}\). Now, consider any optimal placement \(\Pi_1\) corresponding to \(I_1\). We construct a placement \(\Pi_2\) for \(I_2\) from \(\Pi_1\) that induces \(O'\) and has dilation \(\alpha_1/\sqrt{\psi}\). For each ball \(B(p, r)\) in \(\Pi_1\), we add the ball \(B(p, r/\sqrt{\psi})\) to \(\Pi_2\). It is not hard to see that the dilation of \(\Pi_2\) is \(\alpha_1/\sqrt{\psi}\). We show that it is a feasible placement for \(I_2\) that induces \(O'\). Consider any ball \(B(p, r)\) in \(\Pi_1\) that covers the cluster \(C\). Then, for any \(q \in C\), \(d_1(p, q) \leq r\). It follows that \(d_2(p, r) \leq r/\sqrt{\psi}\). Hence, the ball \(B(p, r/\sqrt{\psi})\) covers the cluster \(C\) w.r.t. \(I_2\). Thus, \(\Pi_2\) is a feasible placement for \(I_2\).

4.3.2 NUkC with a Constant Number of Radii Classes

In this section, we show a polynomial time reduction from NUkC to the Constrained Resource Minimization for Fire Containment on Trees problem, which is a variant of the firefighter problem.

**Definition 55.** (Constrained Resource Minimization for Fire Containment on Trees (CRMFC-T)). Given a rooted tree \(T = (V, E)\) with height \(t + 1\), a set of forbidden nodes \(F \subseteq V\), and integers \(k_1, \ldots, k_t\), the goal is to decide if there is a collection of non-root nodes \(U \subseteq (V \setminus F)\) such that (a) for every leaf-root path \(\pi\), \(U\) contains at least one node from \(\pi\), and (b) \(|U \cap L_i| \leq k_i\) for \(1 \leq i \leq t\), where \(L_i\) is the layer \(i\) nodes of \(T\), i.e., the nodes at distance exactly \(i\) from the root.
Given any instance $\mathcal{I} = (P, d, t)$ of NUkC under 2-PR or $(3, \epsilon)$-PR, we will show how to construct an instance $\mathcal{I}'$ of CRMFC-T such that $\mathcal{I}$ has a feasible placement with dilation 1 iff $\mathcal{I}'$ has a feasible solution. In the constructed instance $\mathcal{I}'$, the height of the tree is one more than the number of radii classes in NUkC. We show that CRMFC-T can be solved in polynomial time if the height of the input tree is a constant. It follows that the perturbation resilient version of NUkC can be solved in polynomial time if the number of classes is a constant. Thus, we obtain the following theorem.

**Theorem 56.** NUkC under 2-PR (resp. $(3, \epsilon)$-PR) can be solved in polynomial time if the number of radii classes is a constant.

Next, we show the reduction to the CRMFC-T problem.

### 4.3.2.1 Tree Construction

Let $G$ be the complete graph that defines the distances between the input points. Note that we are also given the input radii $r_1 > r_2 > \ldots > r_t$. We construct the tree $T$ in $t$ rounds that contains $t$ levels other than the root level. We denote the nodes at level $i$ by $L_i$ for $i \in \{0, \ldots, t\}$. $L_0$ contains a singleton node – the root of the tree. For $i \geq 1$, in $i^{th}$ round, we construct the nodes $L_i$ and connect them with the nodes in $L_{i-1}$. Each node $v$ in $T$ corresponds to a connected subgraph $G_v$ of $G$. The root corresponds to $G$ itself. Also, each node is marked with either yes or no denoting if the node can be selected or it is in the forbidden set.

For each index $i \in \{1, \ldots, t\}$, in $i^{th}$ round, we consider all the nodes $v \in L_{i-1}$ and the subgraph $G_v$ corresponding to $v$. We remove all the edges with weight more
than \( r_i \) from \( G_v \). Let \( G^1_v, \ldots, G^l_v \) be the connected components formed from \( G_v \) due to the removal of these edges. We add \( l \) children of \( v \) to \( L_i \) corresponding to these connected \( l \) subgraphs. For each such child \( u \), if there is a node \( w \) in \( G_u \), such that for all node \( x \) in \( G_u \), \( d(w, x) \leq r_i \), we label \( u \) with yes. Otherwise, we label \( u \) with no (forbidden). Lastly, for each level \( i \geq 1 \), the quantity number of nodes that can be chosen from \( L_i \) in CRMFC-T is set to \( k_i \). The following lemma establishes the connection between the two instances \( \mathcal{I} \) and \( \mathcal{I}' \).

**Lemma 57.** \( \mathcal{I} \) has a feasible placement with dilation 1 iff \( \mathcal{I}' \) has a feasible solution to CRMFC-T.

**Proof.** First, suppose there is a feasible solution to \( \mathcal{I}' \). For each chosen node \( v \), \( v \) must be a yes node. Let \( i \) be the integer such that \( v \in L_i \). Then, the points in \( G_v \) can be covered by an \( r_i \) ball centered at some point in \( G_v \). We choose this ball in our placement. Note that we select at most \( k_i \) balls of radius \( r_i \) for all \( i \). We prove that each point is covered in the constructed placement. Consider any point \( p \). The way we construct the tree, each point can lie in the connected subgraph \( G_v \) of exactly one node \( v \) of \( L_j \) for all \( j \). Let \( \pi \) be the root-leaf path in \( T \), such that for any \( v \in \pi \), \( p \) is in \( G_v \). Now, there must be a node along \( \pi \) that is chosen in the solution of CRMFC-T. Let \( u \) be such a node. As we place a ball of radius \( r_i \) that covers all the points of \( G_u \), \( p \) gets covered. Thus, \( \mathcal{I} \) has a feasible placement with dilation 1.

Now, suppose \( \mathcal{I} \) has a feasible placement with dilation 1. Let \( \mathcal{O} \) be the clustering induced by the placement. Now, consider any cluster \( C \in \mathcal{O} \), which is covered by a ball of radius \( r_j \). Thus, \( c\text{-radius}(C, d) \leq r_j \). The way the tree \( T \) is constructed it fol-
lows that all the points in $C$ remain in the same connected subgraph $G_v$ corresponding to a unique vertex $v \in L_i$ for each $i \leq j$. Let $G_u$ be the subgraph corresponding to level $j - 1$. As $\mathcal{I}$ is a 2-PR (resp. $(3, \epsilon)$-PR) instance, from Lemma 51 (resp. Lemma 49), we know that, for any $p \in C$ and $q \in P \setminus C$, $d(p, q) > r_j$. Thus, when the edges with weight more than $r_j$ are removed from $G_u$, $p$ and $q$ cannot remain in the same component. But, as $c$-radius$(C, d) \leq r_j$ all the points of $C$ remain in the same component. Also, by the first property of Observation 45, the center of the $r_j$-ball that covers $C$ must lie in $C$. It follows that there is a yes node $C(v) \in L_j$ such that $G_{C(v)}$ contains only the points of $C$ as vertices. For each cluster $C \in \mathcal{O}$, we select the yes node $C(v)$ in the solution to CRMFC-T. It is not hard to see that we choose at most $k_j$ nodes from $L_j$. Now, consider any root-leaf path $\pi$ in $T$ corresponding to a leaf $l$. Let $p$ be a point in $G_l$. Also, let $p$ is in the cluster $C \in \mathcal{O}$. Then, there must be a yes node $C(v)$ in $\pi$ such that $G_{C(v)}$ contains only the points of $C$. As we choose $v$ in our solution, we have at least one node from the path $\pi$. Hence, the constructed solution is feasible.

\[\square\]

4.3.2.2 The Algorithm for CRMFC-T

In this section, we design a dynamic programming based algorithm that decides the feasibility of any instance of CRMFC-T. The algorithm runs in polynomial time when the height of the tree is a constant. Let $T$ be the input tree having height $t$, i.e., $T$ has $t + 1$ levels $L_0, \ldots, L_t$. $L_0$ contains only the root of $T$. Let $n_i = |L_i|$. We also assume that the nodes of $L_i$ are ordered for all $i \geq 1$, i.e., $L_i = \{v_{i1}, \ldots, v_{in_i}\}$. For $j \leq l$, let $F(i, j, l)$ be the union of the induced subtrees of $T$ rooted at the vertices
We construct the tree $T(i, j, l)$ from $F(i, j, l)$ by connecting the roots of the subtrees to a common root.

Let $\text{feasible}(T(i, j, l), l_i, l_i+1, \ldots, l_t)$ be the function that decides if there is a feasible solution to CRMFC-T for the tree $T(i, j, l)$ by selecting at most $l_m$ nodes from level $m$, where $i \leq m \leq t$. Note that computing the function $\text{feasible}(T = T(1, 1, n_1), k_1, \ldots, k_t)$ solves the CRMFC-T problem. We consider the following recursive definition of $\text{feasible}()$. In the base case, if $i = t - 1$, the function can be computed in polynomial time. Otherwise, if $l_i$ is 0, let $j'$ be the minimum index such that $v_{i+1,j'}$ is a child of $v_{ij}$ and $l'$ be the maximum index such that $v_{i+1,l'}$ is a child of $v_{il}$. In this case, $\text{feasible}(T(i, j, l), l_i, l_i+1, \ldots, l_t) = \text{feasible}(T(i+1, j', l'), l_{i+1}, \ldots, l_t)$. Otherwise, there must be a minimum index $j \leq j^* \leq l$ such that a yes node $v_{ij^*}$ is selected to be in the solution. For such a fixed $j < j^* < l$, let $j'$ be the minimum index such that $v_{i+1,j'}$ is a child of $v_{ij}$ and $l'$ be the maximum index such that $v_{i+1,l'}$ is a child of $v_{i,j^*}$. In this case, if there are values $l^1_i, \ldots, l^1_t, l^2_i, l^2_{i+1}, \ldots, l^2_t$ such that $l^2_i = l_i - 1$, $l_m = l^1_m + l^2_m$ for all $i + 1 \leq m \leq t$, and both $\text{feasible}(T(i + 1, j', l'), l^1_{i+1}, \ldots, l^1_t)$ and $\text{feasible}(T(i, j^* + 1, l), l^2_i, l^2_{i+1}, \ldots, l^2_t)$ return yes, then $\text{feasible}(T(i, j, l), l_i, l_{i+1}, \ldots, l_t)$ also returns yes. Otherwise if for all $j^*$ there are no such values, $\text{feasible}(T(i, j, l), l_i, l_{i+1}, \ldots, l_t)$ returns no. The corner cases when $j^* = j$ or $j^* = l$ can be handled similarly.

It is not hard to verify that $\text{feasible}(T(i, j, l), l_i, l_{i+1}, \ldots, l_t)$ correctly decides whether there is a feasible solution or not for $T(i, j, l)$. To compute the $\text{feasible}()$ function for all possible values one can use a simple dynamic programming based technique. In particular, one can store the values of the function for all possible
parameters in a table. The table is filled up in a bottom-up manner, where the values corresponding to a level \( j \) subtree is computed before computations of the values corresponding to a level \( i \) subtree for \( i < j \). It is not hard to see that the procedure would take polynomial time and space for a constant \( t \).

### 4.3.3 Hardness of Approximation

In this section, we will prove the following theorem.

**Theorem 58.** For any constant \( c \) and any \( \gamma \leq c^\alpha \), NUkC under \( \sqrt{\gamma} \)-PR is hard to approximate in polynomial time within a factor of \( \gamma \), unless \( \text{NP} = \text{RP} \).

Instead of proving this theorem directly, we prove that NUkC under \( \gamma \)-NI-PR is hard to approximate in polynomial time within a factor of \( \gamma \), unless \( \text{NP} = \text{RP} \). Then, by Lemma 54, Theorem 58 follows. To prove the latter result, we use a chain of reductions from 1-in-3SAT to NUkC under \( \psi \)-NI-PR that involves a number of problems.

1-in-3SAT [109]

**INSTANCE:** An ordered pair \((B, C)\) consisting of a set \( B \) of Boolean variables and a set \( C \) of clauses over \( B \) in conjunctive normal form.

**QUESTION:** Is there a truth assignment for \( B \) such that every clause in \( C \) contains exactly one true literal?

The last reduction in the chain of reductions is the same from RMFC-T to NUkC described in the previous section. Additionally, we will argue that the instances of NUkC to which the instances of RMFC-T map are perturbation resilient. However, to ensure that the constructed instance of NUkC has a unique optimal solution,
we will consider an “Unambiguous” version of RMFC-T where if an instance has a feasible solution, it is unique. We also need to ensure that the reduction preserves the number of solutions. Such a reduction is called a parsimonious reduction. We show the hardness of Unambiguous RMFC-T by using a chain of parsimonious reductions from 3SAT [57] to Unambiguous RMFC-T. We will use the above defined problems as the intermediates in this chain of reductions. Also, we consider the Unambiguous version of all of these problems. To refer the Unambiguous version of a problem we add a prefix ‘U-’ to the problem name.

In a celebrated work, Valiant and Vazirani [112] showed that U-3SAT is hard, unless $\text{NP} = \text{RP}$. Schaeffer [109] showed a reduction from 3SAT to 1-in-3SAT to prove the $\text{NP}$-hardness of the latter problem. As noted in [31] the reduction is parsimonious. We use the same reduction (now from U-3SAT to U-1-in-3SAT) to prove the hardness of U-1-in-3SAT. King and MacGillivray [85] showed a reduction from a version of 3SAT (RESTRICTED NAE-3SAT) to the RMFC-T problem to prove the $\text{NP}$-hardness of RMFC-T. We note that one can make a simple modification to the reduction due to King and MacGillivray [85] to obtain a reduction from 1-in-3SAT to RMFC-T. However, their reduction is not parsimonious. Nevertheless, one can modify this reduction by adding asymmetric gadgets to force a unique solution. It follows that there is a parsimonious reduction from U-1-in-3SAT to U-RMFC-T. We summarize our finding in the following lemma.

**Lemma 59.** Given a tree $T$ whose leaves are at the same distance from the root, it is hard to distinguish between the following two cases, unless $\text{NP} = \text{RP}$. 
• **YES**: There is a solution to the U-RMFC-T instance with \( m = 1 \).

• **NO**: There is no solution to the U-RMFC-T instance with \( m = 1 \).

4.3.3.1 Hardness of Perturbation Resilient Version of NUkC

To show the hardness of the \( \psi \)-NI-perturbation resilient version of NUkC, we prove that the constructed instances of U-NUkC in the reduction from U-RMFC-T to U-NUkC in tree metric are \( \gamma \)-NI-PR. First, we remind the reader of the tree metric \( d^* \) we used there. We are given a parameter \( \gamma \) and a tree \( T_\gamma \) with height \( h \) whose leaves are at the same distance from the root. The points in the metric space correspond to all the leaves of \( T_\gamma \). Let \( n \) be the number of leaves. Also, let \( L_i \) be the nodes of \( T_\gamma \) at level \( i \) for \( 1 \leq i \leq h \). For an edge \((u, v)\) of \( T \) such that \( u \in L_h \) and \( v \in L_{h-1} \), we assign a weight \( l(u, v) = (\gamma + 1)/2 \) to \((u, v)\). For each \( u \in L_i \), \( v \in L_{i-1} \) for \( i \leq h - 1 \) such that \((u, v)\) is an edge in \( T_\gamma \), we assign a weight \( l(u, v) = ((\gamma + 1)^{h-i+1} - (\gamma + 1)^{h-i})/2 \).

For any two leaves \( w, w' \), \( d^*(w, w') \) is the length of the shortest path between \( w \) and \( w' \), i.e., if the least common ancestor of \( w, w' \) is in \( L_j \), then \( d^*(w, w') = (\gamma + 1)^{h-j} \).

We set \( t = h \), \( r_t = 0 \) and for any \( 1 \leq j < t \), \( r_j = (\gamma + 1)^{t-j} \). Also, \( k_1 = \ldots = k_t = 1 \).

Let \( L(\gamma) \) be the set of leaves of \( T_\gamma \). As the distance between any two points and the \( r_j \)'s are of the form \((\gamma + 1)^i\) for some \( i \), we have the following observation.

**Observation 60.** The optimal dilation of the instance \( I = \{L(\gamma), d^*, t\} \) is \((\gamma + 1)^i \) for some integer \( i \geq 0 \).

As we have shown before, for any constant \( c \) and any \( \gamma \leq c^{n^c} \), U-NUkC is hard to approximate within a factor of \( \gamma \) for the metric space \((T_\gamma, d^*)\), unless \( \text{NP} = \text{RP} \).
Next, we prove the following lemma.

**Lemma 61.** The instance $\mathcal{I} = \{L(\gamma), d^*, t\}$ is $\gamma$-NI-PR.

**Proof.** Let $\mathcal{O}$ be the optimal clustering of $\mathcal{I}$ and $\alpha$ be its dilation. Consider any $\gamma$-NI-perturbation $d'$ of $d^*$. We prove that the optimal clustering $\mathcal{O}'$ of the instance $\mathcal{I}' = \{L(\gamma), d', t\}$ is same as $\mathcal{O}$. Suppose for the sake of contradiction that $\mathcal{O}'$ is not same as $\mathcal{O}$. As $d'$ is a $\gamma$-NI-perturbation (the distances are non-increasing), the dilation of $\mathcal{O}'$ is at most $\alpha$. We show that $\mathcal{O}'$ is also a feasible clustering for $\mathcal{I}$ with dilation at most $\alpha$.

Consider any non-singleton cluster $C \in \mathcal{O}'$ with center $c_1$ that is covered by an $r_j$-ball for $j < t$. Then, for all pairs of points $p, q \in C$, $d'(p, q) \leq \alpha r_j$. From Observation 60, it follows that $\alpha r_j = (\gamma + 1)^i$ for some $i$. As $d'$ is a $\gamma$-NI-perturbation of $d^*$, $d^*(p, q) \leq \gamma \cdot d'(p, q) < (\gamma + 1)^{i+1}$. Now, the way $T_\gamma$ is constructed, there is no distance values strictly between $(\gamma + 1)^i$ and $(\gamma + 1)^{i+1}$. Hence, $d^*(p, q) \leq (\gamma + 1)^i = \alpha r_j$, and the ball $B(c_1, \alpha r_j)$ covers the points of the cluster $C$ w.r.t. $d^*$. It follows that $\mathcal{O}'$ is also a feasible clustering for $\mathcal{I}$ with dilation at most $\alpha$. But, as per our assumption $\mathcal{O}$ and $\mathcal{O}'$ are different, and thus the optimal clustering of $\mathcal{I}$ is not unique. This is a contradiction, and hence $\mathcal{O}$ and $\mathcal{O}'$ must be same. \qed

Hence, we obtain the following theorem.

**Theorem 62.** For any constant $c$ and any $\gamma \leq c^{n^c}$, $NUkC$ under $\gamma$-NI-PR is hard to approximate in polynomial time within a factor of $\gamma$, unless $\text{NP} = \text{RP}$. 
CHAPTER 5
CAPACITATED COVERING

In this chapter, we describe the bi-criterion approximations for both the Metric Monotonic Capacitated Covering (MMCC) and Euclidean Monotonic Capacitated Covering (EMCC) problems. We also discuss two inapproximability results for MMCC.

**Definition 63.** An \((\alpha, \beta)\)-approximation for a version of MMCC, is a solution in which the balls may be expanded by a factor of \(\beta\) (i.e. for any ball \(B_i\), and any point \(p_j \in P\) that is assigned to \(B_i\), \(d(c_i, p_j) \leq \beta \cdot r_i\)), and its cost is at most \(\alpha\) times that of an optimal solution (which does not expand the balls).

### 5.1 LP Relaxation for MMCC

Recall that the input for the MMCC consists of a set \(P\) of points and a set \(B\) of balls in some metric space, along with an integer capacity \(U_i > 0\) for ball \(B_i \in B\). We assume that for any two input balls \(B_i, B_j \in B\), it holds that \(r_i > r_j \implies U_i \geq U_j\). The goal is to compute a minimum cardinality subset \(B' \subseteq B\) for which each point in \(P\) can be assigned to a ball \(B'\) containing it in such a way that no more than \(U_i\) points are assigned to ball \(B_i\). Let \(d(p, q)\) denote the distance between two points \(p\) and \(q\) in the metric space. Let \(B(c, r)\) denote the ball of radius \(r\) centered at point \(c\). We let \(c_i\) and \(r_i\) denote the center and radius of ball \(B_i \in B\); thus, \(B_i = B(c_i, r_i)\).

First we consider an integer programming formulation of MMCC. For each set \(B_i \in B\), let \(y_i = 1\) if the ball \(B_i\) is selected in the solution, and \(0\) otherwise. Similarly,
for each point \( p_j \in X \) and each ball \( B_i \in \mathcal{B} \), let the variable \( x_{ij} = 1 \) if \( p_j \) is assigned to \( B_i \), and \( x_{ij} = 0 \) otherwise. We relax these integrality constraints, and state the corresponding linear program as follows:

\[
\begin{align*}
\text{minimize} & \quad \sum_{B_i \in \mathcal{B}} y_i \\
\text{s.t.} & \quad x_{ij} \leq y_i \quad \forall p_j \in P, \forall B_i \in \mathcal{B} \\
& \quad \sum_{p_j \in P} x_{ij} \leq y_i \cdot U_i \quad \forall B_i \in \mathcal{B} \\
& \quad \sum_{B_i \in \mathcal{B}} x_{ij} = 1 \quad \forall p_j \in P \\
& \quad x_{ij} = 0 \quad \forall p_j \in P, \forall B_i \in \mathcal{B} \text{ such that } p_j \notin B_i \\
& \quad x_{ij} \geq 0 \quad \forall p_j \in P, \forall B_i \in \mathcal{B} \\
& \quad 0 \leq y_i \leq 1 \quad \forall B_i \in \mathcal{B}
\end{align*}
\]

Subsequently, we will refer to an assignment \((x, y)\) that is feasible or infeasible with respect to Constraints 5.1-5.6 as just a solution. The cost of the LP solution \( \sigma = (x, y) \) (feasible or otherwise), denoted by cost(\( \sigma \)), is defined as \( \sum_{B_i \in \mathcal{B}} y_i \).

### 5.2 The Algorithmic Framework

In this section, we describe our framework for extracting an integral solution from a fractional solution to the above LP. The framework consists of two major steps – Preprocessing and the Main Rounding. The Main Rounding step is in turn divided into two smaller steps – Cluster Formation and Selection of Objects. For simplicity of exposition, we first describe the framework with respect to the MMCC problem as an algorithm and analyze the approximation factor achieved by this algorithm for
5.2.1 The Algorithm for the MMCC Problem

Before we describe the algorithm we introduce some definitions and notation which will heavily be used throughout this section. For point \( p_j \in P \) and ball \( B_i \in \mathcal{B} \), we refer to \( x_{ij} \) as the flow from \( B_i \) to \( p_j \); if \( x_{ij} > 0 \), then we say that the ball \( B_i \) serves the point \( p_j \). Each ball \( B_i \in \mathcal{B} \) can be imagined as a source of at most \( y_i \cdot U_i \) units of flow, which it distributes to some points in \( P \).

We now define an important operation, called rerouting of flow. “Rerouting of flow for a set \( P' \subseteq P \) of points from a set of balls \( \mathcal{B}' \) to a ball \( B_k \notin \mathcal{B}' \)” means obtaining a new solution \((\hat{x}, \hat{y})\) from the current solution \((x, y)\) in the following way:

(a) For all points \( p_j \in P' \), \( \hat{x}_{kj} = x_{kj} + \sum_{B_i \in \mathcal{B}'} x_{ij} \); (b) for all points \( p_j \in P' \) and balls \( B_i \in \mathcal{B}' \), \( \hat{x}_{ij} = 0 \); (c) the other \( \hat{x}_{ij} \) variables are the same as the corresponding \( x_{ij} \) variables. The relevant \( \hat{y}_i \) variables may also be modified depending on the context where this operation is used.

Let \( 0 < \alpha \leq \frac{1}{2} \) be a parameter to be fixed later. A ball \( B_i \in \mathcal{B} \) is heavy if the corresponding \( y_i = 1 \), and light, if \( 0 < y_i \leq \alpha \). Corresponding to a feasible LP solution \((x, y)\), let \( \mathcal{H} = \{B_i \in \mathcal{B} \mid y_i = 1\} \) denote the set of heavy balls, and \( \mathcal{L} = \{B_i \in \mathcal{B} \mid 0 < y_i \leq \alpha\} \) denote the set of light balls. We emphasize that the set \( \mathcal{L} \) of light and \( \mathcal{H} \) of heavy balls are defined w.r.t. an LP solution; however, the reference to the LP solution may be omitted when it is clear from the context.

Now we move on towards the description of the algorithm. The algorithm,
given a feasible fractional solution \( \sigma = (x, y) \), rounds \( \sigma \) to a solution \( \hat{\sigma} = (\hat{x}, \hat{y}) \) such that \( \hat{y} \) is integral, and the cost of \( \hat{\sigma} \) is within a constant factor of the cost of \( \sigma \). The \( \hat{x} \) variables are non-negative but may be fractional. Furthermore, each point receives unit flow from the balls that are chosen (\( y \) values are 1), and the amount of flow each chosen ball sends is bounded by its capacity. Notably, no point gets any non-zero amount of flow from a ball that is not chosen (\( y \) value is 0). Moreover, for any ball \( B_i \) and any \( p_j \in P \), if \( B_i \) serves \( p_j \), then \( d(c_i, p_j) \) is at most a constant times \( r_i \). We expand each ball by a constant factor so that it contains all the points it serves.

We note that in \( \hat{\sigma} \) points might receive fractional amount of flow from the chosen balls. However, since the capacity of each ball is integral, we can find, using a textbook argument for integrality of flow, another solution with the same set of chosen balls, such that the new solution satisfies all the properties of \( \hat{\sigma} \) and the additional property, that for each point \( p \), there is a single chosen ball that sends one unit of flow to \( p \) [39]. Thus, choosing an optimal LP solution as the input \( \sigma = (x, y) \) of the rounding algorithm yields a constant approximation for MMCC by expanding each ball by at most a constant factor.

Our LP rounding algorithm consists of two steps. The first step is a preprocessing step where we construct a fractional LP solution \( \sigma = (\bar{x}, \bar{y}) \) from \( \sigma \), such that each ball in \( \bar{\sigma} \) is either heavy or light, and for each point \( p_j \in P \), the amount of flow that \( p_j \) can potentially receive from the light balls is at most \( \alpha \). The latter property will be heavily exploited in the next step. The second step is the core step of the algorithm where we round \( \bar{\sigma} \) to the desired integral solution.
We note that throughout the algorithm, for any intermediate LP solution that we consider, we maintain the following two invariants: (i) Each ball \( B_i \) sends at most \( U_i \) units of flow to the points, and (ii) Each point receives exactly one unit of flow from the balls. With respect to a solution \( \sigma = (x, y) \), we define the *available capacity* of a ball \( B_i \in \mathcal{B} \), denoted \( \text{AvCap}(B_i) \), to be \( U_i - \sum_{p_j \in P} x_{ij} \). We now describe the preprocessing step.

### 5.2.1.1 The Preprocessing Step

**Lemma 1.** Given a feasible LP solution \( \sigma = (x, y) \), and a parameter \( 0 < \alpha \leq \frac{1}{2} \), there exists a polynomial time algorithm to obtain another LP solution \( \overline{\sigma} = (\overline{x}, \overline{y}) \) that satisfies Constraints 5.1-5.6 except 5.4 of MMCC-LP. Additionally, \( \overline{\sigma} \) satisfies the following properties.

1. Any ball \( B_i \in \mathcal{B} \) with non-zero \( \overline{y}_i \) is either heavy \( (\overline{y}_i = 1) \) or light \( (0 < \overline{y}_i \leq \alpha) \).

2. For each point \( p_j \in P \), we have that

\[
\sum_{B_i \in \mathcal{L}, x_{ij} > 0} \overline{y}_i \leq \alpha, \tag{5.7}
\]

where \( \mathcal{L} \) is the set of light balls with respect to \( \overline{\sigma} \).

3. For any heavy ball \( B_i \), and any point \( p_j \in P \) served by \( B_i \), \( d(c_i, p_j) \leq 3r_i \).

4. For any light ball \( B_i \), and any point \( p_j \in P \) served by \( B_i \), \( d(c_i, p_j) \leq r_i \).

5. \( \text{cost}(\overline{\sigma}) \leq \frac{1}{\alpha} \text{cost}(\sigma) \).

**Proof.** The algorithm starts off by initializing \( \overline{\sigma} \) to \( \sigma \). While there is a violation of Inequality 5.7, we perform the following steps.
1. We pick an arbitrary point $p_j \in P$, for which Inequality 5.7 is not met. Let $\mathcal{L}_j$ be a subset of light balls serving $p_j$ such that $\alpha < \sum_{B_i \in \mathcal{L}_j} \overline{y}_i \leq 2\alpha$. Note that such a set $\mathcal{L}_j$ always exists because the $\overline{y}_i$ variables corresponding to light balls are at most $\alpha \leq \frac{1}{2}$. Let $B_k$ be a ball with the largest radius from the set $\mathcal{L}_j$. (If there is more than one ball with the largest radius, we consider one having the largest capacity among those. Throughout the paper we follow this convention.) Since $r_k \geq r_m$ for all other balls $B_m \in \mathcal{L}_j$, we have, by the monotonicity assumption, that $U_k \geq U_m$.

2. We set $\overline{y}_k \leftarrow \sum_{B_i \in \mathcal{L}_j} \overline{y}_i$, and $\overline{y}_m \leftarrow 0$ for all $B_m \in \mathcal{L}_j \setminus \{B_k\}$. Note that $\overline{y}_k \leq 2\alpha \leq 1$. Let $A = \{p_t \in P \mid \overline{x}_{it} > 0 \text{ for some } B_i \in \mathcal{L}_j \setminus \{B_k\}\}$ be the set of “affected” points. We reroute the flow for all the affected points in $A$ from $\mathcal{L}_j \setminus \{B_k\}$ to the ball $B_k$. Since $U_k \geq U_m$ for all other balls $B_m \in \mathcal{L}_j$, $B_k$ has enough available capacity to “satisfy” all “affected” points. In $\sigma$, all other $\overline{x}_{ij}$ and $\overline{x}_i$ variables remain same as before. (Note: Since $B_k$ had the largest radius from the set $\mathcal{L}_j$, all the points in $A$ are within distance $3r_k$ from its center $c_k$, as seen using the triangle inequality. Also, since $\overline{y}_k > \alpha$, $B_k$ is no longer a light ball.)

Finally, for all balls $B_i$ such that $\overline{y}_i > \alpha$, we set $\overline{y}_i = 1$, making them heavy. Thus cost($\sigma$) is at most $\frac{1}{\alpha}$ times cost($\sigma$), and $\sigma$ satisfies all the conditions stated in the lemma.
Remark. As a byproduct of Lemma 1, we get a simple $(4, 3)$-approximation algorithm for the soft capacitated version of our problem that allows making multiple copies of the input balls.

5.2.1.2 The Main Rounding Step

The main rounding step can be logically divided into two stages. The first stage, Cluster Formation, is the crucial stage of the algorithm. Note that there can be many light balls in the preprocessed solution. Including all these balls in the final solution may incur a huge cost. Thus we use a careful strategy based on flow rerouting to select a small number of balls. The idea is to use the capacity of a selected light ball to reroute as much flow as possible from other intersecting balls. This in turn frees up some capacity at those balls. The available capacity of each heavy ball is used, when possible, to reroute all the flow from some light ball intersecting it; this light ball is then added to a cluster centered around the heavy ball. Notably, for each cluster, the heavy ball is the only ball in it that actually serves some points, as we have rerouted flow from the other balls in the cluster to the heavy ball. In the second stage, referred to as Selection of Objects, we select exactly one ball (in particular, a largest ball) from each cluster as part of the final solution, and reroute the flow from the heavy ball to this ball, and expand it by the required amount. Together, these two stages ensure that we do not end up choosing many light balls.

We now describe the two stages in detail. Recall that any ball in the preprocessed solution is either heavy or light. Also $\mathcal{L}$ denotes the set of light balls and $\mathcal{H}$ the set of heavy balls. Note that any heavy ball $B_i$ may serve a point $p_j$ which is at
a distance $3r_i$ from $c_i$. We expand each heavy ball by a factor of 3 so that $B_i$ can contain all points it serves.

1. **Cluster Formation.** In this stage, each light ball, will be added to either a set $O$ (that will eventually be part of the final solution), or a cluster corresponding to some heavy ball. Till the very end of this stage, the sets of heavy and light balls remain unchanged. The set $O$ is initialized to $\emptyset$. For each heavy ball $B_i$, we initialize the cluster of $B_i$, denoted by $\text{cluster}(B_i)$ to $\{B_i\}$. We say a ball is clustered if it is added to a cluster.

At any point, let $\Lambda$ denote the set consisting of each light ball that is (a) not in $O$, and (b) not yet clustered. While the set $\Lambda$ is non-empty, we perform the following steps.

(a) While there is a heavy ball $B_i$ and a light ball $B_t \in \Lambda$ such that (1) $B_t$ intersects $B_i$; and (2) $\text{AvCap}(B_i)$ is at least the flow $\sum_{p_j \in P} x_{tj}$ out of $B_t$:

1. For all the points served by $B_t$, we reroute the flow from $B_t$ to $B_i$.

2. We add $B_t$ to $\text{cluster}(B_i)$.

After the execution of this while loop, if the set $\Lambda$ becomes empty, we stop and proceed to the Selection of Objects stage. Otherwise, we proceed to the following.

(b) For any ball $B_j \in \Lambda$, let $A_j$ denote the set of points currently being served by $B_j$. Also, for $B_j \in \Lambda$, let $k_j = \min\{|U_j|, |A_j|\}$, i.e. $k_j$ denotes the minimum of its capacity, and the number of points that it currently serves.
We select the ball $B_t \in \Lambda$ with the maximum value of $k_j$, and add it to the set $O$.

(c) Since we have added $B_t$ to the set $O$ that will be among the selected balls, we use the available capacity at $B_t$ to reroute flow to it. This is done based on the following three cases depending on the value of $k_t$.

1. $k_t = |A_t| \leq U_t$. In this case, for each point $p_t$ in $B_t$ that gets served by $B_t$, we reroute the flow of $p_t$ from $B \setminus O$ to $B_t$. Note that after the rerouting, $p_t$ is no longer being served by a ball in $\Lambda$. The rerouting increases the available capacity of other balls intersecting $B_t$. In particular, for each $B_i \in H$, $\text{AvCap}(B_i)$ increases by $\sum_{p_t: B_t \text{ serves } p_t} x_{it}$.

2. $k_t = U_t < |A_t|$, but $k_t = U_t > 1$. Observe that the flow out of ball $B_t$ is $\sum_{p_j \in A_t} x_{tj} \leq \alpha U_t$; thus $\text{AvCap}(B_t) \geq (1 - \alpha) U_t = (1 - \alpha) k_t$. In this case, we select a point $p_j \in A_t$ arbitrarily, and reroute the flow of $p_j$ from $B \setminus O$ to $B_t$. This will increase the available capacity of other balls in $B \setminus O$ that were serving $p_j$. Also note that $p_j$ is no longer being served by a ball in $\Lambda$.

We repeat the above flow rerouting process for other points of $A_t$ until we encounter a point $p_t$ such that rerouting the flow of $p_t$ from $B \setminus O$ to $B_t$ violates the capacity of $B_t$. Thus the flow assignment of $p_t$ remains unchanged. Note that we can reroute the flow of at least $\lfloor (1 - \alpha) k_t \rfloor = \lfloor (1 - \alpha) U_t \rfloor \geq 1$ points of $A_t$ in this manner, since $U_t > 1$ and $\alpha \leq 1/2$.

3. $k_t = U_t = 1 < |A_t|$. Note that $B_t$ has used $\sum_{p_j \in A_t} x_{tj} \leq \alpha U_t = \alpha$
capacity. In this case, we pick a point $p_j \in A_t$ arbitrarily, and then perform the following two steps:

(i). Reroute the flow of $p_j$ from $\Lambda$ to $B_t$; after this, $p_j$ is no longer being served by a ball in $\Lambda$. Note that in this step, we reroute at most $\alpha$ amount of flow. Therefore, at this point we have $\text{AvCap}(B_t) \geq 1 - 2\alpha$. Let $f$ be the amount of flow $p_j$ receives from the balls in $O$.

(ii). Then we reroute $\min\{\text{AvCap}(B_t), 1 - f\}$ amount of flow of $p_j$ from the set $H$ to $B_t$.

When the loop terminates, we have that each light ball is either in $O$ or clustered. We set $y_i \leftarrow 1$ for each ball $B_i \in O$, thus making it heavy. For convenience, we also set $\text{cluster}(B_i) = \{B_i\}$ for each $B_i \in O$. Note that, throughout the algorithm we ensure that, if a point $p_j \in P$ is currently served by a ball $B_i \in \Lambda$, then the amount of flow $p_j$ receives from any ball $B_i' \in O \cup \Lambda$ is the same as that in the preprocessed solution, i.e., the flow assignment of $p_j$ w.r.t. the balls in $O \cup \Lambda$ remains unchanged.

2. **Selection of Objects.** At the start of this stage, we have a collection of clusters, each centered around a heavy ball, such that the light balls in each cluster intersect the heavy ball. We are going to pick exactly one ball from each cluster and add it to a set $C$. Let $C = \emptyset$ initially. For each heavy ball $B_i$, we consider $\text{cluster}(B_i)$ and perform the following steps.

(a) If $\text{cluster}(B_i)$ consists of only the heavy ball, we add $B_i$ to $C$. 
(b) Otherwise, let \( B_j \) be a largest ball in cluster\((B_i)\). If \( B_j = B_i \), then we expand it by a factor of 3. Otherwise, \( B_j \) is a light ball intersecting with \( B_i \), in which case we expand it by a factor of 5. In this case, we also reroute the flow from the heavy ball to the selected ball \( B_j \). Note that since we always choose a largest ball in the cluster, its capacity is at least that of the heavy ball, because of the monotonicity assumption. We add \( B_j \) to \( C \), and we set \( y_s \leftarrow 0 \) for any other ball \( B_s \) in the cluster.

After processing the clusters, we set \( y_t \leftarrow 1 \) for each ball \( B_t \in C \). Finally, we return the current set of heavy balls (i.e., \( C \)) as the set of selected balls. Note that the flow out of each such ball is at most its capacity, and each point receives one unit of flow from the (possibly expanded) balls that contain it. As mentioned earlier, this can be converted into an integral flow.

### 5.2.1.3 The Analysis of the Rounding Algorithm

Let \( OPT \) be the cost of an optimal solution. We establish a bound on the number of balls our algorithm outputs by bounding the size of the set \( C \). Then we conclude by showing that any input ball that is part of our solution expands by at most a constant factor to cover the points it serves.

For notational convenience, we refer to the solution \( \sigma = (\pi, \overline{y}) \) at hand after preprocessing, as \( \sigma = (x, y) \). Now we bound the size of the set \( O \) computed during Cluster Formation. The basic idea is that each light ball added to \( O \) creates significant available capacity in the heavy balls. Furthermore, whenever there is enough available capacity, a heavy ball clusters intersecting light balls, thus preventing them from being
added to $O$. The actual argument is more intricate because we need to work with a notion of $y$-accumulation, a proxy for available capacity. The way the light balls are picked for addition to $O$ plays a crucial role in the argument.

Let $H_1$ (resp. $L_1$) be the set of heavy (resp. light) balls after preprocessing, and $I$ be the total number of iterations in the Cluster Formation stage. Also let $L_j$ be the light ball selected (i.e. added to $O$) in iteration $j$ for $1 \leq j \leq I$. Now, $L_t$ maximizes $k_j$ amongst all balls from $\Lambda$ in iteration $t$ (Recall that $k_j$ was defined as the minimum of the number of points being served by $L_j$, and its capacity). Note that $k_1 \geq k_2 \geq \cdots \geq k_t$. For any $B_i \in H_1$, denote by $F(L_t, B_i)$, the total amount of flow rerouted in iteration $t$ from $B_i$ to $L_t$ corresponding to the points $B_i$ serves. This is the same as the increase in $\text{AvCap}(B_i)$ when $L_t$ is added to $O$. Correspondingly, we define $Y(L_t, B_i)$, the “$y$-credit contributed by $L_t$ to $B_i$”, to be $\frac{F(L_t, B_i)}{k_t}$. Now, the increase in available capacity over all balls in $H_1$ is $F_t = \sum_{B_i \in H_1} F(L_t, B_i)$. The approximation guarantee of the algorithm depends crucially on the following simple lemma, which states that in each iteration we make “sufficiently large” amount of flow available for the set of heavy balls.

**Lemma 2.** Consider a ball $B_i \in O$ processed in the Cluster Formation stage, step $c$. For $0 < \alpha \leq 3/8$, $F_t \geq \frac{1}{5} \frac{1}{\alpha} k_t$.

**Proof.** The algorithm ensures that the flow assignment of each point in $A_t$ w.r.t. the balls in $O \cup \Lambda$ is the same as that in the preprocessed solution. Thus by property 2 of Lemma 1, each such point gets at most $\alpha$ amount of flow from the balls in $O \cup \Lambda$. Now there are three cases corresponding to the three substeps of step $c$. 

1. \( k_t = |\mathcal{A}_t| \leq U_t \). For each point in \( \mathcal{A}_t \), at most \( \alpha \) amount of flow comes from the balls in \( \mathcal{O} \cup \Lambda \). So the remainder is rerouted from the balls in \( \mathcal{H}_1 \) resulting in a contribution of at least \( 1 - \alpha \) towards \( F_t \). Therefore, we get that \( F_t \geq (1 - \alpha)k_t \geq \frac{1}{5}k_t \), since \( 0 < \alpha \leq 3/8 \).

2. \( 1 < k_t = U_t < |\mathcal{A}_t| \). It is possible to reroute the flow of at least \( \lfloor (1 - \alpha)U_t \rfloor = \lfloor (1 - \alpha)k_t \rfloor \) points of \( \mathcal{A}_t \) from \( \mathcal{B} \setminus \mathcal{O} \) to \( B_t \). Therefore, we get that \( F_t \geq (1 - \alpha)\lfloor (1 - \alpha)k_t \rfloor \). When \( k_t > 1 \), the previous quantity is at least \( \frac{1}{5}k_t \), again by using the fact that \( 0 < \alpha \leq 3/8 \).

3. When \( 1 = k_t = U_t < |\mathcal{A}_t| \), \( F_t \geq (1 - 2\alpha) \geq \frac{1}{5}k_t \), as \( 0 < \alpha \leq 3/8 \).

\[ \Box \]

At any moment in the Cluster Formation stage, for any ball \( B_i \in \mathcal{H}_1 \), define its \( y \)-accumulation as

\[ \tilde{y}(B_i) = \left( \sum_{L_t \in \mathcal{O}} Y(L_t, B_i) \right) - \left( \sum_{B_j \in \mathcal{L} \cap \text{cluster}(B_i)} y_j \right). \]

The idea is that \( B_i \) gets \( y \)-credit when a light ball is added to \( \mathcal{O} \), and loses \( y \)-credit when it adds a light ball to \( \text{cluster}(B_i) \); thus, \( \tilde{y}(B_i) \), a proxy for the available capacity of \( B_i \), indicates the “remaining” \( y \)-credit. The next lemma gives a relation between the \( y \)-accumulation of \( B_i \) and its available capacity.

**Lemma 3.** Fix a heavy ball \( B_i \in \mathcal{H}_1 \), and an integer \( 1 \leq t \leq I \). Suppose that \( L_1, L_2, \cdots, L_t \) have been added to \( \mathcal{O} \). Then \( \text{AvCap}(B_i) \geq \tilde{y}(B_i) \cdot k_t \).
Proof. The proof is by induction on \( t \). For this proof, we abbreviate \( \text{AvCap}(B_i) \) by \( A_i \).

In the first iteration, just after adding \( L_1 \), \( A_i \geq F(L_1, B_i) = Y(L_1, B_i) \cdot k_1 \geq \tilde{y}(B_i) \cdot k_1 \).

Assume inductively that we have added balls \( L_1, \ldots, L_{t-1} \) to the set \( O \), and that just after adding \( L_{t-1} \), the claim is true. That is, if \( \tilde{y}(B_i) \) and \( A_i \) are, respectively, the \( y \)-accumulation and the available capacity of \( B_i \) just after adding \( L_{t-1} \), then \( A_i \geq \tilde{y}(B_i) \cdot k_{t-1} \).

Consider the iteration \( t \). At step (a) of Cluster Formation, \( B_i \) uses up some of its available capacity to add 0 or more balls to \( \text{cluster}(B_i) \), after which at step (b) we add \( L_t \) to \( O \). Suppose that at step (a), one or more balls are added to \( \text{cluster}(B_i) \).

Let \( B_j \) be the first such ball, and let \( k \) and \( C_1 \) be the number of points \( B_j \) serves and the capacity of \( B_j \), respectively. Then the amount of capacity used by \( B_j \) is at most

\[
\min\{C_1 \cdot y_j, k \cdot y_j\} = \min\{C_1, k\} \cdot y_j \leq k_{t-1} \cdot y_j
\]

where the last inequality follows because of the order in which we add balls to \( O \). Now, after adding \( B_j \) to \( \text{cluster}(B_i) \), the new \( y \)-accumulation becomes \( \tilde{y}(B_i)' = \tilde{y}(B_i) - y_j \).

As for the available capacity,

\[
A_i' \geq A_i - k_{t-1} \cdot y_j \geq (\tilde{y}(B_i) \cdot k_{t-1}) - k_{t-1} \cdot y_j \geq (\tilde{y}(B_i) - y_j) \cdot k_{t-1} = \tilde{y}(B_i)' \cdot k_{t-1}
\]

Therefore, the claim is true after addition of the first ball \( B_j \). Note that \( B_i \) may add multiple balls to \( \text{cluster}(B_i) \), and the preceding argument would work after each such addition.

Now consider the moment when \( L_t \) is added to \( O \). Let \( \tilde{y}(B_i) \) denote the \( y \)-accumulation just before this. Now, the new \( y \)-accumulation of \( B_i \) becomes \( \tilde{y}(B_i)' = \)
\( \tilde{y}(B_i) + Y(L_t, B_i) \). If \( \tilde{y}(B_i) \leq 0 \), then the new available capacity is

\[
A'_i \geq F(L_t, B_i) = Y(L_t, B_i) \cdot k_t \geq \tilde{y}(B_i)' \cdot k_t.
\]

If \( \tilde{y}(B_i) > 0 \), the new available capacity, using the inductive hypothesis, is

\[
A'_i \geq \tilde{y}(B_i) \cdot k_{t-1} + Y(L_t, B_i) \cdot k_t \geq (\tilde{y}(B_i) + Y(L_t, B_i)) \cdot k_t = \tilde{y}(B_i)' \cdot k_t
\]

where, in the second inequality we use \( k_t \leq k_{t-1} \).

Now, in the next lemma, we show that any ball \( B_i \in \mathcal{H}_1 \) cannot have “too-much” \( y \)-accumulation at any moment during Cluster Formation.

**Lemma 4.** At any moment in the Cluster Formation stage, for any ball \( B_i \in \mathcal{H}_1 \), we have that \( \tilde{y}(B_i) \leq 1 + \alpha \).

**Proof.** The proof is by contradiction. Let \( B_i \in \mathcal{H}_1 \) be the first ball that violates the condition. As \( \tilde{y}(B_i) \) increases only due to addition of a light ball to set \( \mathcal{O} \), suppose \( L_t \) was the ball whose addition to \( \mathcal{O} \) resulted in the violation.

Let \( \tilde{y}(B_i) \) and \( \tilde{y}(B_i)' = \tilde{y}(B_i) + Y(L_t, B_i) \) be the \( y \)-accumulations of \( B_i \) just before and just after the addition of \( L_t \). Because of the assumption, \( \tilde{y}(B_i) \leq 1 + \alpha \). So the increase in the \( y \)-accumulation of \( B_i \) must be because \( Y(L_t, B_i) > 0 \). Thus, \( L_t \) intersects \( B_i \). However, \( Y(L_t, B_i) \leq 1 \) by definition. Therefore, we have \( \tilde{y}(B_i) > \alpha \).

Now, by Lemma 3, just before addition of \( L_t \), \( \text{AvCap}(B_i) \geq \tilde{y}(B_i) \cdot k_{t-1} > \alpha \cdot k_{t-1} \geq \alpha \cdot k_t \), as \( k_t \leq k_{t-1} \). However, \( L_t \) is a light ball, and so the total flow out of \( L_t \) is at most \( \alpha k_t \). Therefore, the available capacity of \( B_i \) is large enough that we can add \( L_t \) to \( \text{cluster}(B_i) \), instead of to the set \( \mathcal{O} \), which is a contradiction.
Lemma 5. At the end of Cluster Formation stage, we have $|\mathcal{O}| \leq 5 \cdot \left( (1 + \alpha) \cdot |\mathcal{H}_1| + \sum_{B_j \in \mathcal{L}_1} y_j \right)$, where $0 < \alpha \leq 3/8$.

Proof. At the end of Cluster Formation stage,

$$\sum_{B_i \in \mathcal{H}_1} \tilde{y}(B_i) \geq \sum_{B_i \in \mathcal{H}_1} \sum_{1 \leq i \leq l} Y(L_t, B_i) - \sum_{B_i \in \mathcal{H}_1} \sum_{B_j \in \text{cluster}(B_i)} y_j$$

$$\geq \sum_{1 \leq i \leq l} \left( F_i / k_i \right) - \sum_{B_j \in \mathcal{L}_1} y_j$$

$$(\because F_i = \sum_{B_i \in \mathcal{H}_1} F(L_t, B_i) = k_t \cdot \sum_{B_i \in \mathcal{H}_1} Y(L_t, B_i))$$

$$\geq \frac{1}{5} \cdot |\mathcal{O}| - \sum_{B_j \in \mathcal{L}_1} y_j$$

(5.8)

Where we used Observation 2 to get the last inequality.

Now, adding the inequality of Lemma 4 over all $B_i \in \mathcal{H}_1$, we have that $\sum_{B_i \in \mathcal{H}_1} \tilde{y}(B_i) \leq (1 + \alpha) \cdot |\mathcal{H}_1|$. Combining this with (5.8) yields the desired inequality. \qed

Lemma 6. The cost of the solution returned by the algorithm is at most 21 times the cost of an optimal solution.

Proof. Let $\sigma = (x, y)$ be the preprocessed LP solution. Now, the total number of balls in the solution is $|\mathcal{O}| + |\mathcal{H}_1|$. Using Lemma 5,

$$|\mathcal{O}| + |\mathcal{H}_1| \leq 5 \cdot \left( (1 + \alpha) \cdot |\mathcal{H}_1| + \sum_{B_j \in \mathcal{L}_1} y_j \right) + |\mathcal{H}_1|$$

$$\leq (6 + 5\alpha) \left( \sum_{B_j \in \mathcal{H}_1} y_j + \sum_{B_j \in \mathcal{L}_1} y_j \right)$$

$$\leq (6 + 5\alpha) \cdot \text{cost}(\sigma)$$

$$\leq \left( \frac{6 + 5\alpha}{\alpha} \right) \cdot OPT = 21 \cdot OPT$$

(by setting $\alpha = 3/8$) \qed
Lemma 7. *In the algorithm each input ball is expanded by at most a factor of 9.*

*Proof.* Recall that when a light ball becomes heavy in the preprocessing step, it is expanded by a factor of 3. Therefore after the preprocessing step, any heavy ball in a solution may be an expanded or unexpanded ball.

Now, consider the selection of the balls in the second stage. If a cluster consists of only a heavy ball, then it does not expand any further. Since it might be an expanded light ball, the total expansion factor is at most 3.

Otherwise, for a fixed cluster, let \( r_l \) and \( r_h \) be the radius of the largest light ball and the heavy ball, respectively. If \( r_l \geq r_h \), then the overall expansion factor is 5.

Otherwise, if \( r_l < r_h \), then the heavy ball is chosen, and it is expanded by a factor of at most 3. Now as the heavy ball might already be expanded by a factor of 3 during the preprocessing step, here the overall expansion factor is 9. \( \square \)

If the capacities of all balls are equal, then one can improve the expansion factor to 6.47 by using an alternative procedure to the *Selection of Balls* stage.

Lemma 8. *If the capacities of all balls are equal, then there exists an alternative procedure to the Selection of Balls stage, that guarantees that in the algorithm any ball is expanded by at most a factor of 6.47.*

*Proof.* If the capacities of all balls are equal (say \( U \)), then we proceed in the same way until the *Selection of Balls* stage. Then, we use the following scheme that guarantees a smaller expansion factor for this special case. We first describe the scheme and then analyze it.
Fix a cluster obtained after the *Cluster Formation* stage. If the cluster contains only a heavy ball, then we add it to a set $C$ (initialized to $\emptyset$), without expansion.

Otherwise, let $r_l$ denote the radius of a largest ball in the cluster, and let $r_h$ be the radius of the heavy ball. Let $B_l$ and $B_h$ be the corresponding balls. We consider the following 3 cases:

- $r_l \geq r_h$: In this case, let $B = B_l$. We set its new radius to be $3r_l + 2r_h$.
- $\frac{1}{\sqrt{3}} \leq r_l < r_h$: Let $B = B_l$. We set its new radius to be $3r_l + 2r_h$.
- $r_l < \frac{1}{\sqrt{3}} r_h$: Let $B = B_h$. We set its new radius to be $r_h + 2r_l$.

Finally, if $B \neq B_h$, then we reroute the flow from $B_h$ to $B$, set $y_h \leftarrow 0$, and add $B$ to the set $C$ respectively. Finally, we set $y_i \leftarrow 1$ for all balls $B_i \in C$, and return $C$ as the solution.

To analyze the scheme, note that a heavy ball at the end of *Cluster Formation* stage may have been a light ball that was expanded by a factor of 3 in the preprocessing step. Therefore, if a cluster contains only a heavy ball, then the total expansion factor is at most 3. Otherwise, we analyze each of the 3 cases discussed above separately.

In the first case, $3r_l + 2r_h \leq 5r_l$.

In the second case, $3r_l + 2r_h \leq (3 + 2 \cdot \sqrt{3})r_l < 6.47r_l$.

In the third case, $r_h + 2r_l \leq (1 + 2/\sqrt{3})r_h$. But $B_h$ might be originally a light ball that was expanded by a factor of 3 in the preprocessing step. Therefore, the total expansion factor is at most $3 + 2 \cdot \sqrt{3} < 6.47$.

Lastly, from Lemmas 6 and 7, we get the following theorem.
Theorem 64. There is a polynomial time \((21, 9)\)-approximation algorithm for the MMCC problem. Furthermore, if the capacities of all balls in the input are equal, then there is a polynomial time \((21, 6.47)\)-approximation algorithm.

5.2.2 The Algorithm for the EMCC Problem

Now we describe the algorithm in detail. For simplicity, at first we consider the \(d = 2\) case. Our algorithm takes an additional input – a constant \(\epsilon > 0\), and gives an \(O(\epsilon^{-8} \log(1/\epsilon))\) approximation, where each ball in the solution may be expanded by a factor of at most \(1 + \epsilon\). For the EMCC problem, the Preprocessing stage is as follows.

Lemma 9. Given a feasible LP solution \(\sigma = (x, y)\) corresponding to an EMCC instance in \(\mathbb{R}^2\), and parameters \(0 < \alpha \leq \frac{1}{2}\), and \(\epsilon > 0\), there exists a polynomial time algorithm to obtain another LP solution \(\bar{\sigma} = (\bar{x}, \bar{y})\) that satisfies Constraints 5.1-5.6 except 5.4 of MMCC-LP. Additionally, \(\bar{\sigma}\) satisfies the following properties.

1. Any ball \(B_i \in \mathcal{B}\) with non-zero \(\bar{y}_i\) is either heavy \((\bar{y}_i = 1)\), or light \(0 < \bar{y}_i \leq \alpha\).

2. For each point \(p_j \in P\), we have that

\[
\sum_{B_i \in \mathcal{L}: \bar{x}_{ij} > 0} \bar{y}_i \leq \alpha
\]

(5.9)

where \(\mathcal{L}\) is the set of light balls with respect to \(\bar{\sigma}\).

3. For any heavy ball \(B_i\), and any point \(p_j \in P\) served by \(B_i\), \(d(c_i, p_j) \leq (1 + \epsilon) \cdot r_i\).

4. For any light ball \(B_i\), and any point \(p_j \in P\) served by \(B_i\), \(d(c_i, p_j) \leq r_i\).

5. \(\text{cost}(\bar{\sigma}) = O(\epsilon^{-2} \log(1/\epsilon)) \cdot \text{cost}(\sigma)\).
Proof. As in Lemma 1, in each iteration, we pick an arbitrary point \( p_j \in P \) for which the Inequality 5.9 is not met, and consider a set \( \mathcal{L}_j \) of light balls serving \( p_j \) such that \( \alpha < \sum_{B_i \in \mathcal{L}_j} y_i \leq 2\alpha \). We select a subset of these balls, and for each such selected ball \( B_i \), we set \( y_i \leftarrow 1 \). For each ball \( B_i \) in \( \mathcal{L}_j \) which is not selected, we set \( y_i \leftarrow 0 \).

We show that the corresponding solution satisfies the desired properties. Let \( r \) be the radius of a maximum radius ball from the set \( \mathcal{L}_j \). Now all balls from the set \( \mathcal{L}_j \) contain a common point \( p_j \). Thus any point that belongs to a ball with radius less than \( r\epsilon/2 \), is within distance \((1 + \epsilon) \cdot r\) from the center of a maximum radius ball. As we are going to select such a maximum radius ball and its capacity is larger than the capacity of any ball with radius less than \( r\epsilon/2 \), we discard balls with radius smaller than \( r\epsilon/2 \). We reroute all the flow from those balls to the selected ball.

Now we divide the balls into \( O(\log(1/\epsilon)) \) classes such that the \( i^{th} \) class contains balls of radii between \( 2^{i-1}r\epsilon \) and \( 2^i r\epsilon \) for \( 0 \leq i \leq O(\log(1/\epsilon)) \). We consider each class separately and select a subset of balls from each class. Consider the \( i^{th} \) class. Note that there exists an axis-parallel square of side \( 2^{i+2}r\epsilon \) such that the centers of the balls in \( i^{th} \) class are contained in it. We subdivide this square into smaller squares, by overlaying a grid of granularity \( 2^{i-2}r\epsilon^2 \). Note that the number of smaller squares (henceforth referred to as a cell) in the larger square is \( O(\epsilon^{-2}) \). We show how to select at most one light ball from each cell.

Consider a cell from the subdivision, and let \( \mathcal{L}_j' \) be the balls in \( i^{th} \) class (with radius at least \( 2^{i-1}r\epsilon \)) whose centers belong to this cell. Now, we select a ball \( B_m \in \mathcal{L}_j' \) with the maximum radius \( r_m \) from the set \( \mathcal{L}_j' \), and reroute the flow from the other
balls to $B_m$. Since the center of $B_m$ is within distance $2^{i-1}r\epsilon^2$ from the center of any ball $B_l \in \mathcal{L}'_j$, all the points contained in any ball $B_l \in \mathcal{L}_j$ are within distance $2^{i-1}r\epsilon^2 + r_i \leq \epsilon r_m + r_m = (1+\epsilon) \cdot r_m$ from the center $c_m$ of the ball $B_m$.

Note that the capacity $U_m$ of the ball $B_m$ is at least that of the capacity of any ball from the set $\mathcal{L}'_j$, because of the monotonicity property. Furthermore, the ball $B_m$ has enough capacity to receive all the redirected flow, since

$$\sum_{x_{ij} 
 p_j \in P, B_l \in \mathcal{L}'_j : B_l \text{ serves } p_j} x_{ij} \leq \sum_{B_l \in \mathcal{L}'_j} U_l \cdot y_l \leq U_m \cdot \sum_{B_l \in \mathcal{L}_j} y_l \leq U_m \cdot 2 \cdot \alpha \leq U_m.$$  

As $\sum_{B_l \in \mathcal{L}_j} y_l > \alpha$ and we select at most $O(\epsilon^{-2} \log(1/\epsilon))$ balls in total from $\mathcal{L}_j$ the increase in cost is by at most a factor of $O(\epsilon^{-2} \log(1/\epsilon))$ by a suitable choice of $\alpha$. It is easy to verify that the other properties in the statement of the lemma are also satisfied. 

As mentioned before, the Cluster Formation stage for EMCC is exactly the same as the one for MMCC. Note that the Cluster Formation stage increases the cost of the solution only by a constant factor. We describe and analyze the Selection of Objects stage in the following lemma. The main idea remains similar to that of Lemma 9.

**Lemma 10.** There exists a scheme for the Selection of Objects stage for the EMCC problem, such that for any $\epsilon > 0$,

1. From each cluster, we choose at most $O(\epsilon^{-6})$ balls.

2. For any chosen ball $B(c_i, r_i)$ that serves a point $p_j \in P$, we have that $d(c_i, p_j) \leq (1+\epsilon) \cdot r_i$. 
3. For any chosen ball $B_i$ with capacity $U_i$, we have that $\sum_{p_j \in P} x_{ij} \leq U_i$

Proof. We show how to process each cluster $C_i$ by choosing a set of balls $\mathcal{B}_i \subseteq C_i$ of size $O(\epsilon^{-6})$, such that each ball in $\mathcal{B}_i$ is expanded by at most $1 + \epsilon$ factor. Finally, for each point $p_j \in P$ that is served by the heavy ball $B_h \in C_i$, we reroute the flow from $B_h$ to an arbitrary ball $B' \in \mathcal{B}_i$ (possibly $B_h$) such that $p_j$ is contained in $B'$. We also set $y_l \leftarrow 0$ for all balls $B_l \in C_i \setminus \mathcal{B}_i$, and $y_l \leftarrow 1$ for all balls $B_l \in \mathcal{B}_i$. The feasibility of this solution follows easily from the monotonicity property. Finally, we return $\bigcup_i \mathcal{B}_i$ over all clusters $C_i$ as the solution. It only remains to describe how to choose the set $\mathcal{B}_i$ for each cluster $C_i$.

If the cluster $C_i$ contains only the heavy ball $B_h$, we set $\mathcal{B}_i = \{B_h\}$. In this case, we do not need any expansion. Otherwise, let $r_h$ be the radius of the heavy ball $B_h$ at the center of the cluster $C_i$, and let $r_m$ be the maximum radius of any ball from the cluster $C_i$.

If $r_m \leq r_h \cdot \epsilon/2$, then we expand $B_h$ by a factor of $1 + \epsilon$, and set $\mathcal{B}_i = \{B_h\}$.

Otherwise, we consider one of the following three cases. In each case, we subdivide the enclosing square of side $4r_m$ into a grid, which is very similar to Lemma 9. Therefore, we discuss in brief the granularity of the grid and the balls that are added to the set $\mathcal{B}_i$.

1. $r_h < r_m \cdot \epsilon/4$. In this case, $B_m$ can expand by a factor of $1 + \epsilon$ and can cover the points covered by the balls in $C_i$ that have radius smaller than $r_m \cdot \epsilon/4$, and discard them. Then, we overlay a grid of granularity $r_m \cdot \epsilon^2/8$, which adds $O(\epsilon^{-4})$ balls with radius at least $r_m \cdot \epsilon/4$, to the set $\mathcal{B}_i$. 
2. \( r_m \cdot \epsilon/4 \leq r_h \leq r_m/c \), for some constant \( c > 1 \). In this case, we discard balls from \( C_i \) with radii less than \( r_h \cdot \epsilon/4 \), and then overlay a grid of granularity \( r_h \cdot \epsilon^2/8 \geq r_m \cdot \epsilon^3/32 \), which adds \( O(\epsilon^{-6}) \) balls with radius at least \( r_m \cdot \epsilon^2/(16) \), to the set \( B_i \).

3. \( r_h \geq r_m/c \) for some constant \( c > 1 \). In this case, we discard balls from \( C_i \) with radii smaller than \( r_m \cdot \epsilon/(2c) \), and then overlay a grid of granularity \( r_m \epsilon^2/(4c) \), which adds \( O(\epsilon^{-4}) \) balls with radius at least \( r_m \cdot \epsilon/(2c) \), to the set \( B_i \).

In the second and the third cases above, we also add the heavy ball \( B_h \) to the set \( B_i \), if it is not added already.

We note that Lemma 9, and Lemma 10 can be modified to work in \( \mathbb{R}^d \). In this case, the increase in the cost of solution become \( O(\epsilon^{-d}\log(1/\epsilon)) \), and \( O(\epsilon^{-3d}) \), respectively (where the constants inside the Big-Oh may depend exponentially on the dimension \( d \)). If the radii of all balls are equal, then we can improve both the bounds to \( O(\epsilon^{-d}) \), since grids of granularity \( O(\epsilon^{-1}) \) suffice. Therefore, with suitable modifications to Lemma 9, the analysis of the Cluster Formation stage from the MMCC algorithm, and Lemma 10, we have the following theorem.

**Theorem 65.** There is a polynomial time \( (O(\epsilon^{-d}\log(1/\epsilon)), 1 + \epsilon) \)-approximation algorithm for the EMCC problem in \( \mathbb{R}^d \), for any \( \epsilon > 0 \).

### 5.2.3 Soft Capacitated Version of MMCC

We remind the reader that in this variant, we are allowed to open multiple identical copies of the given ball at the same location, and each such ball has a capacity same as that of the original ball. However, we need to pay a cost of 1 for each copy. The LP corresponding to the soft capacitated version, is the same as
MMCC-LP, except that Constraint 5.6 is relaxed to simply $y_i \geq 0$. We solve this LP, and obtain an optimal solution $(x^*, y^*)$. Then, using the procedure from Lemma 1, we can ensure that the flow that each point receives from the set of non-light balls $(B \setminus \mathcal{L})$ is at least $1 - \alpha$. Then, opening $\frac{1}{1-\alpha} \lceil y_i \rceil$ identical copies of each non-light ball $B_i$ ensures that at least one demand of each point is satisfied exclusively by these balls. We now expand each of the opened balls by a factor of 3. As $y_i \geq \alpha$ for each non-light ball $B_i$, choosing $\alpha = \frac{1}{2}$ yields a simple 4-approximation for this version, where each ball is expanded by a factor of at most 3.

5.3 Hardness of Approximation
5.3.1 Hardness of MMCC

In this section, we consider the Metric Monotonic Capacitated Covering (MMCC) problem, and show that for any constant $c \geq 1$, there exists a constant $\epsilon_c > 0$ such that it is \textsc{NP}-hard to obtain a $(1 + \epsilon_c, c)$-approximation for the MMCC problem. Contrast this result with the result that follows from the reduction described in the Introduction and states that it is \textsc{NP}-hard to obtain a $(o(\log n), c)$-approximation for the MMCC for $1 \leq c < 3$ – the following construction shows that even if we relax the expansion requirement above 3, it is not possible to obtain a PTAS for this problem. To show this, we use a gap-preserving reduction from (a version of) the 3-Dimensional Matching problem.

Consider the Maximum Bounded 3-Dimensional Matching (3DM-3) problem (defined in [107]). In this problem, we are given 3 disjoint sets of elements $X, Y, Z$, with $|X| = |Y| = |Z| = N$, and a set of “triples” $T \subseteq X \times Y \times Z$, such that each
element \( w \in W := X \cup Y \cup Z \) appears in exactly 1, 2 or 3 triples of \( T \). A triple \( t = (x, y, z) \in T \) is said to cover \( x \in X, y \in Y, z \in Z \). The goal is to find a maximum cardinality subset \( M \subseteq T \) of triples that does not agree in any coordinate. Here, the elements \( U \subseteq W \) that are covered by the triples in \( M \) are said to be the *matched* elements. If \( W = U \), then the corresponding \( M \) is said to be a *perfect matching*. We have the following result for the 3DM-3 problem from Petrank [107].

Lemma 11 (Restatement of Theorem 4.4 from [107]). There exists a constant \( 0 < \beta < 1 \), such that it is \( \mathbf{NP} \)-hard to distinguish between the instances of the 3DM-3 problem in which a perfect matching exists, from the instances in which at most \( 3\beta N \) elements are matched.

Reduction from 3DM-3 to MMCC  Given an instance \( I \) of 3DM-3 problem, we show how to reduce it to an instance \( I' \) of the MMCC problem. Recall that in the version of the MMCC problem, we are allowed to expand the balls in the input by a constant factor \( c \geq 1 \).

First, we show how to construct the metric space \((P \cup C, d)\) for the MMCC instance \( I' \), that is induced by the shortest path metric on the following graph \( G = (P \cup C, E) \). Recall that \( C \) is the set of centers, and \( P \) is the set of points that need to be covered by the balls centered at centers in \( C \). Before describing this graph, we construct some objects that will be useful in the description.

Consider a vertex \( c_1 \in C \), that is connected to 4 other vertices \( p_1, \ldots, p_4 \in P \) (for convenience we refer to them as left, right, top, bottom vertices respectively) by
Figure 5.1. The element $x$ belongs to three triples, whose gadgets are attached to the element gadget of $x$ at three locations. The triple gadgets are also attached to other element gadgets. Note that each ball contains 4 points, but has the capacity of only 3.

an edge of weight 1. We also add a ball of radius 1 at the center $c_1$. For convenience, we refer to this object (the 5 vertices and the ball) as a small cluster, and the ball as a small ball. Similarly, if we the radius and the edge weights of the ball are $c$, then we refer to such an object as a large cluster, and the ball as a large ball.

Now, consider $p = \lceil \frac{c(c+1)}{2} \rceil + 1$ copies of small clusters, numbered $\kappa_1, \cdots, \kappa_p$. For each $1 < i \leq p$, we “glue” small clusters $\kappa_{i-1}$ and $\kappa_i$, by setting the right vertex of $\kappa_{i-1}$ equal to the left vertex of $\kappa_i$. This forms an object in which two consecutive clusters share exactly one vertex. We refer to this object as a small chain. For a particular small chain, we refer to its $\kappa_1$ as its leftmost small cluster, and to $\kappa_p$ as its rightmost small cluster. Now, consider a big cluster that is “glued to” two small chains on two sides. That is, the left vertex of the big cluster is the same as the right vertex of the $\kappa_p$ of a small chain (the left half), and the right vertex of the big cluster is the same as the left vertex of the $\kappa_1$ of another small chain (the right half). We call this object (which contains $2p$ small clusters and balls, and 1 large cluster and
ball) a large chain.

We now describe the element gadget. In an element gadget we consider two large chains \( ch_1, ch_2 \) that are glued together, such that the they share a common small cluster. That is, the rightmost small cluster of the right half of \( ch_1 \) is the same as the leftmost small cluster of the left half of \( ch_2 \). Denote this common cluster by \( \kappa \).

The respective 3 bottom points of a) The cluster \( \kappa \), b) The leftmost small cluster of the left half of \( ch_1 \), and c) The rightmost small cluster of the right half of \( ch_2 \) are referred to as ideal points. Note that each element gadget contains \( 4p + 1 \) balls and \( 3(4p + 1) + 1 \) vertices.

For each element \( w \in W \), we add an element gadget. Now we describe the triple gadget. This gadget is similar to a large cluster, the only difference is that in addition to the central point \( c_1 \), it contains only 3 other points \( p_1, p_2, p_3 \). We add a triple gadget for each triple \( t = (x, y, z) \in T \). Now, for each such triple \( t = (x, y, z) \), we identify \( p_1, p_2, p_3 \) with one of the ideal points from the gadgets of \( x, y, z \) respectively. Here, we ensure that if an element is contained in multiple triples, then a different ideal point is assigned to each triple. Finally, we set the capacity of each ball to be 3.

The total number of balls in all the element gadgets is \( \mathfrak{B} := 3N \cdot (4p + 1) \). We refer to these balls as element balls. Similarly, the total number of balls in all the triple gadgets is \( |T| \), which we refer to as triple balls. As mentioned above, the metric is induced by the graph \( G = (P \cup C, E) \) as described above.

This completes the description of the instance \( I' \) of the MMCC problem. It is easy to see that the instance \( I' \) has not only monotonic capacities, but even uniform
capacities. It is also worth highlighting that there are only two distinct radii in the instance $I'$. We are able to show that such a restricted version of the MMCC problem remains APX hard, even when we are allowed to expand the balls by a constant $c \geq 1$.

**Claim 66.** Consider the instance $I'$ of MMCC problem, that is obtained from an instance $I$ of 3DM-3, using the above reduction. We can always convert a solution to $I'$, where a selected ball may be expanded by a factor at most $c \geq 1$, to another feasible solution in polynomial time, where all element balls in the element gadgets are selected. Furthermore, we can ensure that in this assignment, every selected ball serves a point that is contained in it, (without any expansion).

**Proof.** First, notice that in each element gadget, the number of points is equal to the total capacity of all the balls in the element gadget, plus 1.

Now, consider a solution that does not satisfy the required property. Initially, we discard any balls in the solution if they do not use any of their capacity. Now, consider an element gadget from which a ball is not chosen in the solution. Note that this unchosen ball cannot be the central **large ball** of any of its chains, because no other ball can cover the top and bottom points corresponding to it, even after expanding by a factor of $c$. We can also infer from this that, all such **large balls** have at most 1 capacity that can be used for serving other points. Without loss of generality, we assume that this remaining capacity is assigned to either left or right point, since one can always find such an assignment.

Now, any unchosen ball has to be a **small ball**. Without loss of generality, we can assume that 3 of the 4 points that it contains, are not served by any other
small ball, since we can always reassign capacities to ensure that that is not the case. Therefore, the points contained in it must be covered by triple balls, such that the corresponding triples cover the corresponding element. However, note that because of the existence of the large ball at the center of a small chain, two different triple balls cannot serve points from a common small ball. Therefore, all 3 points must be served by a single triple ball. But any triple ball also has a capacity of 3, and therefore, we can swap out the triple ball for this small ball, without increasing the cost of the solution.

By repeating this process, we can include all element balls, by swapping out some of the triple balls if necessary. Now at this point, all element balls, as well as enough triple balls are included in the solution, such that the solution is feasible. Now, we assign the capacities of the selected triple balls to the corresponding ideal points. It is easy to see that in each element gadget, at least one ideal point is served by the corresponding triple ball, and therefore, there exists a simple capacity assignment scheme to ensure that each element ball serves only the points contained in it. □

**Lemma 12.** Consider an instance $I$ of 3DM-3, and let $I'$ be the reduced MMCC instance. In the 3DM instance $I$, $M \subseteq T$ is a minimum size set of triples that covers all the elements of $W$, with $|M| = K$, if and only if the minimum cost of a solution to $I'$, wherein the balls may be expanded by up to a factor $c \geq 1$, is $2B + K$.

**Proof.** We first show that given a minimum size cover $M$ of size $K$, of the elements of $W$, how to select $2B + K$ balls in the instance $I'$. Firstly, for all triples in $M$, include the corresponding balls in the triple gadget in the solution, which accounts
for $K$ balls. Now, for such ball in a triple gadget, assign the capacity to serve the *ideal points* of the corresponding element gadgets. Now, the number of points in an element gadget that are not served is at most the total capacity of each element gadget. Therefore, by selecting all $B$ balls, each point can be covered by one of the selected balls that it is contained in. Now we prove that there is no solution with a lesser cost.

Consider a minimum cost solution to $I'$, wherein the balls may be expanded by a factor at most $c \geq 1$. Then, we use Claim 66 on to obtain another solution of at most the same size, in which all element balls are selected, and no ball is expanded. Now, this solution must have the same size, because the cost of an optimal solution where the balls may not be expanded, is at least the minimum cost of a solution where the balls may be expanded. Therefore, the costs of optimal solutions to the original and the relaxed version of the MMCC instance are equal for the instance $I'$.

In the new solution, each selected triple ball serves some of the ideal points. The number of selected triple balls must be at least $K$, because otherwise the set of selected triple balls corresponds to a cover of the instance $I$ of size less than $K$, which is a contradiction.

Now assume that the minimum cost solution to the instance $I'$, where the balls may be expanded by a factor $c \geq 1$, is $B + K$. Using the argument from the previous paragraph, we can obtain a cover of size $K$ for the 3DM-3 instance $I$. To show that it is optimal, assume for contradiction, a smaller size solution, and use the argument from the first paragraph to obtain a solution to the MMCC instance $I'$ of
size smaller than $\mathfrak{B} + K$, but where the balls are not expanded. Now, this contradicts
the optimality of the initial solution, because as argued before, the optimal costs
of the strict and relaxed versions of the MMCC problem are equal for the instance
$I'$.

Using this lemma, we obtain the following two corollaries, that show the gap
between the instances that have a perfect matching, and those that do not have.

**Corollary 1.** If there exists a perfect matching in the 3DM-3 instance $I$, then the
corresponding MMCC instance $I'$ has an optimal solution of size exactly $\mathfrak{B} + N$.

**Corollary 2.** If in the 3DM-3 instance $I$, the maximum number of elements that can
be matched is at most $3\alpha N$ ($0 < \alpha < N$), then the minimum cost of any solution
where the balls may be expanded by a factor $c \geq 1$, in the corresponding MMCC
instance $I'$, is at least $\mathfrak{B} + \alpha N + \frac{3(1-\alpha)N}{2} = \left(1 + \frac{1-\alpha}{3p+1}\right) \cdot (\mathfrak{B} + N)$.

**Proof.** In the 3DM-3 instance $I$, the maximum number of elements that can be
matched is at most $3\alpha N$, for some $0 < \alpha < 1$. If $M \subseteq T$ is a minimum size set of
triples that covers all the elements in $W$, then we first show that $|M| \geq \alpha N + \frac{3(1-\alpha)N}{2}$.

Let $M = M_1 \cup M_2$, where $M_1$ is the maximal set of triples such that each
set covers 3 distinct elements, and $M_2$ is the remaining triples. By assumption, the
number of matched elements is at most $3\alpha N$, and therefore, $|M_1| \leq \alpha N$. The number
of elements left to be covered by $M_2$ is $3N - 3|M_1|$, and since each triple in $M_2$ covers
at most 2 new elements each, $|M_2| \geq \frac{3}{2}(N - |M_1|)$. Therefore, since $M_1$ and $M_2$ are
disjoint, $|M| = |M_1| + |M_2| \geq \alpha N + \frac{3(1-\alpha)N}{2}$.
Now, using Lemma 12, we conclude that the cost of an optimal solution to $I'$ is at least $\mathcal{B} + \alpha N + \frac{3(1-\alpha)N}{2}$. It is easy to verify that the previous quantity is exactly equal to \( \left(1 + \frac{1-\alpha}{8(3p+1)}\right) \cdot (\mathcal{B} + N) \), recalling that $\mathcal{B} = 3N \cdot (4p + 1)$.

Now, from Lemma 11, Corollary 1, and Corollary 2, we obtain the following APX-hardness result.

**Theorem 67.** For any constant $c \geq 1$, there exists a constant $\epsilon_c > 0$ such that it is NP-hard to obtain a $(1 + \epsilon_c, c)$-approximation for the uniform capacitated version of MMCC.$^1$

### 5.3.2 Hardness of MMCC with Weights

We consider a generalization of the Metric Monotonic Capacitated Covering (MMCC) problem. Like in the MMCC problem, here also we are given a set of balls $\mathcal{B}$ and a set of points $P$ in a metric space. Each ball has a capacity, and the capacities of the balls are monotonic. Additionally, each ball has a non-negative real number associated with it which denotes its weight. The weight of a subset $\mathcal{B}'$ of $\mathcal{B}$ is the sum of the weights of the balls in $\mathcal{B}'$. The goal is to find a minimum weight subset $\mathcal{B}'$ of $\mathcal{B}$ and compute an assignment of the points in $P$ to the balls in $\mathcal{B}'$ such that the number of points assigned to a ball is at most its capacity. We refer to this problem as Metric Monotonic Capacitated Covering with Weights (MMCC-W). In the case where all balls have the same radius and the same capacity one can get an $(1, O(1))$-approximation for MMCC-W by using a constant approximation algorithm

$^1$\(\epsilon_c\) depends inversely on $c^2$. 
for the Budgeted Center problem [9]. However, as we prove, there are instances of MMCC-W that consist of balls of only two distinct radii for which it is \( \text{NP-hard} \) to obtain a \( (o(\log |P|), c) \)-approximation for any constant \( c \).

The reduction is from the Set Cover problem. Recall that in Set Cover we are given a set system \((X, \mathcal{F})\) with \( n = |X| \) elements and \( m = |\mathcal{F}| \) subsets of \( X \). For each element \( e_i \in X \), let \( m_i \) be the number of sets in \( \mathcal{F} \) that contain \( e_i \). Also for each set \( X_j \in \mathcal{F} \), let \( n_j \) be the number of elements in \( X_j \). Note that \( \sum_{i=1}^{n} m_i = \sum_{j=1}^{m} n_j \).

Given any instance \( I \) of Set Cover we construct an instance \( I' = (P, B) \) of MMCC-W.

Let \([t] = \{1, \ldots, t\}\). Fix a constant \( c \), which is the factor by which the balls in the solution are allowed to be expanded, and a real \( \alpha > 0 \). Let \( N = \max\{m, n\} \) and \( M = c^{1+1/\alpha} N^{2/\alpha} \). \( P \) contains \( M \cdot m_i \) points corresponding to each element \( e_i \). To describe the distances between the points we define a weighted graph \( G \) whose vertex set is \( P \cup C \), where \( C \) is the set of centerpoints of the balls in \( B \). The graph contains a set \( V_i \) of \( 2M \cdot m_i - 1 \) vertices corresponding to each element \( e_i \). The subgraph of \( G \) induced by the vertices of \( V_i \) is a path of \( 2M \cdot m_i - 1 \) vertices. We denote this path by \( \pi_i \). Refer a degree 1 vertex on \( \pi_i \) as its 1\textsuperscript{st} vertex, the vertex connected to it as the 2\textsuperscript{nd} vertex and in general for \( i \geq 2 \), the index of the vertex connected to the \( i \textsuperscript{th} \) vertex other than the \((i - 1)\textsuperscript{th}) \) vertex is \( i + 1 \). The odd indexed vertices on this path belong to \( P \), and the even indexed vertices belong to \( C \). Thus \( M \cdot m_i \) (resp. \( M \cdot m_i - 1 \)) vertices of the path are in \( P \) (resp. \( C \)). The weight of each path edge is set to be \( cR/M \), where \( R \) is a positive real. Corresponding to each set \( X_j \in \mathcal{F} \), \( G \) contains a vertex \( u_j \) that belongs to \( C \). Now for each \( e_i \in X \), consider a one-to-one mapping
from the set \([m_i]\) to the set of \(m_i\) subsets of \(X\) that contain \(e_i\). For \(1 \leq i \leq m_i\), we connect the \(((i - 1) \cdot M + 1)th\) vertex of \(\pi_i\) to the vertex corresponding to the set \(f(i)\) by an edge of weight \(R\). Note that for any set \(X_j \in \mathcal{F}\), \(u_j\) gets connected to \(n_j\) vertices of \(G\). This concludes the description of \(G\).

We consider the metric space \((P \cup C, d)\) for \(I'\), where \(d\) is the shortest path metric on \(G\). Now we describe the set of balls in \(I'\). For each \(X_j \in \mathcal{F}\), we add the ball \(B(u_j, R)\) to \(\mathcal{B}\) and set its capacity to \(n_j\). Note that \(B(u_j, R)\) contains exactly \(n_j\) points of \(P\). For each \(e_i \in X\), now consider the set of vertices \(V_i\). For each point \(p\) of \(C \cap V_i\), we add the ball \(B(p, cR/M)\) to \(\mathcal{B}\) and set its capacity to 1. We note that \(B(p, cR/M)\) contains 2 points. The balls in \(\mathcal{B}\) have only two distinct radii. It is not hard to see that the capacities of these balls are monotonic w.r.t their radii. We set the weight of each ball \(B(p, r)\) to \(r^{1+\alpha}\).

**Lemma 13.** The elements in \(X\) can be covered by \(k\) sets of \(\mathcal{F}\) iff there is a solution to MMCC-W for the instance \(I'\) with weight at most \((k + 1) \cdot R^{1+\alpha}\) where the balls in the solution can be expanded by a factor of \(c\).

**Proof.** Let \(X\) can be covered by a collection \(\mathcal{F}'\) of \(k\) sets. We construct a feasible solution \(\mathcal{B}' \subseteq \mathcal{B}\) to MMCC-W whose weight is at most \((k + 1) \cdot R^{1+\alpha}\). For each set \(X_j \in \mathcal{F}'\), we add the ball \(B(u_j, R)\) to \(\mathcal{B}'\). We assign the \(n_j\) points in \(B(u_j, R)\) to it. Now for each \(e_i \in X\), we add balls in the following manner. Note that at least one point of \(V_i \cap P\) has already been assigned to a ball in \(\mathcal{B}'\). Now for each point \(p \in C\) on \(\pi_i\), we add the ball \(B(p, cR/M)\) to \(\mathcal{B}'\). As one point of \(V_i \cap P\) is already assigned to a ball in \(\mathcal{B}'\), only \(M \cdot m_i - 1\) points of \(V_i \cap P\) are left for assigning. As we have
chosen $M \cdot m_i - 1$ balls each of capacity 1 corresponding to $\pi_i$ one can easily find a valid assignment of these points. Thus $\mathcal{B}'$ is a feasible solution to MMCC-W. Now the weight of the balls selected w.r.t. the sets is $k \cdot R^{1+\alpha}$. The weight of the balls chosen w.r.t. each path $\pi_i$ is at most $M \cdot m_i (cR/M)^{1+\alpha}$. The total weight of the balls w.r.t. all such paths is at most

$$n \cdot M \cdot m \cdot (cR/M)^{1+\alpha} \leq R^{1+\alpha}$$

Thus the weight of $\mathcal{B}'$ is at most $(k + 1) \cdot R^{1+\alpha}$.

Now suppose there is a solution $\mathcal{B}'$ to MMCC-W with weight at most $(k + 1) \cdot R^{1+\alpha}$. The total number of points in $P$ is $\sum_{i=1}^{n} M \cdot m_i > \sum_{j=1}^{m} n_j$. Now the total capacities of the balls w.r.t. the sets is $\sum_{j=1}^{m} n_j$. Thus there must be at least one ball in $\mathcal{B}'$ which is w.r.t. a path. Also the weight of the ball w.r.t. each set is $R^{1+\alpha}$. Thus there must be at most $k$ balls w.r.t. the sets in $\mathcal{F}$ that are in $\mathcal{B}'$. We consider the collection $\mathcal{F}' \subseteq \mathcal{F}$ of sets corresponding to these balls (at most $k$ in number). We claim that $\mathcal{F}'$ covers all elements of $X$. Consider any element $e_i \in X$ and the path $\pi_i$ corresponding to it. Note that $\pi_i$ contains $M \cdot m_i$ points of $P$. Consider one such point $p$ and any center $p'$ on the path $\pi_j$ where $i \neq j$. Now distance between the $p'$ and $p$ is at least $2R$ and thus even after $c$ factor expansion the ball $B(p', cR/M)$ cannot contain $p$. Now consider a center point $u_t$ such that $e_i \notin X_t$. Then due to the construction the distance between $p$ and $u_t$ is at least $3R + (cR/M) \cdot 2M > cR$. Thus even after $c$ factor expansion the ball $B(u_t, R)$ cannot contain $p$. Hence $p$ must be assigned to either a ball corresponding to $\pi_i$ or a ball corresponding to a set $X_j$ that contains $e_i$. Now there are only $M \cdot m_i - 1$ balls w.r.t. $\pi_i$ in $\mathcal{B}$ each of whose
capacity is 1 and thus there must be a point $p \in P$ lying on $\pi_i$ that is assigned to a ball corresponding to a set $X_j$. It follows that $e_i \in X_j$. Thus $\mathcal{F}'$ covers all the points of $X$.

As Set Cover is $\text{NP}$-hard to approximate within a factor of $o(\log n)$, from Lemma 13, we obtain the following theorem.

**Theorem 68.** For any constant $c \geq 1$, there exists a constant $c' > 0$, such that it is $\text{NP}$-hard to obtain a $(c' \log |P|, c)$-approximation for MMCC-W. This result holds even for the particular weight function, where the weight of a ball is equal to a constant power of its original radius.
CHAPTER 6
CONCLUSION AND OPEN QUESTIONS

6.1 Introduction

In the previous chapters of this thesis, we have discussed a handful of results we have obtained in the areas of clustering and covering. In this chapter, we pose a collection of open problems, again in the domains of clustering and covering, which we would like to consider as future work. Each of the problems we are going to pose belongs to one of the following three topics - $k$-clustering, capacitated covering, and Non-Uniform-$k$-center. We start our discussion with the $k$-clustering problem.

6.2 $k$-clustering

We are mainly interested in the case where $\alpha > 1$. Recall that, for some constant $c$, there is a $c^\alpha$-approximation for the metric version of the problem using the primal-dual method, which is the best known approximation for the problem in polynomial time [37]. In fact, currently this is the best known bound even in the Euclidean plane. Note that there is an algorithm for the Euclidean case that yields a $(1 + \varepsilon)$-approximation, but might use as many as $(1 + \varepsilon)k$ clusters. On the other hand, no inapproximability result is known even in the arbitrary metric case if $\alpha = o(\log n)$, where $n$ is the number of points. The problem is known to be $\mathsf{NP}$-hard in the Euclidean plane. For the related Minimum Cost Covering problem, we have presented a quasi-polynomial time $(1 + \varepsilon)$-approximation (QPTAS). We note that, there is already a polynomial time $(1 + \varepsilon)$-approximation (PTAS) for the Euclidean
version of this problem [26]. In the light of the above discussion, we pose the following open problems (see also Table 6.1).

<table>
<thead>
<tr>
<th>Problem</th>
<th>Known approx.</th>
<th>Open questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metric $k$-clustering</td>
<td>$c^\alpha$</td>
<td>APX-hardness and $&lt; c^\alpha$-approx.</td>
</tr>
<tr>
<td>Euclidean $k$-clustering</td>
<td>$c^\alpha$</td>
<td>PTAS</td>
</tr>
<tr>
<td>Metric Min Cost Covering</td>
<td>QPTAS and $3^\alpha$</td>
<td>$&lt; 3^\alpha$-approx.</td>
</tr>
</tbody>
</table>

Table 6.1. $k$-clustering: known results and open questions

1. Is the metric $k$-clustering problem APX-hard for $\alpha = o(\log n)$, i.e., is it hard to approximate in polynomial time within a factor of $(1 + \varepsilon)$ for some $\varepsilon > 0$?

2. Does there exist a PTAS for Euclidean $k$-clustering?

3. Is it possible to find a polynomial time approximation algorithm for metric $k$-clustering (resp. Minimum Cost Covering) with approximation factor better than $c^\alpha$ (resp. $3^\alpha$)?

Next, we move on towards our next topic: capacitated covering.

### 6.3 Capacitated Covering

Recall that, we have obtained a (21, 9) bi-criteria approximation for the Metric Capacitated Covering (MCC) problem. On the other hand, for any $c \geq 1$, there exists a constant $\varepsilon_c$, such that it is NP-hard to obtain a $(1 + \varepsilon_c, c)$ bi-criteria approximation
for MCC, i.e., even if we are allowed to expand the balls by an arbitrary constant factor, it is not possible to find a near optimal solution efficiently. As we pointed out before, if we are allowed to expand the balls by less than 3 factor, then it is \( \text{NP} \)-hard to obtain an \( f \)-approximation for \( f < o(\log n) \). In contrast, in the fixed dimensional Euclidean space (the ECC problem), only a \( 1 + \varepsilon \)-factor expansion is sufficient to obtain a constant \( (O(\varepsilon^{-4d} \log(1/\varepsilon))) \) approximation for any \( \varepsilon > 0 \). Also the only inapproximability result known for ECC is an \( \text{APX} \)-hardness result in \( \mathbb{R}^3 \), which follows from the \( \text{APX} \)-hardness of the uncapacitated version of the problem. On the other hand, the best known approximation factor without expansion is only \( O(\log n) \). Thus, there is a huge gap between the upper and the lower bounds on the approximation factor for ECC. In view of our discussion, we pose the following open questions (see also Table 6.2).

<table>
<thead>
<tr>
<th>ECC version</th>
<th>Known approx.</th>
<th>Open questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>( O(\log n) ) and ( (O(\varepsilon^{-4d} \log(1/\varepsilon)), 1 + \varepsilon) ) and ( \text{APX} )-hard</td>
<td>( o(\log n) )-approx. and ( (1 + \varepsilon, 1 + \varepsilon) )-approx.</td>
</tr>
<tr>
<td>( \mathbb{R}^2 )</td>
<td>( O(\log n) )</td>
<td>( \text{APX} )-hard/PTAS</td>
</tr>
<tr>
<td>Unit radii in ( \mathbb{R}^2 )</td>
<td>( O(\log n) ) and ( (1, O(1)) )</td>
<td>( \text{APX} )-hard/PTAS</td>
</tr>
</tbody>
</table>

Table 6.2. ECC: known results and open questions

1. Is it possible to find a sublogarithmic approximation for ECC?

2. Is it possible to find a \( (1 + \varepsilon, 1 + \varepsilon) \) bi-criteria approximation for ECC for any
3. Is ECC APX-hard for dimension 2?

4. Is it possible to obtain a PTAS for ECC in $\mathbb{R}^2$ when all the balls have unit radius?

Next, we consider the Non-Uniform-$k$-center problem.

### 6.4 Non-Uniform-$k$-center

Remember that, we designed a constant approximation for the NUkC problem with three radii classes. However, there we use about $2k_2$ and $2k_3$ balls of radius $r_2$ and $r_3$, respectively. Our result is based on the LP aware reduction in [33]. Using this reduction, we obtain a tree with height 3 and a feasible fractional LP solution to the RMFC-T problem on this tree instance. Using a randomized rounding scheme, we show how to round the solution to obtain a feasible integral solution using about $2k_2$ and $2k_3$ firefighters at level 2 and 3 of the tree, respectively. We also show tree instances that admit feasible fractional solution, but if we are allowed to use at most $k_1$ firefighters at level 1 and less than $2k_2$ firefighters at level 2, then any integral solution to those instances needs to use at least $(2 - \epsilon)k_3$ firefighters at level 3. Thus, using the LP aware reduction in [33], one cannot hope to find any true approximation. We also showed that the problem with three radii classes becomes exactly solvable in polynomial time under perturbation resilience assumption. However, for the general version, getting a constant approximation for the problem still remains an open question. We proved that the Euclidean version of the problem is also as hard to
approximate as the general version. Moreover, we showed that the perturbation resilience assumption does not make the problem any easier in general. In consideration of the above discussion, we pose the following open questions (see also Table 6.3).

<table>
<thead>
<tr>
<th>NUkC version</th>
<th>Known approx.</th>
<th>Open questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t \ (\geq 3)$ radii classes</td>
<td>$(O(1), O(1))$</td>
<td>$O(t)$-approx.</td>
</tr>
<tr>
<td>2 radii classes</td>
<td>$1 + \sqrt{5}$</td>
<td>$&lt; 1 + \sqrt{5}$-approx.</td>
</tr>
</tbody>
</table>

Table 6.3. NUkC: known results and open questions

1. Is it possible to obtain a constant approximation for NUkC when the number of distinct radii is 3 or a constant in general?

2. Is it possible to obtain a better than $1 + \sqrt{5}$-approximation for NUkC when the number of distinct radii is at most 2?
REFERENCES


