# APPROXIMATION ALGORITHMS FOR PROXIMITY AND CLUSTERING PROBLEMS

**YOGISH SABHARWAL**


**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**
**INDIAN INSTITUTE OF TECHNOLOGY DELHI**
**DECEMBER 2006**

# Approximation Algorithms for Proximity and Clustering Problems

by

**Yogish Sabharwal**
Department of Computer Science and Engineering

*Submitted*
*in fulfillment of the requirements of the degree of*
Doctor of Philosophy

to the


**Indian Institute of Technology Delhi**
**December 2006**

# Certificate

This is to certify that the thesis titled “Approximation Algorithms for Proximity and Clustering Problems” being submitted by Yogish Sabharwal to the Indian Institute of Technology Delhi, for the award of the degree of Doctor of Philosophy, is a record of bona-fide research work carried out by him under our supervision. In our opinion, the thesis has reached the standards fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree/diploma.

Sandeep Sen
Department of Computer Science and Engineering
Indian Institute of Technology Delhi
New Delhi 110 016

Amit Kumar
Department of Computer Science and Engineering
Indian Institute of Technology Delhi
New Delhi 110 016

In loving memory of my father

## Acknowledgements

I thank my advisor Sandeep Sen, for his continuous support during my Ph.D. program. Without his encouragement and constant guidance, I could not have finished this thesis. He showed me different ways to approach a research problem. He was always there to meet and talk about my ideas, to proofread and mark up my papers and chapters, and to ask me good questions to help me think through my problems.

I thank my advisor Amit Kumar, for his continuous support and encouragement. His invaluable advice, discussions, comments and suggestions helped me complete this thesis. I have learnt a lot about writing academic papers from him.

I am forever indebted to my wife Monica, and my son Hriday, for their understanding, endless patience and encouragement when it was most required.

I would also like to thank my in-laws, sisters and mother for taking care of my personal responsibilities when I was busy working on my thesis. Special thanks to my sister Anjana and my mother-in-law for taking extra-special care of my son.

I would like to thank the management at IBM India Research Lab for supporting me during these years. Special thanks to Rahul Garg from whom I learnt to be persistent to accomplish any goal and who made me realize that I had more in me than I thought myself. I would also like to thank my friends, Vinayaka Pandit and Raghav Singh for their invaluable encouragement.

Finally, I would like to thank Rashid Ansari, Ajit Jain, Dushyant Thakor, Parijat A. Sharma, Sandeep Mamgain and T. S. Mahesh who have had a positive impact on me at some point of time during the course of my life.

## Abstract

Proximity problems refer to a class of geometric problems that involve the notion of a distance between points in a $d$-dimensional space. These include, amongst others, nearest neighbor searching and clustering problems. In nearest neighbor searching, given a set of data points in $d$-dimensional Euclidean space, we are interested in suitably pre-processing these points and constructing a data structure such that given a query point we can efficiently find its closest point in this point set. In clustering problems, the goal is to group points, based on some measure of similarity between the points. These problems are of wide interest in both Euclidean spaces as well as metric spaces.

Our results are as follows:

- For the nearest neighbor searching problem, we present an algorithm that constructs an approximate Voronoi diagram of size $O(\frac{n}{\varepsilon^d}\log n/\varepsilon)$ that can be used to answer nearest neighbor queries in time $O(\log n/\varepsilon)$. The pre-processing time required to create the approximate Voronoi diagram is $O(\frac{n}{\varepsilon^d}\log^2 n/\varepsilon)$. This improves on previously known results based on construction of approximate Voronoi diagrams, in terms of space requirement and pre-processing time, by a factor of $\log n$.

- We present a general framework for developing linear time $(1+\varepsilon)$ approximation algorithms for a class of clustering problems. We show that the $k$-means, $k$-median and discrete $k$-means problems belong to this class of clustering problems. We therefore obtain $(1+\varepsilon)$-approximation algorithms for these problems using our framework. These are the first algorithms for these problems that run in time linear in the size of the input ($nd$ for $n$ points in $d$ dimensions) for fixed values of $k$. All these algorithms succeed with constant probability. We also present an algorithm for approximating the 1-median of a set of points in constant time. The previous best known algorithm took time linear in the number of points.

- Finally, we consider a generalized version of the $k$-median problem in metric spaces, called the priority $k$-median problem and show that there exists a polynomial time constant factor approximation algorithm for this problem when there are two priorities. We also show that the natural integer program for the problem has an arbitrarily large integrality gap when there are four or more priorities.

# Contents

# List of Figures

# Chapter 1

# Introduction

*Proximity* problems form an important class of problems in computational geometry. Loosely speaking, *proximity* problems refer to a class of geometric problems which involve the notion of a distance between points in a $d$-dimensional space. These include, amongst others, *nearest neighbor searching* and *clustering* problems. *Clustering* of a group of data items into similar groups is a widely studied class of problems. These problems have been widely studied in both geometric settings as well as metric spaces.

One of the most fundamental proximity problems is to build a data structure on a set of points $P$, that supports *nearest neighbor searching* efficiently. This problem has satisfactory solutions when $P$ is planar but becomes non-trivial in higher dimensional Euclidean space – even in three dimensions [7, 29]. This problem has diverse applications in many fields including pattern recognition, management of multimedia data and vector compression [31, 21, 58, 56, 57, 32]. The most general approach to nearest neighbor searching is to build a *Voronoi diagram* of $P$ (denoted as $Vor(P)$) which is a partition of the space into cells such that a cell consists of all points that are closer to a specific point of $P$ than any other point of $P$. *Voronoi diagrams* are fundamental computing tools in computational geometry that have many other applications, including clustering, motion planning, learning, surface reconstructions etc. Generalization of Voronoi diagrams to objects beyond point sets are also equally useful (refer to Aurenhammer [8] for a detailed survey). Despite its numerous applications, *Voronoi diagrams* suffer from the drawback of high structural complexity (and consequent computational bottleneck). It is known that the worst case complexity in $\mathbb{R}^d$ is $\theta(n^{\lceil d/2 \rceil})$ with constants exponential in $d$. Therefore building and storing these diagrams becomes computationally infeasible with growing dimensions. As a result, alternate solutions for nearest neighbor searching

in higher dimensions have been a hot topic of research in recent years ([4, 20, 5, 40, 47, 38]). Since the exact problem seems intimately related to the complexity of *Voronoi diagrams*, the related relaxed problem of *approximate nearest neighbor searching* has gained a lot of importance and has also shown promise of practical solutions.

Another widely studied proximity problem in computer science is that of *clustering* a group of data items into similar groups. *Clustering* has applications in a variety of areas, for example, data mining, information retrieval, image processing, and web search ([12, 24, 60, 30, 22, 61, 48, 25, 11, 64, 63]). Given the wide range of applications, many different definitions of *clustering* exist in the literature ([27, 10]). Most of these definitions begin by defining a notion of distance (dissimilarity) between two data items and then try to form clusters so that data items with small distance between them get clustered together. Often, *clustering* problems arise in a geometric setting, i.e., the data items are points in a high dimensional Euclidean space. In such settings, it is natural to define the distance between two points as the Euclidean distance between them. Two of the most popular versions of *clustering* are the $k$*-means clustering* problem and the $k$*-median clustering* problem. Given a set of points $P$, the $k$*-means clustering* problems seeks to find a set $K$ of $k$ centers, such that the sum of the squares of the distances of the points to their closest centers is minimized, whereas the $k$*-median clustering* problems seeks to find a set $K$ of $k$ centers, such that the sum of the distances of the points to their closest centers is minimized. Note that the points in $K$ can be arbitrary points in the Euclidean space. Clearly, in an optimal solution, each point in $P$ gets assigned to the closest center. We can think of the points that get assigned to the same center, form a cluster. These problems are NP-hard for even $k = 2$ (when dimension is not fixed) [26]. Interestingly, the center in the optimal solution to the 1-*mean clustering* problem is the same as the center of mass of the points. However, in the case of the 1-*median clustering* problem, also known as the Fermat-Weber problem, no such closed form is known. There exist variations to these clustering problems, for example, the discrete versions of these problems, where the centers that we seek are constrained to lie on the input set of points. Applications of clustering problems often involve mapping subjective features to points in the Euclidean space. Since there is an error inherent in this mapping, finding an approximate solution does not lead to a deterioration in the solution for the actual application.

Often, in the $k$-median problem, there are limited locations where a center may be opened and these may be different from the points that are to be assigned to the centers. The points that are to be assigned to the centers are referred to as demands and the candidate locations for centers are called facilities. The definition of $k$-median clustering makes a crucial underlying assumption : all

facilities and demands are of the same *kind*. An interesting generalization, that has not been studied before, is to allow the demands and facilities to be of different priorities. Let us see a few motivating examples :

- There is a retail chain store and it wants to open several stores in a city. But because of various restrictions (space, regulations, etc.) there are some locations where it can open small stores, and there are locations where it can open bigger stores and hence sell more products. Now customers can be different kinds. Those having lower income may be happy with small stores, but those with more lavish lifestyle may prefer to go to bigger stores only.

- In planning emergency evacuation plan for a city, the authorities want to build locations from which people can be evacuated. But they want to build better facilities for evacuating important people, like the mayor of the city. But only some places in the city will be well equipped to provide such facilities, e.g., a low population zone, or a port. For evacuating other people, they may have more options where to build the facilities.

There is a common theme in both the examples. The facilities that can be built are of different *grades*. A grade 1 facility is better than a grade 2 facility and so on. The demands are of different kinds as well. Some demands may be happy with any facility, but others may require facilities of a certain grade or better only. Motivated by the discussion above, we formulate the *$k$-median problem with priorities*, in which demands and facilities have different priorities and a demand can then only be assigned to a facility of its priority or better. A closely related problem is the *facility location problem* where we wish to locate facilities to service a set of demands. This problem has been widely studied in computer science and operations research communities [52, 51]. The tradeoff involved in such problems is the following – we would like to spend as little in opening new facilities as possible, but the clients should not be located too far from the nearest facility. The *facility location problem* balances the two costs by minimizing (weighted) average of the cost of opening facilities and the distances demands have to travel to the nearest open facility. The variant of the *facility location* problem where the facilities and demands have priorities has been studied before. Shmoys et al. [59] presented a constant factor approximation algorithm for this problem.

We now discuss in detail the problems considered in this thesis and highlight our contributions.

## 1.1 Nearest Neighbor Searching and Voronoi Diagrams

Let $P$ be a set of $n$ points in $d$-dimensional Euclidean space $\Re^d$. In the *nearest neighbor searching* (*NNS*) problem, given a query point $q \in \Re^d$, the goal is to find a point of $P$ that is closest to $q$, i.e. determine $argmin_{p \in P}\{\mathrm{d}(p,q)\}$. We also refer to this point as the *NN* of $q$. In the approximation version of the problem, the goal is to return a point of $P$ that is at a distance of no more than a factor of $(1+\varepsilon)$ of the distance to the closest point of $P$, i.e. determine a point $p \in P$ such that $\mathrm{d}(p,q) \leq (1+\varepsilon) \cdot \mathrm{d}(p',q)$ for all $p' \in P$. The point $p$ is said to be an $\varepsilon$-nearest neighbor ($\varepsilon$-*NN*) of $q$. We also refer to the approximate version of the problem as the $\varepsilon$-*NNS* problem.

*Nearest neighbor searching* queries are often answered by building a *Voronoi diagram* of $P$. A *Voronoi diagram* is the partitioning of $\Re^d$ into regions, $\mathcal{R}$, where each region is associated with a point in $P$, such that if $R$ is a region and $p \in P$ is the associated point, then for any point $q$ in the region $R$, $p$ is the closest point of $P$, i.e., $\mathrm{d}(p,q) \leq \mathrm{d}(p',q)$ for all $p' \in P$. In an *approximate Voronoi diagram*, the region $R$ associated with a point $p$ is such that for any point $q$ in the region $R$, $\mathrm{d}(p,q) \leq (1+\varepsilon) \cdot \mathrm{d}(p',q)$ for all $p' \in P$.

### 1.1.1 Previous Results

Some important results for approximate nearest neighbor searching (query time and space bounds) are listed in Table 1.1.

For the approximate *nearest neighbor searching* problem, Arya and Mount [4] proposed a randomized data structure which achieves logarithmic query time in the expected case, and nearly linear space with both bounds containing factors that grow as $(1/\varepsilon)^{O(d)}$. This was later improved by Arya et al. [5] who provided a data structure that achieves $O((1/\varepsilon)^d \log n)$ query time and use $O(nd)$ space. Duncan [28] observed that their query time can be improved to $O(\frac{1}{\varepsilon^d} \log(1/\varepsilon) + \log n)$ by observing that this heap has only very few del-min, and many insertions. These structures are optimal in space, but the dependence on $\varepsilon$ factor in the query time is very large. Har-Peled and Mendel [35] have also presented a data structure with similar bounds for the problem.

There have also been a number of results showing that with significantly more storage, it is possible to improve the dimensional dependence in query time. Clarkson [20] showed that the query time could be reduced to $O((1/\varepsilon)^{d/2} \log n)$ with $O((1/\varepsilon)^{d/2}(\log \rho)n)$ space, where $\rho$ is the ratio between the furthest-pair and closest-pair interpoint distances. This was improved by Chan [14] who removed the factor of $\rho$ from the space complexity. Indyk and Motwani solve the problem using point location in equal balls. Kleinberg also gave an algorithm with logarithmic query time

| Reference | Query Time | Space |
|---|---|---|
| Arya and Mount [4] | $O(\frac{1}{\varepsilon^d}\log n)$ | $O(\frac{1}{\varepsilon^d}n)$ |
| Arya et al. [5] | $O(\frac{1}{\varepsilon^d}\log n)$ | $O(dn)$ |
| Clarkson [20] | $O((1/\varepsilon)^{d/2}\log n)$ | $O((1/\varepsilon)^{d/2}(\log\rho)n)$ |
| Chan [14] | $O((1/\varepsilon)^{d/2}\log n)$ | $O((1/\varepsilon)^{d/2}n)$ |
| Indyk and Motwani [40] | $O(d\,\text{polylog}(dn))$ | $O(\frac{1}{\varepsilon^d}n\,\text{polylog}(dn))$ |
| Kleinberg [42] | $O(d^2\log^2 d + d\log^2 d\log n)$ | $O((n\log d)^{2d})$ |
| Har-Peled [33] | $O(d\log(n/\varepsilon))$ | $O(\frac{n}{\varepsilon^d}\log n\log\frac{n}{\varepsilon})$ |
| Arya et al. [3] | $O(d\log(n/\varepsilon))$ | $O(\frac{n}{\varepsilon^{(d-1)}})$ |
| *Our Result* | $O(d\log(n/\varepsilon))$ | $O(\frac{n}{\varepsilon^d}\log\frac{n}{\varepsilon})$ |

Table 1.1: Some important results for approximate nearest neighbor searching

using substantially more space. They obtain polylogarithmic query time using near linear space.

For the problem of constructing *approximate Voronoi diagrams*, Har-Peled [33] showed that it is possible to construct such a space decomposition consisting of $O(\frac{n}{\varepsilon^d}\log n\log\frac{n}{\varepsilon})$ cells. A cell is either an axis-parallel hyper-rectangle or the difference of two such cells. After pre-processing this data structure can be used to answer *approximate nearest neighbor searching* queries in time $O(\log n/\varepsilon)$ where the constant factor is only quadratic in dimension. Arya et al. [3], in work independent of ours, provided an alternate construction with linear space bounds.

Recently, there have been some related results by Har-Peled and Mendel [35] and Indyk and Andoni [39] (c.f. Section 2.4).

### 1.1.2 Our Contributions

We present an algorithm that constructs an *approximate Voronoi diagram* which can then be used to answer *approximate nearest neighbor searching* queries efficiently. We improve the space bound and pre-processing time of Har-Peled [33] by a factor of $\log n$.

**Our Result:** *Given a set $P$ on $n$ points in $\Re^d$ and a parameter $\varepsilon > 0$, one can compute a set $\mathcal{R}$ of $O(\frac{n}{\varepsilon^d}\log n/\varepsilon)$ regions, where each region is a cube or an annulus of cubes. The regions of $\mathcal{R}$*

*are disjoint and cover the space. Moreover, each region, $R$, has an associated point $p \in P$ such that for any query point $q$ contained in $R$, the point $p$ is an $\varepsilon$-nearest neighbor of $q$ in $P$. Moreover, this point, $p$ can be determined in $O(d^2 \log n/\varepsilon)$ steps. The time required for constructing the data structure is $O(\frac{n}{\varepsilon^d}\log^2 n/\varepsilon)$.*

One of the main contributions of this result is the elimination of the recursive process in forming the data structure. This eventually led to an algorithm for construction of approximate Voronoi diagram of size linear in the number of input points [55].

## 1.2 A Framework for Clustering Problems

We give a general definition of clustering problems. We shall define a clustering problem by two parameters – an integer $k$ and a real-valued cost function $f(Q, x)$, where $Q$ is a set of points, and $x$ is a point in an Euclidean space. We shall denote this clustering problem as $\mathcal{C}(f, k)$. The input to $\mathcal{C}(f, k)$ is a set of points in a Euclidean space.

Given an instance $P$ of $n$ points, $\mathcal{C}(f, k)$ seeks to partition them into $k$ sets, which we shall denote as *clusters*. Let these clusters be $C_1, \ldots, C_k$. A solution also finds $k$ points, which we call *centers*, $c_1, \ldots, c_k$. We shall say that $c_i$ is the center of cluster $C_i$ (or the points in $C_i$ are assigned to $c_i$). The objective of the problem is to minimize the quantity $\sum_{i=1}^{k} f(C_i, c_i)$.

This is a fairly general definition. Some important special cases are:

- $k$-median : $f_1(Q, x) = \sum_{q \in Q} \mathrm{d}(q, x)$.
- $k$-means : $f_2(Q, x) = \sum_{q \in Q} \mathrm{d}(q, x)^2$.

We also address the discrete versions of these problems, i.e., cases where the centers have to be one of the points in $P$. In such problems, we can make $f(Q, x)$ unbounded if $x \notin Q$.

We give here some more definitions. Although we should parameterize all our definitions by $f$, we avoid this because the clustering problem will be clear from the context.

**Definition** Given a point set $P$, let $\mathsf{OPT}_k(P)$ be the cost of the optimal solution to the clustering problem $\mathcal{C}(f, k)$ on input $P$.

**Definition** Given a set of points $P$ and a set of $k$ points $C$, let $\mathsf{OPT}_k(P, C)$ be the cost of the optimal solution to $\mathcal{C}(f, k)$ on $P$ when the set of centers is $C$.

### 1.2.1 Previous Results

Some important results for geometric clustering are summarized in Table 1.2.

A lot of research has been devoted to solving these problems exactly (see [37] and the references therein). Even the best known algorithms for the $k$-median and the $k$-means problem take at least $\Omega(n^d)$ time for $k \geq 2$. A lot of work has been devoted to finding $(1+\varepsilon)$-approximation algorithms for these problems, where $\varepsilon$ can be an arbitrarily small constant. This has led to algorithms with much improved running times for these problems.

The fastest exact algorithm for the $k$-means clustering problem was proposed by Inaba et al. [37]. They observed that the number of Voronoi partitions of $k$ points in $\Re^d$ is $O(n^{kd})$ and so the optimal $k$-means clustering could be determined exactly in time $O(n^{kd+1})$. Matousek [50] proposed a deterministic $(1+\varepsilon)$-approximation algorithm for the $k$-means problem with a running time of $O(n\varepsilon^{-2k^2d}\log^k n)$.

Based on the work of Arora [1], which proposed a new technique for geometric approximation algorithms, Arora et al. [2] presented a $O(n^{O(1/\varepsilon)+1})$ time $(1+\varepsilon)$-approximation algorithm for points in the plane. This was significantly improved by Kolliopoulos et al. [43] who proposed an algorithm with a running time of $O(\varrho n \log n \log k)$ for the discrete version of the problem, where the medians must belong to the input set and $\varrho = exp[O((1+\log 1/\varepsilon)/\varepsilon)^{d-1}]$.

Badoiu et al. [9] proposed a $(1+\varepsilon)$-approximation algorithm for $k$-median clustering with a running time of $O(2^{(k/\varepsilon)^{O(1)}} d^{O(1)} n \log^{O(k)} n)$. Their algorithm can also be extended to $k$-means with some modifications. de la Vega et al. [23] proposed a $(1+\varepsilon)$-approximation algorithm for the $k$-means problem which works well for points in high dimensions. The running time of this algorithm is $O(g(k,\varepsilon) n \log^k n)$ where $g(k,\varepsilon) = exp[(k^3/\varepsilon^8)(\ln(k/\varepsilon)\ln k]$.

Subsequently, Har-Peled et al. [34] proposed $(1+\varepsilon)$-approximation algorithms for the $k$-medians and the $k$-means clustering problems in low dimensions (discrete and non-discrete versions). They had a running time of $O(n + \varrho k^{O(1)} \log^{O(1)} n)$, where $\varrho = exp[O((1+\log 1/\varepsilon)/\varepsilon)^{d-1}]$ for the $k$-median problem and its discrete version and $O(n + k^{k+2}\varepsilon^{-(2d+1)k}\log^{k+1} n \log^k \frac{1}{\varepsilon})$ for the $k$-means problem and its discrete version.

For approximating the 1-median of a set of points, Indyk [38] proposed an algorithm that finds a $(1+\varepsilon)$ approximate 1-median in time $O(n/\varepsilon^2)$ with constant probability.

Recently, there have been some related results by Ke Chen [19] (c.f. Section 5.7).

| Problem | Result | Reference |
|---|---|---|
| 1-median | $O(n/\varepsilon^2)$ | Indyk [38] |
| | $O(2^{(1/\varepsilon)^{O(1)}}d)$ | *Our Result* |
| $k$-medians | $O(n^{O(1/\varepsilon)+1})$ for $d=2$ | Arora [1] |
| | $O(\varrho n \log n \log k)$ (discrete only) | Kolliopoulos & Rao [43] |
| | $O(2^{(k/\varepsilon)^{O(1)}}d^{O(1)}n\log^k n)$ | Badoiu et al. [9] |
| | $O(n+\varrho k^{O(1)}\log^{O(1)}n)$ (discrete also) where $\varrho = exp[O((1+\log 1/\varepsilon)/\varepsilon)^{d-1}]$ | Har-Peled & Mazumdar [34] |
| | $O(2^{(k/\varepsilon)^{O(1)}}nd)$ | *Our Result* |
| $k$-means | $O(n\varepsilon^{-2k^2d}\log^k n)$ | Matousek [50] |
| | $O(g(k,\varepsilon)n\log^k n)$ | de la Vega et al. [23] |
| | where $g(k,\varepsilon) = exp[(k^3/\varepsilon^8)(\ln(k/\varepsilon)\ln k]$ | |
| | $O(n+k^{k+2}\varepsilon^{-(2d+1)k}\log^{k+1}n\log^k\frac{1}{\varepsilon})$ (discrete also) | Har-Peled & Mazumdar [34] |
| | $O(2^{(k/\varepsilon)^{O(1)}}nd)$ | *Our Result* |

Table 1.2: Some important results for geometric clustering

### 1.2.2 Our Contributions

We present a general framework that solves a wide range of clustering problems in linear time. Using this framework we obtain $(1+\varepsilon)$-approximation algorithms for the *$k$-means clustering*, *$k$-medians clustering* and *discrete $k$-means clustering* problems.

All these algorithms are randomized algorithms that succeed with constant probability (which can be made as close to 1 as we wish by a constant number of repetitions). The running time of all these algorithms is $O(2^{(k/\varepsilon)^{O(1)}}nd)$. These are the first algorithms for these problems that are linear in the size of the input ($nd$, for $n$ points in $d$ dimensions), for fixed values of $k$.

**Our Results:** *Let $P$ be a set of $n$ points in $\Re^d$. Given an approximation factor $\varepsilon > 0$, there exist*

*algorithms which in time $O(2^{(k/\varepsilon)^{O(1)}}nd)$ produce a $(1+\varepsilon)$-approximation to the*

- $k$-means clustering of $P$:
  *i.e., finds a partitioning $C_1, C_2, ..., C_k$ of $P$ and centers $c_1, c_2, ..., c_k$, such that*

$$\sum_{i=1}^{k} f_2(C_i, c_i) \leq (1+\varepsilon) \sum_{i=1}^{k} f_2(O_i, o_i)$$

  *where $O_1, O_2, ..., O_k$ denote the clusters in the optimal $k$-means clustering of $P$ and $o_1, o_2$, ..., $o_k$ denote the corresponding centers.*

- $k$-median clustering of $P$:
  *i.e., finds a partitioning $C_1, C_2, ..., C_k$ of $P$ and centers $c_1, c_2, ..., c_k$, such that*

$$\sum_{i=1}^{k} f_1(C_i, c_i) \leq (1+\varepsilon) \sum_{i=1}^{k} f_1(O_i, o_i)$$

  *where $O_1, O_2, ..., O_k$ denote the clusters in the optimal $k$-median clustering of $P$ and $o_1, o_2$, ..., $o_k$ denote the corresponding centers.*

- discrete $k$-means clustering of $P$:
  *i.e., finds a partitioning $C_1, C_2, ..., C_k$ of $P$ and centers $c_1, c_2, ..., c_k \in P$, such that*

$$\sum_{i=1}^{k} f_2(C_i, c_i) \leq (1+\varepsilon) \sum_{i=1}^{k} f_2(O_i, o_i)$$

  *where $O_1, O_2, ..., O_k$ denote the clusters in the optimal discrete $k$-means clustering of $P$ and $o_1, o_2, ..., o_k \in P$ denote the corresponding centers.*

*All these algorithm succeed with constant probability.*

A key ingredient of the $k$-median result is a randomized algorithm that for a given set $P$ in $\Re^d$, generates a candidate center set of size $O(2^{1/\varepsilon^{O(1)}})$, such that at least one of the points in this set is a $(1+\varepsilon)$-approximate 1-median of $P$ with constant probability in time $O(2^{1/\varepsilon^{O(1)}}d)$. Using this candidate center set we present an improved approximation algorithm for the 1-median problem. Here, it is assumed that the input is provided as an array where it is possible to randomly sample

a point in constant time. The algorithm does not have to read the entire input but only inspects a subset of the input elements thereby making sublinear running time possible [18].

**Our Result (1-median clustering) :** *Let $P$ be a set of $n$ points in $\Re^d$. Let $o$ be an optimal 1-median of $P$. Given an approximation factor $\varepsilon > 0$, there exists an algorithm which in time $O(2^{(1/\varepsilon)^{O(1)}} d)$ produces a $(1+\varepsilon)$-approximation to the optimal 1-median clustering of $P$, i.e., it finds a point $c \in \Re^d$ such that*

$$f_1(P, c) \leq (1+\varepsilon) f_1(P, o).$$

*The algorithm succeeds with constant probability.*

## 1.3 $K$-Median Problem with priorities

In the *priority $k$-median* problem, we are given a set of demands, $\mathcal{D}$ and a set of facilities $\mathcal{F}$ in a metric space. Each demand has a weight $d_j$ associated with it, denoting the quantity of demand to be assigned to an open facility.

There are $m$ *types* of demands, with the type indicating the priority of the demand. Thus $\mathcal{D}$ is the disjoint union of $\mathcal{D}_1, \ldots, \mathcal{D}_m$, where we say that $\mathcal{D}_k$ are demands of *type* $k$. Similarly, there are $m$ *types* of facilities, i.e., $\mathcal{F}$ is a disjoint union of $\mathcal{F}_1, \ldots, \mathcal{F}_m$, where we say that $\mathcal{F}_k$ are facilities of *type* $k$. The *type* of a facility specifies its capability in serving the demands – a facility of *type* $k$ can serve demands of *type* $\geq k$, i.e., a demand of *type* $k$ can be served by a facility of *type* $\leq k$.

Let $c_{ij}$ denote distance between $i$ and $j$ where $i, j$ can be demands or facilities. A feasible solution opens a set of facilities $F$, and assigns each demand to an open facility. We are given bounds $k_r$ on the number of facilities that can be opened from $\mathcal{F}_r$. As mentioned above, a demand $j$ can only be assigned to an open facility of type $type(j)$ or lower. Let $i(j)$ denote the facility that a demand $j$ is assigned to. Then the cost of the solution is defined as $\sum_{j \in \mathcal{D}} d_j \cdot c_{ji(j)}$. The goal of the *priority $k$-median* problem is to obtain a solution of minimum cost.

### 1.3.1 Previous Results

The *Priority $k$-median* problem has not been studied before.

The $k$-median problem in metric spaces has been extensively studied in the past and several constant factor approximation algorithms are known for this problem. Lin and Vitter [49] gave

a bicriteria constant factor approximation algorithm for this problem, even when distances do not obey triangle inequality. Assuming that distances obey triangle inequality, the first constant factor approximation algorithm was given by Charikar et. al. [16]. Jain et. al. [41] gave a primal-dual constant factor approximation algorithm for this problem. Several constant factor approximation algorithms based on local search techniques are known [6, 15, 44].

A slightly related problem is the so called priority Steiner tree problem. In this problem we seek to build a Steiner tree over some demand vertices. However, different demands can be of different priorities and edges can be of different kinds. A high priority demand can be connected using types of higher kinds only. Charikar et. al. [17] gave approximation algorithms for this problem.

### 1.3.2 Our Contributions

Our main result is stated below.

**Our Result:** *There exists an algorithm that solves the priority $k$-median problem with two priorities within a constant factor of approximation in polynomial time.*

The solution is based on formulation of a natural integer program formulation for this problem and rounding its relaxed linear program to obtain a simpler problem for which another integer program is formulated. We show that the optimal solution to the new linear program is half integral, which can then be rounded off to obtain an integral solution.

We also show that the linear program relaxation to the integer program described above admits an arbitrarily large integrality gap for four or more priorities.

## 1.4 Thesis Organization

The remaining thesis is organized as follows. In Chapter 2, we present the improved algorithm for constructing approximate Voronoi diagrams. The next three chapters describe the general framework for clustering problems. In Chapter 3, we present a linear time algorithm for the 2-means clustering problem. We also present here some key ideas that are extended in the general algorithm. In Chapter 4, we present a simplified version of the general algorithm for the case of 2-means clustering for exposition. This makes it easier to understand the more general algorithm. The general algorithm is presented in Chapter 5 along with the proofs of the properties for various clustering

problems and their weighted version. Finally, in Chapter 6, we discuss the prioritized $k$-median problem. We present a constant factor approximation algorithm for the case of 2 priorities.

# Chapter 2

# Approximate Voronoi Diagram

In the nearest neighbor searching (*NNS*) problem, given a set $P$, of $n$ points in $d$-dimensional Euclidean space $\Re^d$, we are interested in finding for a query point $q$, a point of $P$ that is closest to $q$. In the approximation version of the problem, the goal is to return a point $p$ of $P$ such that $\mathrm{d}(p,q) \leq (1+\varepsilon)\cdot \mathrm{d}(p',q)$ for all $p' \in P$. The point $p$ is said to be an $\varepsilon$-nearest neighbor ($\varepsilon$-*NN*) of $q$. We also refer to the approximate version of the problem as the $\varepsilon$-*NNS* problem.

*Nearest neighbor searching* queries are often answered by building a *Voronoi diagram* of $P$. A *Voronoi diagram* is the partitioning of $\Re^d$ into regions, $\mathcal{R}$, where each region is associated with a point in $P$, such that if $R$ is a region and $p \in P$ is the associated point, then for any point $q$ in the region $R$, $p$ is the closest point of $P$, i.e., $\mathrm{d}(p,q) \leq \mathrm{d}(p',q)$ for all $p' \in P$. In an *approximate Voronoi diagram*, the point $p$ associated with a region $R$ is such that for any point $q$ in the region $R$, $\mathrm{d}(p,q) \leq (1+\varepsilon)\cdot \mathrm{d}(p',q)$ for all $p' \in P$.

In this chapter we present an improved construction of approximate Voronoi diagrams for approximate nearest neighbor searching. The data structure we present is of near-linear size ($O(\frac{n}{\varepsilon^d}\log n/\varepsilon)$) and can answer nearest neighbor queries in time $O(d^2 \log n/\varepsilon)$. Moreover, the time required for constructing the data structure is $O(\frac{n}{\varepsilon^d}\log^2 n/\varepsilon)$.

This improves the space bound and pre-processing time over previously known results by a factor of $\log n$. Parts of these results have appeared in [54] and [55].

## 2.1 Related Work

In the context of our results, the the work of Indyk and Motwani [40] holds special significance where they reduced the problem of $\varepsilon$-*NNS* to Approximate Point Location in Equal Balls ($\varepsilon$-*PLEB*).

**Definition 2.1.1** *Given $P$ and a parameter $r$, an* $\varepsilon$-PLEB$(P, r)$ *is a data structure which when given a query point $q$ returns a point $p \in P$ such that d$(p, q) \leq r(1 + \varepsilon)$, if $q \in$ b$(p', r)$ for some $p' \in P$. It returns* nil *if $q \notin$ b$(p', r(1 + \varepsilon))$ for all $p' \in P$. Otherwise if $q \in$ b$(p', r(1 + \varepsilon))$ for some $p' \in P$, it returns either a point $p \in P$ such that d$(p, q) \leq r(1 + \varepsilon)$ or it returns* nil. *When $\varepsilon = 0$, then it is called* PLEB$(P, r)$.

Indyk and Motwani [40] used a novel ring-cover tree to reduce $\varepsilon$-*NNS* to $\varepsilon$-*PLEB*. The $\varepsilon$-*PLEB* is solved by using a simple hash-based technique. However, their reduction was quite complex which was improved in all respects in the work of Har-Peled [33] who presented an alternate decomposition of space called an *Approximate Voronoi Diagram* (to be referred as $AVD$) that supports approximate nearest neighbor searching and has a significantly lower space complexity. His result can be summarized as follows.

**Theorem 2.1.1 (Har-Peled)** *Given a set $P$ of $n$ points in $\mathbb{R}^d$ and a parameter $\varepsilon > 0$ one can compute a set $\mathcal{C}(P)$ of $O(n\frac{\log n}{\varepsilon^d}\log n/\varepsilon)$ regions where each region is a cube or an annulus of cubes. The regions of $\mathcal{C}(P)$ are disjoint and cover the space and each region has an associated point $p \in P$, such that for any point $q \in c$, the point $p$ is an* $\varepsilon$-NN *of $q$ in $P$ and it can be computed in $O(\log(n/\varepsilon))$ steps.*

*The time for constructing the data-structure is $O(n\frac{\log n}{\varepsilon^d}\log^2 n/\varepsilon)$.*

This data structure is computed in two phases. In the first phase the *approximate nearest neighbor searching* ($\varepsilon$-*NN*) problem is reduced to locating the smallest ball containing a query point. We call this the *point location in smallest ball (PLSB)* problem.

**Definition 2.1.2** *Let $\mathcal{B} = \{b_1, b_2, \ldots, b_m\}$ be a set of $m$ balls, where each ball $b_i$ is a pair b$(p_i, r_i)$ representing a ball centered at $p_i$ of radius $r_i$. An* $\varepsilon$-PLSB$(\mathcal{B})$ *is a data structure which when given a query point $q$ returns a ball $b_i =$ b$(p_i, r_i) \in \mathcal{B}$ such that d$(p_i, q) \leq (1 + \varepsilon)r_i$ and for every $b_j =$ b$(p_j, r_j)$ containing $q$, $r_i \leq r_j$, if $q \in$ b$(p'_i, r'_i)$ for some $p'_i \in P$. It returns* nil *if $q \notin$ b$(p'_i, r'_i(1 + \varepsilon))$ for all $p'_i \in P$. Otherwise if $q \in$ b$(p'_i, r'_i(1 + \varepsilon))$ for some $p'_i \in P$, it returns either a point $p_i \in P$ such that d$(p_i, q) \leq r_i(1 + \varepsilon)$ or it returns* nil. *When $\varepsilon = 0$, then it is called* PLSB$(\mathcal{B})$.

*Note that* PLEB *is a special case of* PLSB*, with all balls of equal radius.*

In the second phase, Har-Peled [33] used a more sophisticated data structure for solving the *PLSB* problem approximately based on the BBD data-structure of Arya et al[33].

We summarize below the approach of Har-Peled [33] .

**Phase 1:** One of the main observation Har-Peled [33] exploited is that if the query point $q$ is at a distance of more than d$(x, y)/\gamma$ from two points $x$ and $y$, then both $x$ and $y$ are $\gamma$-nearest neighbors of $q$ and therefore either of them can be reported as a $\gamma$-nearest neighbor of $q$. Therefore only one of these points needs to be retained after a certain distance.

Har-Peled [33] generalized this observation to groups (cluster of points) as well, using which he built a hierarchical tree data structure on the input point set. This is explained in more detail later (c.f. Section 2.2.1).

He then used this tree to carefully construct a set of balls around the input points such that solving the *PLSB* problem on the constructed set of balls results in determination of an approximate nearest neighbor. In total $O(n \log^2 n)$ balls are constructed which define the *PLSB* problem.

**Phase 2:** Har-Peled [33] showed that a *PLSB* problem can be solved approximately using a data structure linear in the number of balls. His result can be summarized as follows:

**Lemma 2.1.2** *An* $\varepsilon$-PLSB *problem on* $m$ *balls in d-dimensional Euclidean space can be solved by constructing a data structure of space complexity* $O(m/\varepsilon^d)$ *that can answer queries in* $O(\log m)$ *time. Further the time required to create the data structure is* $O(m \log m/\varepsilon^d)$.

Each ball of radius $r$ is replaced by the set of cubes from the hierarchical grid, of edge length $\frac{r\varepsilon}{3d}$, which intersect that ball. By choosing the lengths of cubes as $2^i$ truncated to the nearest power of two, these cubes are of geometrically decreasing sizes and nested within each other. Then a compressed quadtree is constructed over the cubes generated. The final search structure defines a partition of the space where each cell is a cube or a difference of cubes and each cell is associated with a point of $P$ that is an $\varepsilon$-*NN* of the region inside the cell. To answer a query, among all cubes containing the query point, the cube corresponding to the smallest ball is found. Then the ball of the input data set, to which it corresponds is returned.

## 2.2 Reduction to Point Location in Smallest Ball

In our algorithm, we eliminate the recursive structure of Har-Peled's algorithm by constructing a set of balls directly from the clustering. This also reduces the number of balls by a factor of $\log n$. The search time remains logarithmic and the preprocessing time is also improved by a factor of $O(\log n)$. We will only bound the number of balls in the reduction to *PLSB*. This is linearly related to the space complexity and can be solved using Lemma 2.1.2.

### 2.2.1 Building a Hierarchical Clustering

Har-Peled, in his paper [33], proposes a solution based on the following observation:

**Observation 2.1** *: If the distance between two points $A$ and $B$ is $|AB|$, and a query point $Q$ lies outside $d$-balls of radius $k \cdot |AB|$, where $k$ is set to $1/\epsilon$, then $A$ and $B$ are $\epsilon$- neighbors of $Q$, i.e., $\frac{|QA|}{|QB|} < 1 + \epsilon$*

Har-Peled [33] further proposes a method for clustering the points into a hierarchical tree. Consider the connected components obtained by growing balls around all the points. We also construct a forest (called cluster tree) from the connected component.

Initially we start out with just the set of points themselves. Thus we start with $n$ connected components consisting of the individual points. The corresponding forest consists of $n$ trees with each tree consisting of a single leaf, corresponding to a point.

We then grow the balls around all the points uniformly. When two components meet, we get a new component that replaces the two components corresponding to the points whose balls meet and is a result of merging the two components. In the corresponding forest, we merge the two trees corresponding to the components being merged by hanging one of the trees on the other one arbitrarily (see Figure 2.1).

At this point, we also associate a value of $r_{loss}(p)$ with the root, $p$ of the tree that we hang on the other one. This value corresponds to the radius of the balls when these components meet. It is actually half the distance between the closest points of the two components.

Thus $r_{loss}(p) =$ radius of the ball at $p$, when $p$ ceases to be root.

#### Properties of the cluster tree

- Values of $r_{loss}$ increase along a path towards the root.

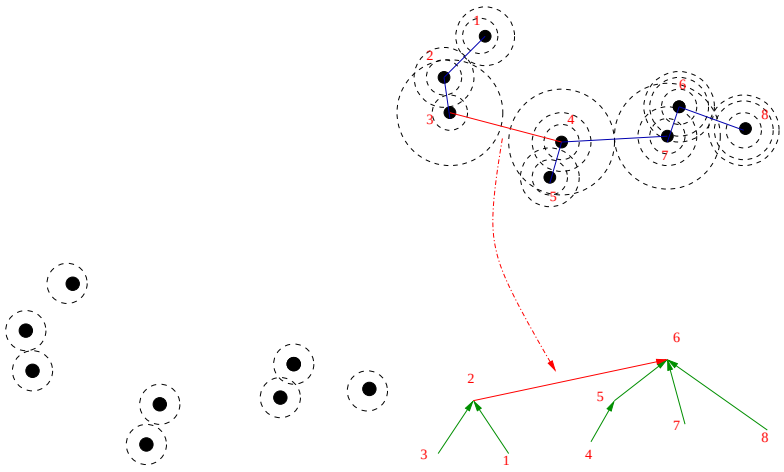


Figure 2.1: Construction of the cluster tree

- If $L_{max}$ is the longest edge of the MST, the tree edge corresponding to this is the last edge to be added.

- If any query point $q$ that is outside of the union of balls of radii $L_{max}.|P|.1/\epsilon$ centered at $p \in P$, i.e., q is too far from $P$, then any point $p \in P$ is an $\epsilon$-nearest neighbor of $q$. Note that the diameter of $P$ is bounded by $L_{max}.|P|$.

The above properties also hold for any subtree (cluster) formed during the construction.

Har-Peled [33], instead of working with an exact MST in $d$-dimensions which takes $O(n^2)$ time to compute for large $d$, settles for an $nd$ factor approximation that can be computed in $O(n\log^2 n)$ time.

Using the $nd$-MST, he constructs an $nd$-stretch Hierarchical Clustering and then gives an algorithm for the construction of a family of *PLEB*s (Point Location in Equal Balls) for answering the approximate nearest neighbor queries in time $O(\log(\frac{n}{\epsilon}))$ using space $O(\frac{n}{\epsilon^d}\log n\log(\frac{n}{\epsilon}))$. The bound on the space depends on the number of balls generated by the family of *PLEB*s.

We present a different perspective, whereby we decrease the number of balls required for answering approximate nearest neighbor queries by pruning overlapping and unwanted balls from the set of balls centered around a point $p \in P$. We give an independent algorithm for the construction of balls around the points of $P$, such that reporting the center of the smallest ball that contains a query point answers the approximate nearest neighbor query.

The following definitions are similar to Har-Peled [33] that have been restated for the convenience of the reader.

**Definition 2.2.1** $\lambda$-MST : *A tree $T$ having the points of P in its nodes, is a $\lambda - MST$ of $P$, if it is a spanning tree of $P$, having a value $len(.)$ associated with each edge of $T$, such that for any edge $e = uv$ of $T$, $len(e) \leq d(P_1, P_2) \leq \lambda len(e)$, where $P_1$,$P_2$ is the partition of $P$ into two sets as induced by the two connected components of $T$-$\{e\}$, and $d(P_1, P_2) = min_{x \in P_1, y \in P_2} ||xy||$ is the distance between $P_1$ and $P_2$.*

One can compute a $nd$-MST of $P$ in $O(nd \log^2 n)$ time. Har-Peled [33] shows how such an $nd$-MST can be constructed, and is similar to the fair-split tree construction of Callahan and Kosaraju [13].

**Definition 2.2.2** $\lambda$-stretch Hierarchical Clustering : *A Directed tree $T$ storing the points of $P$ in its nodes, so that for any point $p$, there is a value $r_{loss}(p)$ associated with the out-edge emanating from p to its parent, satisfying the following property : If $C_1$ denotes connected components obtained on constructing balls of radius $r$ around all points, $C_2$ denotes connected components obtained on constructing balls of radius $\lambda r$ around all points, $F$ is the forest obtained by removing from $T$, all edges larger than $r$ and $X$ is the partition of $P$ into subsets induced by the connected components of the forest $F$, then $C_1 \leq X \leq C_2$, where $X \leq Y \Rightarrow$ if $A$ and $B$ are in the same component in $X$, then they are in the same component in $Y$.*

The $nd$-stretch Hierarchical Clustering is built from the $nd$-MST as follows. We sort the edges of the $nd$-MST in order of the $len(.)$ values of the edges. We then traverse the list. In each iteration, we take the next edge, say $e = xy$, from the sorted list and connect the two subtrees to which $x$ and $y$ belong by hanging the smaller tree onto the larger one by introducing an edge between the roots of the two subtrees. We associate with this edge $e$, the value $r_{loss}(e) = len(xy)/2$.

For the root, $t$, of the tree, we assign $r_{loss}(t) = max$ ( $r_{loss}(p_i)$, where $p_i$ is a child of $t$ ).

It is easy to see that the $nd$-Hierarchical clustering can be constructed in time $O(nd \log^2 n)$ based on the construction of the $nd$-MST.

**Observation 2.2** *: The height of the tree formed in the $\lambda$-stretch hierarchical clustering of $P$ is at most $\log n$ (Hanging the smaller tree below the larger one ensures this property).*

We will use b$(p, r)$ to denote the ball of radius $r$ centered at $p$.

**Definition 2.2.3** $r_{low}$, $r_{high}$ : *For an approximation factor $\gamma$, let $r_{low}(p) = (1/(1+\gamma/3))r_{loss}(p)$ and $r_{high}(p) = (36\lambda r_{loss}(p)n\log n/\gamma)$. Note that $r_{low}(p)$ gives us a value just smaller than $r_{loss}(p)$.*

**Definition 2.2.4** $parent(p)$, $parent^i(p)$ *: Define $parent(p)$ to be the parent of node $p$ in the tree obtained by the $\lambda$-stretch hierarchical clustering of $P$. Define $parent^i(p)$ to be the $i^{th}$ parent of node $p$ in the tree obtained by the $\lambda$-stretch hierarchical clustering of $P$, i.e., $parent^0(p) = p$ and $parent^i(p) = parent(parent^{(i-1)}(p))$. Note that $parent^i(p)$ is defined till it is the root of the tree obtained by the $\lambda$-stretch hierarchical clustering of $P$, and is not defined beyond the root.*

We will use $Subtree(p)$ to denote the subtree of the $\lambda$-stretch hierarchical clustering of $P$ rooted at $p$.

### 2.2.2 Construction of Balls (Algorithm ConstructBalls$(P, \gamma)$)

In this section $\lambda = nd$. Given a $\lambda$-stretch hierarchical clustering of $P$, for each point $p$ in $P$, let $r_0$ be the $r_{loss}$ value for $p$ and let $p_1$, $p_2$, ...,$p_m$ be the children of $p$ in sorted order of their $r_{loss}$ values, i.e., $r_{loss}(p_1) \leq r_{loss}(p_2) \leq ... \leq r_{loss}(p_m)$. Also, let $x$ be the parent of $p$ in the tree formed by the $\lambda$-stretch hierarchical clustering of $P$.

We construct the following ball sets around $p$:

1. Balls with radius $r_i = r_{loss}(p)(1+\gamma/3)^{(j-1)}$ for $j = 0, .., M+1$,
   where $M = \lceil \log_{(1+\gamma/3)}(r_{high}(p)/r_{loss}(p)) \rceil$.

   This defines a ball set in the range $[r_{low}(p), r_{high}(p)]$ around $p$.

2. Corresponding to each child $p_i$ of $p$,
   balls with radius $r_i = r_{loss}(p_i)(1+\gamma/3)^{(j-1)}$ for $j = 0, .., M+1$,
   where $M = \lceil \log_{(1+\gamma/3)}(r_{high}(p_i)/r_{loss}(p_i)) \rceil$.

   This defines a ball set in the range $[r_{low}(p_i), r_{high}(p_i)]$ $\forall$ $1 \leq i \leq m$ around $p$.

We also construct a universal ball (of infinite radius) centered at the point that is the root of the tree formed by the $\lambda$ -stretch hierarchical clustering of $P$, so that any point that is not in any of the balls constructed by the above rules lies in this ball.

### 2.2.3 Correctly answering $(1+\epsilon)$-approximate *NN* queries

In this section we prove that reporting the center of the smallest ball that contains a query point from the set of balls constructed by the algorithm **ConstructBalls** answers the approximate nearest neighbor query correctly.

**Observation 2.3** *$|| ab || \geq 2r_{loss}(p)\ \forall\ a \in Subtree(p)$ & $b \notin Subtree(p)$ as $2r_{loss}(p)$ is the minimum separation between any point in the cluster from any point outside it by definition of $r_{loss}$.*

In order to limit the number of balls that we construct around a given point, we wish to determine the largest ball such that for any query point beyond that we can claim the existence of another point which is a near neighbor of the query point with a limited accumulated approximation error.

The following lemma establishes that $r_{high}(x)$ is a limit on the radius of the largest ball to be constructed around the point $x$, so that if the query point does not lie in this ball, then $parent(x)$ is a near neighbor of the query point with some accumulated approximation error. Thus, we can ignore the point $x$ at the expense of some accumulated approximation error.

By the definition of $r_{high}(x)$, the accumulated error is a factor of $1+\gamma/(3\log n)$. This value is so chosen that repeated accumulation of approximation errors is bounded by a suitable value as we will see later.

**Lemma 2.2.1** *For any query point $q$, if $x$ is a $(1+\alpha)$-*NN *of $q$ in $P$, and if $||qx|| > r_{high}(x)$, then $parent(x)$ is a $((1+\gamma/(3\log n))(1+\alpha))$-*NN *of $q$ in $P$.*

*Proof.* The proof of this lemma is similar to the proof of Lemma 2.13 in [33].

Let $z$ be parent of $x$ in the $\lambda$-stretch hierarchical clustering of the set of points $P$. Note that $||qx|| > r_{high}(x)$ . By the same argument as in Lemma 2.10 of [33], $||zx|| \leq 2n\lambda r_{loss}(x) = (\gamma/(18\log n))r_{high}(x) < (\gamma/(18\log n))||qx|| < (\gamma/(3\log n))||qx||$. Therefore, $||zq|| \leq ||qx|| + ||zx|| \leq (1+\gamma/(3\log n))||qx|| \leq (1+\gamma/(3\log n))(1+\alpha)\mathrm{d}_P(q)$.

It follows that $parent(x)$ is a $((1+\gamma/(3\log n))(1+\alpha))$-*NN* of $q$ in $P$. ∎

In the following lemma, we show that it is possible to recursively consider the parent of the current candidate as the next nearest neighbor candidate if the query point does not lie in the largest ball around the current candidate. Of course, at every step that we shift the candidate to the parent, we accumulate an extra error in our approximation.

**Lemma 2.2.2** *Let $p$ be the* NN *of a query point $q$ in $P$. Let $r$ be the radius of the smallest ball containing $q$ and let it be centered at $x$. If $r > r_{high}(parent^i(p)) \; \forall \; 0 \leq i \leq j-1$, then $parent^j(p)$ is a $((1+\gamma/(3\log n))^j$-approximate* NN *of $q$ in $P$.*

*Proof.* The proof is by induction on $j$ using Lemma 2.2.1. ∎

In the following lemma, we show that we may incur an additional error (multiplicative factor) of $1+\gamma/3$ in our approximation while reporting the nearest neighbor, since we are constructing a discrete set of balls for every ballset, with each ball being larger than the previous one by a factor of $1+\gamma/3$. Consider the scenario where a point $p$ is an approximate nearest-neighbor of a query point $q$ such that $q$ lies in one of the balls constructed in some ballset of $p$. Suppose that we end up reporting another point $x$ because the ball containing $q$ centered at $x$ is smaller than the ball containing $q$ centered at $p$, even though $p$ is closer. This is possible because we only construct a discrete set of balls. The following lemma shows that we only suffer an additional approximation error factor of $1+\gamma/3$, i.e., by reporting $x$ instead of $p$.

**Lemma 2.2.3** *For any query point $q$, if $p$ is a $(1+\alpha)$-*NN *of $q$ in $P$ and there exists a ballset, centered at $p$, in $[r_{low}(t), r_{high}(t)]$ for some point $t$, and if the smallest ball containing $q$ has radius $r$, such that $r_{loss}(t) \leq r \leq r_{high}(t)$ and is centered at $x$, then $x$ is a $(1+\gamma/3)(1+\alpha)$-approximate* NN *of $q$ in $P$.*

*Proof.* If $x = p$, then we are done.

Suppose $x \neq p$. Then two cases arise, either $q \notin \mathrm{b}(p, r_{high}(t))$ or $q \in \mathrm{b}(p, r_{high}(t))$. We will show in either case that $\mathrm{d}_P(q) \leq ||xq|| \leq (1+\gamma/3)(1+\alpha)\mathrm{d}_P(q)$.

- **Case I :** $q \notin \mathrm{b}(p, r_{high}(t))$

  In this case, $|| \; pq \; || > r_{high}(t) \geq r > || \; xq \; ||$. This implies that $x$ is closer to $q$ than $p$ and therefore trivially $||xq|| \; < \; (1+\gamma/3)(1+\alpha)\mathrm{d}_P(q)$.

- **Case II :** $q \in \mathrm{b}(p, r_{high}(t))$

  Clearly, $q \notin \mathrm{b}(p, r_{low}(t))$, because otherwise the smallest ball containing $q$ would have radius $\leq r_{low}(t)$ and $r \geq r_{loss}(t) > r_{low}(t)$ leading to a contradiction that the smallest ball containing $q$ has radius $r$. Hence, $\exists j$, such that $q \in \mathrm{b}(p, r_j)$ and $q \notin \mathrm{b}(p, r_{j+1})$, where $r_0 = r_{low}(t)$, implying that $r_j \; < \; ||pq|| \leq r_{j+1} \; = \; (1+\gamma/3)r_j$.

  Therefore, $||xq|| \leq r \leq r_{j+1} \; = \; (1+\gamma/3)r_j \; < \; (1+\gamma/3)||pq|| \leq (1+\gamma/3)(1+\alpha)\mathrm{d}_P(q)$.

Thus $x$ is a $(1+\gamma/3)(1+\alpha)$-approximate *NN* of $q$ in $P$. ∎

Suppose that the query point $q$ is contained in the ball of radius $r_{loss}(p)$ for a point $p$. Then, clearly the nearest-neighbor of the query point $q$ can only lie amongst the points that belong to the cluster formed by the subtree of the hierarchical clustering tree rooted at $p$. However, it still needs to be determined which of these points is closer, and $p$ itself is also a candidate. This calls for the construction of more balls around $p$ of radii corresponding to the ballsets constructed around the child nodes of $p$. This way we determine which of $p$ and its children is closer to the query point $q$. This is precisely the set of balls constructed by rule 2 in algorithm **ConstructBalls**. Note that we do not need to construct balls around $p$ corresponding to the children further down the hierarchy, as the child nodes of $p$ would then themselves be better candidates.

The following lemma shows that the construction of balls in algorithm **ConstructBalls** for the case described above leads to answering the queries with only a further compounded approximation error of a factor of $1+\gamma/3$.

**Lemma 2.2.4** *For any query point $q$, if $p$ is a $(1+\alpha)$-*NN *of $q$ in $P$, and we have balls constructed around the points of $P$ as defined by algorithm* **ConstructBalls***, and if the smallest ball containing $q$ has radius $r$, such that $r < r_{loss}(p)$ and is centered at $x$, then $x$ is a $(1+\gamma/3)(1+\alpha)$-approximate* NN *of $q$ in $P$.*

*Proof.* If $q \notin \mathrm{b}(p, r_{loss}(p))$, then $||pq|| > r_{loss}(p) > r \geq ||xq||$ implying that $x$ is closer to $q$ than $p$ and therefore trivially $||xq|| \leq (1+\gamma/3)(1+\alpha)\mathrm{d}_P(q)$.

Suppose $q \in \mathrm{b}(p, r_{loss}(p))$. Since $q \in \mathrm{b}(x, r)$ and $q \in \mathrm{b}(p, r_{loss}(p))$, these balls intersect. Then $x \in Subtree(p)$ because all balls of $radius < r_{loss}(p)$ centered around points $\notin Subtree(p)$ cannot intersect since $||\ ab\ || \geq 2r_{loss}(p)\ \forall\ a \in Subtree(p)$ & $b \notin Subtree(p)$. Let $p_1$ be the child of $p$, such that $x \in Subtree(p_1)$. We consider three cases based on the value of $r$.

- **Case I :** $r < r_{loss}(p_1)$

  We know that $x \in Subtree(p_1)$ & $p \notin Subtree(p_1)$. Then $||pq|| > r$, because otherwise $||px|| \leq ||pq|| + ||qx|| \leq r + r = 2r < 2r_{loss}(p_1)$. and this would contradict observation 2.3.

  This implies that $x$ is closer to $q$ than $p$ and therefore trivially $||xq|| \leq (1+\gamma/3)(1+\alpha)\mathrm{d}_P(q)$.

- **Case II :** $r > r_{high}(p_1)$

This case is not possible as we don't construct balls larger than $r_{high}(t)$ around any point $t$ of $P$, and $r_{high}(t) \leq r_{high}(p_1)\ \forall\ t \in Subtree(p_1)$, but we know that there is a ball centered at $x \in Subtree(p_1)$ that contains $q$.

- **Case III :** $r_{loss}(p_1) \leq r \leq r_{high}(p_1)$

  Since there is a ballset around $p$ in the interval $[r_{low}(p_1), r_{high}(p_1)]$, $\exists\ j$, such that $q \in \mathrm{b}(p, r_j)$ and $q \notin \mathrm{b}(p, r_{j+1})$, where $r_0 = r_{low}(p_1)$. This implies that $||pq|| > r_j$. It must be that $r \leq r_{j+1}$, because otherwise $\mathrm{b}(p, r_{j+1})$ would be a ball of radius smaller than $r$ containing the query point $q$.

  Therefore, $||xq|| \leq r \leq r_{j+1} = (1+\gamma/3)r_j < (1+\gamma/3)||pq|| \leq (1+\gamma/3)(1+\alpha)\mathrm{d}_P(q)$.

Thus in all valid cases, $x$ is a $(1+\gamma/3)(1+\alpha)$-approximate *NN* of $q$ in $P$. ∎

In the following theorem, we combine the results of all the previous lemmas to show that the construction of balls in algorithm **ConstructBalls** does indeed answer the approximate nearest-neighbor queries correctly.

**Theorem 2.2.5** *For a point set $P$ and an approximation factor $\epsilon$, Let the smallest ball containing $q$, in the balls formed by algorithm* **ConstructBalls**$(P, \gamma)$*, where $\gamma = \epsilon/2$, be of radius $r$, centered at $x$, then $x$ is a $(1+\epsilon)$-approximate* NN *of $q$ in $P$.*

*Proof.* Let $p$ be the *NN* of $q$ in $P$.

- **Case I :** $r < r_{high}(p)$

  - $r_{loss}(p) \leq r \leq r_{high}(p)$**:**
    If $r_{loss}(p) \leq r \leq r_{high}(p)$, then by the construction in algorithm **ConstructBalls**, since there exists a ballset in $[r_{low}(p), r_{high}(p)]$, using Lemma 2.2.3, $x$ is a $(1+\gamma/3)$-*NN* of $q$ in $P$.
  - $r < r_{loss}(p)$**:**
    If $r < r_{loss}(p)$, then using Lemma 2.2.4, $x$ is a $(1+\gamma/3)$-*NN* of $q$ in $P$.

- **Case II :** $r > r_{high}(parent^i(p))$ and $r < r_{high}(parent^{(i+1)}(p))$ for some $i$

  By Lemma 2.2.2, $parent^{i+1}(p)$ is a $((1+\gamma/(3\log n))^{i+1}$-approximate *NN* of $q$ in $P$.

– If $r_{loss}(parent^{i+1}(p)) \leq r \leq r_{high}(parent^{i+1}(p))$, then by the construction in algorithm **ConstructBalls**, since there exists a ballset in $[r_{low}(parent^{i+1}(p)), r_{high}(parent^{i+1}(p))]$, centered at $parent^{i+1}(p)$. Using Lemma 2.2.3, $x$ is a $((1+\gamma/(3\log n))^{i+1}(1+\gamma/3)$-approximate *NN* of $q$ in $P$.

– If $r < r_{loss}(parent^{i+1}(p))$, then using Lemma 2.2.4, $x$ is a $((1+\gamma/(3\log n))^{i+1}(1+\gamma/3)$-*NN* of $q$ in $P$.

Also $i+1 \leq \log n$, since height of the tree formed by the $\lambda$-stretch hierarchical clustering of $P$ is bound by $\log n$.

Therefore $x$ is a $((1+\gamma/(3\log n))^{\log n}(1+\gamma/3)$- approximate *NN* of $q$ in $P$.

- **Case III :** $r > r_{high}(t)$, where $t$ is the root of the tree formed by the $\lambda$-stretch hierarchical clustering of $P$

  There is no ball in $P$ of radius $> r_{high}(p)$. Thus $t$ is reported as the approximate-*NN* of $q$ in $P$ as $q$ lies in the universal ball centered at $t$.

  By Lemma 2.2.2, $t$ is $((1+\gamma/(3\log n))^{i}$-approximate *NN* of $q$ in $P$, where $i \leq \log n$, since height of the tree formed by the $\lambda$-stretch hierarchical clustering of $P$ is bound by $\log n$.

  Therefore $x$ is a $((1+\gamma/(3\log n))^{\log n}(1+\gamma/3)$- approximate *NN* of $q$ in $P$.

  And as $(1+\gamma/3)(1+\gamma/(3\log n))^{\log n} \leq 1+2\gamma$, $x$ is a $(1+2\gamma)$-approximate *NN* of $q$ in $P$.

Hence $x$ is a $(1+\epsilon)$-approximate *NN* of $q$ in $P$. ∎

### 2.2.4 A bound on the total number of balls required

We now analyze the number of balls that we construct in the algorithm **ConstructBalls**.

**Theorem 2.2.6** *The total number of balls constructed by the algorithm* **ConstructBalls**$(P, \epsilon/2)$ *is* $O((1/\epsilon^2)n\log(n/\epsilon))$.

*Proof.* The number of balls in a ball set is $O((1/\epsilon)\log_{1+O(\epsilon)}(n/\epsilon)) = O((1/\epsilon^2)\log(n/\epsilon))$. For each point, one ballset is constructed for the point by rule 1 of algorithm **ConstructBalls**. Additionally, one ballset is constructed for each child of the point by rule 2 of algorithm **ConstructBalls**. By charging the balls constructed around the child nodes (using rule 2) by algorithm **ConstructBalls** to the respective child nodes, the charge incurred at each point is at most that of 2 ball sets, i.e.,

$O((1/\epsilon^2)\log(n/\epsilon))$. Therefore, the total charge incurred over the $n$ points is $O((1/\epsilon^2)n\log(n/\epsilon))$. ∎

This improves the bound on the number of balls constructed in [33] by a factor of $\log n$.

## 2.3 Construction of the Data Structure

The balls constructed for the *PLSB* problem described in the previous section can be used to construct cells, i.e., quadtree boxes, and then store them in a BBD-tree as described in [33] (also see Lemma 2.1.2) resulting in an improvement of a factor of $\log n$ in space complexity and preprocessing time while keeping the query time logarithmic. This leads to our final result:

**Theorem 2.3.1** *Given a set $P$ on $n$ points in $\Re^d$ and a parameter $\varepsilon > 0$, one can compute a set $\mathcal{R}$ of $O(\frac{n}{\varepsilon^d}\log n/\varepsilon)$ regions, where each region is a cube or an annulus of cubes. The regions of $\mathcal{R}$ are disjoint and cover the space. Moreover, each region, $R$, has an associated point $p \in P$ such that for any query point $q$ contained in $R$, the point $p$ is an $\varepsilon$-nearest neighbor of $q$ in $P$. Moreover, this point, $p$ can be determined in $O(d^2 \log n/\varepsilon)$ steps. The time required for constructing the data structure is $O(\frac{n}{\varepsilon^d}\log^2 n/\varepsilon)$.*

*Proof.* The correctness and space complexity of the algorithm follow from Theorem 2.2.5 and Theorem 2.2.6 respectively together with Lemma 2.1.2.

The running time follows from Lemma 2.1.2 put together with the fact that it requires $O(nd\log^2 n)$ time to construct the $nd$-MST and the $nd$-hierarchical clustering. ∎

## 2.4 Recent Developments

Recently Har-Peled and Mendel [35] presented a near linear time algorithm for constructing hierarchical nets in finite metric spaces with constant doubling dimension. They apply this data structure to obtain an algorithm for nearest neighbor searching with query time $O(\frac{1}{\varepsilon^d} + 2^d \log n)$ and space $O(2^d n)$. Their results match those of Arya et al. [5] in Euclidean settings.

For high dimensions, recently, Indyk and Andoni [39] have presented an algorithm with query time $O(dn^{\rho(c)})$ utilizing space $O(dn+n^{1+\rho(c)})$, where $\rho(c) = 1/c^2 + O(\log\log n/\log^{1/3} n)$ which is optimal for locality-sensitive hashing based algorithms within a small constant factor.

# Chapter 3

# A Linear Time Algorithm for 2-means Clustering

The $k$-means clustering problem is known to be NP-hard even for $k = 2$ for arbitrary dimensions. One of the most popular heuristics used in practice for solving the $k$-means clustering problem is based on a simple iterative scheme for finding a locally optimal solution. This algorithm is often called the *$k$-means algorithm* or *Lloyd's algorithm*. For each center define its neighborhood to be those data points for which this center is the closest. In any locally minimum solution, each center lies at the centroid of its neighborhood. Such a solution is said to be centroidal. Lloyd's algorithm works by starting with any feasible solution and then repeatedly computes the neighborhood of each center and moves this center to the centroid of its neighborhood. This is repeated until some convergence criterion is satisfied. However, there exist examples, in which Lloyd's algorithm converges to a local minimum which is arbitrarily bad compared to the optimum solution. Consider for example points lying on a straight line at co-ordinates $0$, $x$, $x+y$ and $x+y+z$ where $z < x < y$. For $k = 3$, the optimal solution is to place the centers at $0$, $x$ and $x + y + z/2$ which has a cost of $z^2/4$. The local minimum when the centers are placed at points $x/2$, $x + y$ and $x + y + z$ has a cost of $x^2/4$ and is a stable point for Lloyd's algorithm. By increasing the ratio $x/z$, the approximation ratio for Lloyd's algorithm can be made arbitrarily high. For a detailed analysis of Lloyd's algorithm, see [36].

Given the intractability of the problem, the natural direction for research is to design approximation algorithms with arbitrarily small approximation ratios (PTAS). It is interesting that even for $k = 2$, no practical approximation schemes are known in high dimensions. Inaba et al. [37] gave

an algorithm to compute the $(1 + \frac{1}{\Omega(n\log n)})$ approximate clustering on the plane in $O(n^{5/3}\log^2 n)$ time with probability $(1 - \frac{1}{\Omega(n\log n)})$. For higher dimensions, the running time of their algorithm increases exponentially in $d$. Recently, Matousek [50] gave an $O(n\log n)$ deterministic algorithm for $(1+\varepsilon)$-approximate 2-*means clustering*. The algorithm is fairly complicated and relies on several results in computational geometry that depend heavily on the dimensions, leading to a running time of $O(n\log n.\varepsilon^{-2d}\log(1/\varepsilon) + n\varepsilon^{-(4d-2)}\log(1/\varepsilon))$.

One of the questions left open in [50] by Matousek was whether $\Omega(n\log n)$ is a lower bound for this problem. We present a randomized algorithm that determines a $(1+\varepsilon)$-approximate 2-*means clustering* of a given point set in time linear in the number of points for fixed dimensions, with constant probability. Parts of these results have appeared in [53]

In the 2-means clustering problem, given a set $P$ of $n$ data points in $d$-dimensional Euclidean space, $\Re^d$, the problem is to determine two centers $c_1$ and $c_2$ so as to minimize the mean squared Euclidean distance from each data point to its nearest center, Minimize $\sum_{p\in P}\mathrm{d}(p,\{c_1,c_2\})^2$.

The clusters $P_1$ and $P_2$ are derived from the centers $c_1$ and $c_2$ respectively by taking those points of $P$ for which corresponding center is closest. Thus the points are partitioned by the Voronoi diagram of the set of centers. The cost of the clustering is determined by $f_2(P_1,c_1) + f_2(P_2,c_2)$ where the cost function $f_2(Q,x) = \sum_{q\in Q}\mathrm{d}(q,x)^2$.

In Section 3.1, we present some preliminaries. The algorithm is presented in Section 3.2. The correctness of the algorithm is established in Section 3.3. Finally, in Section 3.4, we abstract some ideas of our algorithm that are important in formulating the general algorithm for solving a large class of clustering problems in linear time (cf. Chapters 4 and 5).

## 3.1 Preliminaries

We describe here some definitions and known results that will be used later in our algorithms.

### 3.1.1 Approximating the Centroid of a set of Points

**Definition** For a set of points $P$, define the *centroid*, $c(P)$, of $P$ as the point $\frac{\sum_{p\in P} p}{|P|}$.

**Fact 3.1** *Any optimal solution to the 1-means problem with respect to an input point set $P$ chooses the centroid $c(P)$, as the center.*

Inaba et. al. [37] showed that the centroid of a small random sample of points in $P$ can be a good approximation to $c(P)$.

**Lemma 3.2** [37] *Let $T$ be a set of $m$ points obtained by independently sampling $m$ points uniformly at random from a point set $P$. Then, for any $\delta > 0$,*

$$f_2(S, c(T)) \quad < \quad \left(1 + \frac{1}{\delta m}\right) f_2(S, c(S))$$

*holds with probability at least $1 - \delta$.*

Therefore, if we choose $m$ as $\frac{2}{\varepsilon}$, then with probability at least 1/2, we get a $(1+\varepsilon)$-approximation to the 1-*means clustering* cost by taking the center as the centroid of $T$. Thus, a constant size sample can quickly yield a good approximation to the optimal 1-means solution.

Note that this also establishes the existence of a set of $2/\varepsilon$ points of $P$, whose centroid is an $\varepsilon$-approximation to the centroid of $P$.

### 3.1.2 $\varepsilon$-near Pairs

We present here the concept of $\varepsilon$-near pairs and a result on such pairs due to Matousek [50].

**Definition** For a real number $\varepsilon \geq 0$, we define a relation $\sim_\varepsilon$ on (ordered) pairs of points in $d$-dimensional Euclidean space, $\Re^d$: we let $(x, y) \sim_\varepsilon (x', y')$ if $\|x - x'\| \leq \varepsilon \|x - y\|$ and $\|y - y'\| \leq \varepsilon \|x - y\|$. We say that $(x, y)$ and $(x', y')$ are *$\varepsilon$-near* if $(x, y) \sim_\varepsilon (x', y')$ and $(x', y') \sim_\varepsilon (x, y)$. The relation $\sim_\varepsilon$ is not "quite" symmetric.

The following Lemma is due to Matousek [50]. It states that if there are two pairs of centers that are close to each other, then the cost of the clusterings induced by the two pairs does not differ very much. We reproduce its short proof for the convenience of the reader.

**Lemma 3.3** [50] : *If $(x, y)$ and $(x', y')$ are $\varepsilon$-near pairs, $\varepsilon \leq \frac{1}{9}$ and if $C$ is the cost of the 2-means clustering induced by $(x, y)$ and $C'$ is the cost of the 2-means clustering induced by $(x', y')$, then,*

$$C \quad \leq \quad (1 + 16\varepsilon)C'.$$

*Proof.* Let $(S_1, S_2)$ be the partitioning of the point set w.r.t. $(x, y)$ with the points in $S_1$ ($S_2$ resp.) being closer to $x$ ($y$ resp.). Similarly, let $(S'_1, S'_2)$ be the partitioning of the point set w.r.t. $(x', y')$ with the points in $S'_1$ ($S'_2$ resp.) being closer to $x'$ ($y'$ resp.). By the optimality of the Voronoi clustering, it suffices to show that $f_2(S'_1, c_1) + f_2(S'_2, c_2) \leq (1 + 16\varepsilon) f_2(S_1, c_1) + f_2(S_2, c_2)$. For

this it is enough to show that for each $x \in S_1' \setminus S_1$, we have that $\mathrm{d}(x, c_1) \leq (1 + 6\varepsilon)\mathrm{d}(x, c_2)$ (a symmetric argument applies for $x \in S_2' \setminus S_2$) Now, since $x \in S_1'$, $\mathrm{d}(x, c_1') \leq \mathrm{d}(x, c_2')$. Let $\delta = \mathrm{d}(c_1', c_2')$. Then, $\mathrm{d}(x, c_1) \leq \mathrm{d}(x, c_1') + \varepsilon\delta \leq \mathrm{d}(x, c_2') + \varepsilon\delta \leq \mathrm{d}(x, c_2) + 2\varepsilon\delta$. Moreover, $\mathrm{d}(x, c_2) \geq \mathrm{d}(x, c_2') - \varepsilon\delta \geq \frac{1}{2}\delta - \varepsilon\delta$, which implies that $\delta \leq \frac{2}{1-2\varepsilon}\mathrm{d}(x, c_2) \leq 3\mathrm{d}(x, c_2)$. Thus we have that $\mathrm{d}(x, c_1) \leq (1 + 6\varepsilon)\mathrm{d}(x, c_2)$ as required. ∎

## 3.2 The Algorithm

The main ideas behind the algorithm are as follows. We first obtain candidate centers for the centroid of the larger cluster by random sampling. We then construct a set of lines through each of these candidate centers to obtain a line close enough to the line joining the optimal centroid pair. We reduce the *approximate* 2*-means clustering* problem to a set of *center location on a line* problems determined by these lines. We then solve the approximate version of the *center location on a line* problem (defined below formally) on these lines by random sampling. This gives us an approximation to the centroid of the smaller cluster.

We define the center location on a line problem as follows:

**Definition** *Center location on a line :* Given a set $P$ of $n$ data points in $d$-dimensional Euclidean space, $\Re^d$, a fixed point $c_1$ and a line $\ell$ passing through $c_1$, the problem is to find another point $c_2$ on the line $\ell$ so as to minimize the cost of the 2-means clustering induced by $(c_1, c_2)$. This measure, $\phi$, is the cost of the center location on the line problem and is defined as

$$\phi(P, c_1, \ell) \quad = \quad min_{c_2 \in \ell} \{f_2(P_1, c_1) + f_2(P_2, c_2)\}$$

where $(P_1, P_2)$ is the Voronoi partitioning induced by $(c_1, c_2)$.

The approximation version of this problem is stated as follows:

**Definition** *Approximate center location on a line :* Given a set $P$ of $n$ data points in $d$-dimensional Euclidean space, $\Re^d$, a fixed point $c_1$, a line $\ell$ passing through $c_1$ and an approximation factor $\varepsilon > 0$, the $(1 + \varepsilon)$-approximate center location on a line problem is to determine another point $c_2$ on the line $\ell$ such that

$$f_2(P_1, c_1) + f_2(P_2, c_2) \quad \leq \quad (1 + \varepsilon)\phi(P, c_1, \ell)$$

where $(P_1, P_2)$ is the Voronoi partitioning induced by $(c_1, c_2)$.

We also define a $(p, \theta)$-covering as follows:

**Definition** $(p, \theta)$*-covering :* Given a point $p$ in $d$-dimensional Euclidean space, $\Re^d$, and a parameter $\theta$ ($0 < \theta < 2\pi$), we define a $(p, \theta)$-covering to be a set of lines $\mathcal{Y}$ passing through $p$, such that, for any line $\ell$, passing through $p$, there exists a line $\ell'$ in $\mathcal{Y}$ that makes an angle at most $\theta$ with $\ell$.

The formal algorithm is described by the procedures 2-**means** and **CenterLocation** described in Figures 3.1 and 3.2 respectively.

Algorithm 2-**means** reduces the *approximate* 2*-means clustering* problem to a set of *center location on a line* problems. For this, we start by randomly sampling a set of points, from which we can derive a set of candidate centers that contains an approximation to the centroid of the larger cluster, with constant probability. We proceed with the remaining steps for each of the candidate centers in the set. Now that we have an approximation to one of the centroids, our goal is to approximate the other centroid. We first approximate the line joining the optimal centers. For this, we construct a set of lines (covering) passing through the first center and making a small angle with each other, such that at least one of the lines is close enough, in orientation, to the actual line joining the optimal centers. For each of the lines in the covering above and the corresponding approximation to the centroid, we solve the *center location on a line problem*, by calling algorithm **CenterLocation** (we will later show that by restricting the other center to lie on this line, we only suffer by a small factor in our approximation).

Algorithm **CenterLocation** solves the *approximate center location on a line* problem on each of the candidate lines constructed above (we will later show that it suffices to solve the approximation version of this problem, suffering only by a small factor in our approximation). We first project all the points onto this line, converting it to a problem in one dimension after transformation. Since there are only two directions to the line, the smaller cluster must lie either to the right or to the left. We repeat the remaining steps for each of the two directions. We solve the problem, by guessing the size of the smaller cluster within a constant factor. For each of the $O(\log n)$ possible guesses, we again take a random sample from the points on that side of the line and obtain a set of points that contains an approximation to the centroid of the smaller cluster, with constant probability.

We obtain the costs of the clusterings for all possible pairs formed in this manner and report the clustering with the least cost. We will show that this entire procedure can be performed in time linear in $n$.

**Algorithm** $2$-**means**$(P, \varepsilon)$
**Inputs :** $P$: Point set, $\varepsilon$: approximation factor
**Output :** A $(1+\varepsilon)$ *approximate 2-means clustering* of the points in $P$

1. Let $\alpha = \varepsilon/16$, $\theta = \varepsilon/1024$, $\delta = \varepsilon/4$
2. (Approximate the centroid of the larger cluster)
   (a) Randomly sample a set, $R$, of $8/\alpha$ points (independently drawn) from $P$.
   (b) Let $T$ be the set of centroids of all subsets of $R$ of size $2/\alpha$.
3. For each point $c_1'$ in $T$,
   (a) Construct a set $\mathcal{Y}$ of lines that is a $(c_1', \theta)$-covering
   (b) For each $\ell$ in $\mathcal{Y}$, solve **CenterLocation**( $P$, $c_1'$, $\ell$, $\delta$ )
4. Return the solution which has minimum cost.

Figure 3.1: The 2-means algorithm

For ease of readability, we suppress the linear dependence on the dimension $d$ when we analyze the algorithm, viz., linear time in our case means $O(nd)$ since each point has $d$ coordinates. However, we do state the dependence on the terms depending exponentially on $d$.

The proofs of correctness for the algorithms are detailed in section 3.3.

**Remark 3.1** *The centroid of a set of points, $P$ in $\Re^d$, is the linear combination of the centroids of any partitioning of the point set. Having obtained an approximation to the centroid of the larger cluster of a $2$-means clustering, say $c$, it would have been interesting if we could limit our search for the approximation of the other centroid to the line joining this centroid, $c$, to the centroid of the point set $P$. However, since we are dealing with the squares of the distances, it is not possible to get a decent bound on the approximation obtained by restricting ourselves to this line.*

## 3.3 Proof of Correctness

### 3.3.1 Reduction to approximate center location on lines

In this section we prove the correctness of the **2-means** algorithm and show that we can reduce the *approximate* $2$*-means clustering* problem to a set of *approximate center location on a line* problems so that the solution to one of these problems will correspond to a solution to the *approximate* $2$*-means clustering* problem, with constant probability.

**Algorithm CenterLocation**$(P, c_1, \ell, \delta)$
**Inputs :** $P$: Point set, $c_1$: fixed center,
$\ell$: line passing through $c_1$ on which to locate the second center
$\delta$: approximation factor
**Output :** The solution for a $(1+\delta)$-*approximate center location on line* $\ell$

1. Project the points of $P$ onto line $\ell$. Let $P'$ be the projected points.
2. (Assume center to be located is to the right of $c_1$)
   For $i$ = 1 to $\log |P'|$
   (a) Let $M_R$ be the right most $|P'|/2^i$ points of $P'$
   (b) Randomly sample a set, $R$ of $8/\delta$ points (independently drawn) from $M_R$.
   (c) Let $T$ be the set of centroids of all subsets of $R$ of size $2/\delta$.
   (d) For each $c_2'$ in $T$, Compute the cost of the clustering induced by $c_1, c_2'$.
3. Repeat step 2 assuming the center to be located is to the left of $c_1$
4. Return the solution which has minimum cost.

Figure 3.2: The approximate center location on a line algorithm

The following Lemma shows that we can obtain an approximation to the centroid of the larger cluster by random sampling.

**Lemma 3.4** *Let $(P_1, P_2)$ be the optimal 2-means clustering of $P$ and let $(c_1, c_2)$ be the corresponding centroids. Without loss of generality, assume $|P_1| \geq |P_2|$. Let $R$ be a random sample of size $8/\alpha$ obtained by independent draws at random from $P$. Let $T$ be the set of centroids of all possible subsets of $R$ of size $2/\alpha$. Then with constant probability, $T$ contains a point, $c_1'$, that is an $\alpha$-approximation of the centroid, $c_1$, i.e., $f_2(P_1, c_1') \leq (1+\alpha) f_2(P_1, c_1)$.*
*Moreover, $|T| = O((1/\alpha)^{O(1/\alpha)})$.*

*Proof.* As $|P_1| \geq |P_2|$, with constant probability, the sample $R$ contains at least $2/\alpha$ points of $P_1$. Hence, by Lemma 3.2, with constant probability, $T$ contains an $\alpha$-approximation to the centroid, $c_1$, of $P_1$. Clearly, the size of $T$ is $O((1/\alpha)^{O(1/\alpha)})$. ∎

The following Lemma shows that if we have a point $c_1'$ that is an $\alpha$-approximation to the centroid $c_1$ of one of the clusters in an optimal 2-means clustering, then the cost of the optimal 2-means clustering induced by $c_1'$ is within a factor of $(1+\alpha)$ of the cost of the optimal 2-means clustering.

**Lemma 3.5** *Let $(P_1, P_2)$ be the optimal 2-means clustering of $P$ and let $(c_1, c_2)$ be the corresponding centroids. Let $c_1'$ be an $\alpha$-approximation to the centroid $c_1$. Let $c_2'$ be the other center in the optimal 2-means clustering $(P_1', P_2')$ of $P$ when center $c_1'$ is fixed. Then*

$$f_2(P_1', c_1') + f_2(P_2', c_2') \leq (1+\alpha)(f_2(P_1, c_1) + f_2(P_2, c_2)).$$

*Proof.* As $c_1'$ is an $\alpha$-approximation to the centroid $c_1$, we have,

$$\begin{aligned} f_2(P_1, c_1') + f_2(P_2, c_2) &\leq (1+\alpha) f_2(P_1, c_1) + f_2(P_2, c_2) \\ &\leq (1+\alpha)(f_2(P_1, c_1) + f_2(P_2, c_2)). \end{aligned}$$

Also, by the optimality of the partitioning $(P_1', P_2')$ when the center $c_1'$ is fixed, we know that

$$f_2(P_1', c_1') + f_2(P_2', c_2') \leq f_2(P_1, c_1') + f_2(P_2, c_2).$$

Therefore,

$$f_2(P_1', c_1') + f_2(P_2', c_2') \leq (1+\alpha)(f_2(P_1, c_1) + f_2(P_2, c_2)).$$

∎

The following Lemma shows that when a center $c_1$ is fixed, and $(P_1, P_2)$ is the optimal 2-means clustering induced by $c_1$, and $c_2$ is the centroid of $P_2$, then, instead of finding $c_2$, it suffices to find a suitable point on a line that makes a small angle with the line joining $c_1$ and $c_2$ at the cost of a small factor in our approximation.

**Lemma 3.6** *Let $c_1$ and $c_2$ be points in $d$-dimensional Euclidean space, $\Re^d$. Let $(P_1, P_2)$ be the clustering induced by $(c_1, c_2)$. Let $\ell$ be the line joining $c_1$ and $c_2$. Let $0 < \beta \leq 1$ be a given constant. Let $\ell'$ be a line passing through $c_1$ that makes angle $\beta/64$ with $\ell$. Let $(P_1', P_3)$ be the optimal 2-means clustering induced by $c_1$ when the other center is also constrained to lie on $\ell'$. Let $c_3$ be the optimal center corresponding to $P_3$. Further let $c_3'$ be any other point lying on $\ell'$, such that,*

$$f_2(P_1'', c_1) + f_2(P_3', c_3') \leq (1 + \frac{\beta}{64})(f_2(P_1', c_1) + f_2(P_3, c_3)) \tag{3.1}$$

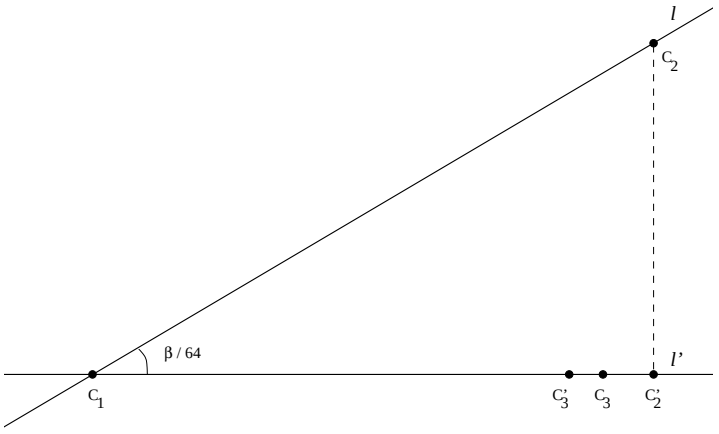


Figure 3.3: Approximation of the line connecting the centers

*where* $(P_1'', P_3')$ *is the clustering induced by* $(c_1, c_3')$*. Then,*

$$f_2(P_1'', c_1) + f_2(P_3', c_3') \quad \leq \quad (1+\beta)(f_2(P_1, c_1) + f_2(P_2, c_2)).$$

*Proof.* Let $c_2'$ be the projection of $c_2$ on $\ell'$ and let $(P_1''', P_2')$ be the clustering induced by $(c_1, c_2')$. By the optimality of the centroids $(c_1, c_3)$ on the line $\ell'$,

$$f_2(P_1', c_1) + f_2(P_3, c_3) \quad \leq \quad f_2(P_1''', c_1) + f_2(P_2', c_2'). \tag{3.2}$$

In the right triangle, $\triangle c_1 c_2 c_2'$, $\angle c_2 c_1 c_2' \leq \frac{\beta}{64}$ (see Figure 3.3). Since $\tan\theta \leq 2\theta$ for $\theta \leq 1/2$, we have, $c_2 c_2'/c_1 c_2 \leq \frac{\beta}{32}$ and $c_2 c_2'/c_1 c_2' \leq \frac{\beta}{32}$. This implies that the pairs of points $(c_1, c_2)$ and $(c_1 c_2')$ are $\frac{\beta}{32}$-near. Therefore, by Lemma 3.3,

$$f_2(P_1''', c_1) + f_2(P_2', c_2') \quad \leq \quad (1+\frac{\beta}{2})(f_2(P_1, c_1) + f_2(P_2, c_2)). \tag{3.3}$$

From equations (3.1), (3.2) and (3.3), it follows that

$$\begin{aligned} f_2(P_1'', c_1) + f_2(P_3', c_3') \quad &\leq \quad (1+\frac{\beta}{64})(1+\frac{\beta}{2})(f_2(P_1, c_1) + f_2(P_2, c_2)) \\ &\leq \quad (1+\beta)(f_2(P_1, c_1) + f_2(P_2, c_2)). \end{aligned}$$

■

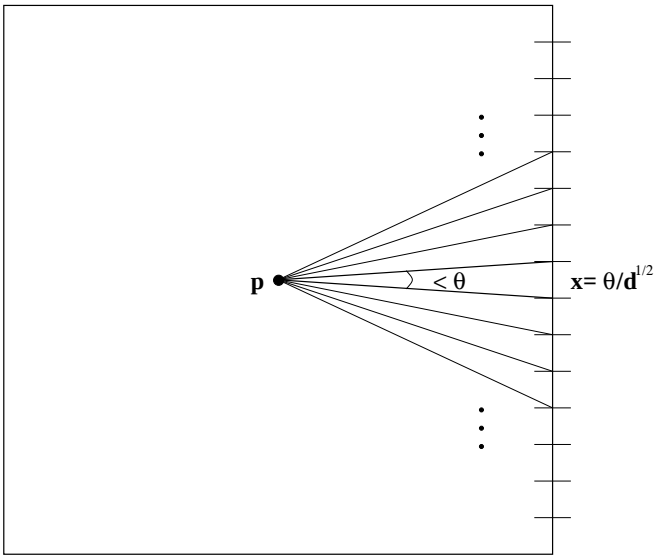


Figure 3.4: A $(p, \theta)$-covering in 2-dimensions ($d = 2$)

The following Lemma states that given a point $p$, it is possible to construct a set of $O((\frac{d}{\theta})^d)$ lines that is a $(p, \theta)$-covering. We present a simple proof for this. Tighter bounds involving more intricate proofs can be obtained for the number of lines constituting such a covering (e.g. see [62]).

**Lemma 3.7** *Given a point $p$ in $d$-dimensional Euclidean space, $\Re^d$, and an approximation angle $0 < \theta \leq 1$, we can construct a set $\mathcal{Y}$ of $O((\frac{d}{\theta})^d)$ lines passing through $p$, in $O((\frac{d}{\theta})^d)$ time, such that given any line, $\ell$, passing through $p$, there exists at least one line in $\mathcal{Y}$ that makes an angle at most $\theta$ with $\ell$.*

*Proof.* Consider a unit $d$-hypercube centered at a point $p$. The $d$ hypercube has $2d$ surfaces which are $(d-1)$-hypercubes. Consider the partitioning of each of these $(d-1)$-hypercubes into smaller equal $(d-1)$-hypercubes of side length $x$. Thus we get a set, $\mathcal{C}$ of $O((\frac{1}{x})^{d-1})$ $(d-1)$-hypercubes.

Consider the set, $\mathcal{L}$, of lines passing through the corners of the hypercubes in $\mathcal{C}$ and the point $p$. Clearly the largest angle between any two lines is formed between the lines passing through the opposite two corners of the hypercube located at the center of any surface (side of the $d$-hypercube) as this is the closest $(d-1)$-hypercube to $p$. Thus in order to obtain a line covering of angle at most $\theta$, it suffices to set $x$ small enough to ensure that the angle between these pair of lines is less than $\theta$.

It can be verified that setting $x = \frac{\theta}{\sqrt{d}}$ satisfies this condition, as $\tan\theta \geq \theta$ for $0 < \theta \leq 1$. Therefore we obtain a set, $\mathcal{L}$, of lines of size $O((\frac{d}{\theta})^d)$.

Figure 3.4 illustrates such a line covering for $d = 2$. The same can be extended to higher dimensions.

Clearly the time required to compute this set of lines is of the same order as its size. ∎

The following Lemma proves the correctness of the 2-**means** algorithm.

**Lemma 3.8** *Given an algorithm* **CenterLocation** *that determines a $(1+\delta)$-approximate solution to the center location on a line problem in time $\mathcal{T}(n, \delta)$, with constant probability, where $n$ is the number of input data points; the algorithm 2-***means** *determines a $(1+\varepsilon)$-approximate 2-means clustering, with constant probability, in time $O((1/\varepsilon)^{O(1/\varepsilon)}(d/\varepsilon)^d.\mathcal{T}(n, \varepsilon/4))$.*

*Proof.* Let $(P_1, P_2)$ be the optimal 2-means clustering and $(c_1, c_2)$ be the corresponding centroids.

Step 1 of algorithm 2-**means** sets $\alpha$, $\theta$ and $\delta$ to suitable constants (fractions of $\varepsilon$).

Applying Lemma 3.4, Step 2 of the algorithm determines a set $T$ of $O((1/\alpha)^{O(1/\alpha)})$ points, such that at least one of these points is an $\alpha$-approximation to the centroid of the larger cluster, with constant probability. Let $c_1'$ be such an $\alpha$-approximation (if it exists).

Step 3 $(a)$ of the algorithm determines a $(p, \theta)$-covering $\mathcal{Y}$ for every center, $p$, in $T$ (including $c_1'$). Applying Lemma 3.7, such a covering set of lines can be constructed in time $O((d/\theta)^d)$ and has size $O((d/\theta)^d)$. Let $\beta = 64\theta$. Then, we have $O((d/\beta)^d)$ lines passing through $p$, such that for any line passing through $p$, we have a line making an angle at most $\beta/64$.

In all, we have $O((d/\beta)^d)$ lines passing through each of the $O((1/\alpha)^{O(1/\alpha)})$ candidate centers, resulting in a total of $O((1/\alpha)^{O(1/\alpha)}(d/\beta)^d)$ lines. One of these lines, with constant probability, passes through $c_1'$, and makes an angle of less than $\beta/64$ with the line joining $c_1'$ and $c_2$ where $c_2$ is the other center of the optimal 2-means clustering, when the first center, $c_1'$ is fixed.

Therefore, applying Lemma 3.5 and 3.6, the solution to the $(1+\delta)$ *approximate center location on a line* problem on this line, computed in step $3-(b)$ of the algorithm, gives us a $(1+\alpha)(1+\beta)(1+\delta)$-*approximate 2-means clustering* of $P$. With the appropriate choices of $\alpha$, $\beta$ and $\delta$ (as specified in step 1 of the algorithm), we have a set of $O((1/\varepsilon)^{O(1/\varepsilon)}(d/\varepsilon)^d)$ lines, such that, the solution of the $(1+\varepsilon/4)$-*approximate center location on a line* problem on one of them gives us a solution to the $(1+\varepsilon)$-*approximate 2-means clustering* of $P$, with constant probability.

Since we are reporting the minimum cost clustering in step 4 of the algorithm, it is a $(1+\varepsilon)$-*approximate 2-means clustering*, with constant probability. ∎

### 3.3.2 Solving the problem for center location on a line

In this section we prove the correctness of the algorithm **CenterLocation** and show that we can solve the *approximate center location on a line* problem in linear time.

In the following Lemma, we show that the center location on a line problem is reducible to one dimension.

**Lemma 3.9** *Let $P$ be a set of points in $d$-dimensional Euclidean space, $\Re^d$. Let $\ell$ be a given line and let $c_1$ be a fixed point on $\ell$. Let $P'$ be the set of points obtained by projecting the points of $P$ on $\ell$. Consider the problem $\mathcal{P}$, of locating a point $c_2$ on $\ell$ such that the clustering of $P'$ induced by $(c_1, c_2)$ is optimal when $c_1$ and $c_2$ are constrained to lie on $\ell$ with $c_1$ fixed.*

*Then, the problem $\mathcal{P}$ is equivalent to the center location on a line problem. Moreover, an approximate solution to $\mathcal{P}$ is also an approximate solution to the center location on a line problem within the same approximation factor.*

*Proof.* Consider the transformation of the original axis to an orthogonal axis in which one of the axis is parallel to $\ell$. Let the point $p_i \in P$ be denoted by the vector $(x_{i1}, x_{i2}, .., x_{id})$ on the transformed orthogonal axis with $x_{i1}$ being the contribution towards the axis parallel to line $\ell$. Let $c_1$ and $c_2$ be represented by the vectors $(c_{11}, c_{12}, .., c_{1d})$ and $(c_{21}, c_{22}, .., c_{2d})$ respectively on the transformed axis. Then the cost of partition $(P_1, P_2)$ with centers $(c_1, c_2)$ can be computed as

$$\Delta = \sum\nolimits_{p_i \in P_1} \sum\nolimits_{j=1}^{d} (x_{ij} - c_{1j})^2 + \sum\nolimits_{p_i \in P_2} \sum\nolimits_{j=1}^{d} (x_{ij} - c_{2j})^2.$$

Now, since $c_1$ lies on $\ell$ and $c_2$ is also constrained to lie on $\ell$, therefore minimization of the above cost function can be reduced to the axis parallel to $\ell$ as follows:

$$\Delta_1 = \sum\nolimits_{p_i \in P_1} (x_{i1} - c_{11})^2 + \sum\nolimits_{p_i \in P_2} (x_{i2} - c_{22})^2.$$

This is because, the difference in the cost of a point $p_i \in P$ towards two distinct centers constrained to lie on line $\ell$ is only affected by the component along the axis parallel to $\ell$. The cost incurred along all the remaining axis remains unaffected.

Moreover, we also claim that a $(1+\delta)$-approximate clustering on the single dimension problem corresponds to a $(1+\delta)$-approximate clustering in the original space.

Let $(P_1', P_2')$ be a $(1+\delta)$-approximate clustering in a single dimension (axis parallel to $\ell$) with

centers $(c_1, c_2')$ both lying on $\ell$. Then

$$f_2((P_1', c_1) + f_2(P_2', c_2')) \quad \leq \quad (1+\delta)(f_2(P_1, c_1) + f_2(P_2, c_2)),$$

i.e.,

$$\begin{aligned}\sum\nolimits_{p_i \in P_1'}(x_{i1} - c_{11})^2 + \sum\nolimits_{p_i \in P_2'}(x_{i2} - c_{22}')^2 \\ \leq (1+\delta)(\sum\nolimits_{p_i \in P_1}(x_{i1} - c_{11})^2 + \sum\nolimits_{p_i \in P_2}(x_{i2} - c_{22})^2).\end{aligned}$$

Therefore,

$$\begin{aligned}\sum\nolimits_{p_i \in P_1'}\sum\nolimits_{j=1}^{d}(x_{ij} - c_{1j})^2 + \sum\nolimits_{p_i \in P_2'}\sum\nolimits_{j=1}^{d}(x_{ij} - c_{2j}')^2 \\ \leq (1+\delta)(\sum\nolimits_{p_i \in P_1}\sum\nolimits_{j=1}^{d}(x_{ij} - c_{1j})^2 + \sum\nolimits_{p_i \in P_2}\sum\nolimits_{j=1}^{d}(x_{ij} - c_{2j})^2).\end{aligned}$$

as components along all other dimensions are unaffected. ∎

We state here some observations that will be useful in our next proof.

**Observation 3.1** *Given two points $(c_1, c_2)$, let $\mathcal{H}$ be the perpendicular bisecting hyperplane of the line joining these points. This hyperplane determines the optimal clustering of a point set when there are two fixed centers $(c_1, c_2)$; the clusters are determined by the points lying on the same side of this hyperplane. As we move a hyperplane parallel to $\mathcal{H}$ in either direction, the cost function of the clustering due to the hyperplane with fixed centers $(c_1, c_2)$ is a monotonic non-decreasing function. This is clear since, as we move the hyperplane, the cost remains unchanged till we cross a point. When we cross a point, we disassociate this point from its closer center and associate it to the further center and this can only result in an increase in the cost of the clustering.*

**Observation 3.2** *Given a set of points, along with the centroid of the set and the cost of the cluster, on addition or removal of a point to/from this set, we can compute the new centroid and the cost of the new cluster in constant time. If $\mathbf{x}$ is the old centroid of a set $P$ of $n$ points and $\mathbf{x}_{n+1}$ is a new point, so that $P' = P \cup \{\mathbf{x}_{n+1}\}$ then the new centroid $\mathbf{x}_{new}$ is obtained as follows:*

$$\mathbf{x}_{new} = \frac{n}{n+1}\mathbf{x} + \frac{1}{n+1}\mathbf{x}_{n+1},$$

*and the cost of the clustering at $\mathbf{x}_{new}$ is obtained as follows:*

$$f_2(P', \mathbf{x}_{new}) = f_2(P, \mathbf{x}) + n \cdot d(\mathbf{x}, \mathbf{x}_{new})^2 + d(\mathbf{x}_{n+1}, \mathbf{x}_{new})^2.$$

*It follows that given a 2-partitioning of $P$ along with the centroids and the cost of the clustering due to the two partitions, if we move one point from one partition to another partition, then the cost of the new clustering and the new centroids can be determined in constant time.*

**Remark 3.2** *The Center Location on a line problem is solvable exactly by traversing the points from left to right one at a time and updating the cost of the clustering using the above observation. This can be done in two passes, first assuming the center lies to the right and then assuming that it lies to the left. However, note that this requires $O(n \log n)$ time as the points have to be sorted before the traversal can be performed. Our goal is to achieve linear running time.*

In the following Lemma, we prove the correctness of the algorithm **CenterLocation** and determine its running time.

**Lemma 3.10** *Let $P$ be a set of points in $d$-dimensional Euclidean space, $\Re^d$, and $\ell$ be a given line. Let $c_1$ be a fixed point on $\ell$. Then, algorithm* **CenterLocation** *determines in time $O((1/\delta)^{O(1/\delta)}n)$, a point $c_2'$ such that with constant probability, the cost of the clustering induced by $(c_1, c_2')$ is within a factor of $(1+\delta)$ of the cost of the clustering induced by $(c_1, c_2)$, i.e.,*

$$f_2(P_1', c_1) + f_2(P_2', c_2') \quad \leq \quad (1+\delta)(f_2(P_1, c_1) + f_2(P_2, c_2))$$

*where $c_2$ is the point on $\ell$ for which the clustering of $P$ induced by $(c_1, c_2)$ is optimal when $c_1$ is fixed, $(P_1, P_2)$ is the corresponding partitioning of the points and $(P_1', P_2')$ is the partitioning of $P$ induced by $(c_1, c_2')$.*

*Proof.* Step 1 of algorithm **CenterLocation** projects the points $P$ onto the line $\ell$. Applying Lemma 3.9, the problem of solving the *approximate center location on a line* problem is equivalent to solving the approximate version of the problem after projecting the points on $\ell$.

Step 2 of the algorithm assumes the center of the smaller cluster lies to the right of $c_1$ (The case when the center lies to the left of $c_1$ is handled similarly in step 3). In each iteration half the remaining points from the left side are eliminated as in step 2-(a) of the algorithm. Suppose in the current iteration we have a set $M$ of $m$ points. Initially $m = n$. We first locate the mid-point in $M$,

i.e., $m/2^{th}$ point, say $q$. This takes $O(m)$ time and partitions the point set, $M$, into two sets $M_L$ and $M_R$. $M_R$ is the remaining points after elimination. Thus, $M_R$ can be obtained from the points remaining in the previous iteration in $O(m)$ time.

Clearly, in some iteration, the number of points remaining, $|M_R|$, will be at most twice the size of $P_2$ and $P_2$ will be contained in $M_R$. Also, since the number of points are being halved in each iteration, the number of iterations required is at most $\log n$.

Assuming that we have the right partition, in step 2-(b) of the algorithm, we randomly sample a set $R$ of $8/\delta$ points from $M_R$. Note that although we only require $2/\delta$ points to approximate the centroid (Lemma 3.2), we take ample points in order to sample $2/\delta$ points from $P_2$ with constant probability, since we are approximating its size within a factor of 2. In step 2-(c) of the algorithm, we now take all possible subsets of the random sample $R$ of size $2/\delta$ and compute their centroids. Note that there are $O((1/\delta)^{O(1/\delta)})$ such points. With constant probability, one of these is a $\delta$-approximation to the centroid of $P_2$, i.e.,

$$f_2(P_2, c_2') \leq (1+\delta) f_2(P_2, c_2).$$

Therefore,

$$\begin{aligned} f_2(P_1', c_1) + f_2(P_2', c_2') &\leq f_2(P_1, c_1) + f_2(P_2, c_2') \\ &\leq f_2(P_1, c_1) + (1+\delta) f_2(P_2, c_2) \\ &\leq (1+\delta)(f_2(P_1, c_1) + f_2(P_2, c_2)). \end{aligned}$$

In step 2-(d) of the algorithm, we compute the cost of the clustering. We now show that the clustering cost can be computed in time linear in the number of remaining points. In each iteration, we are dealing with half the points from the previous iteration. Let $q$ be the mid point of $M$ as above. The cost of the clustering obtained by partitioning at $q$ can be determined in $O(m)$ time. This is possible because we already know the cost of the clustering due to $(M, P \backslash M)$ from the previous iteration and therefore we can partition the set $M$ into $M_L$ and $M_R$ in $O(m)$ time and then apply observation 3.2 over the set $M_L$. Note that in the current iteration we perform the sampling necessary for the guess $m/4 \leq |P_2| \leq m/2$. Let $\mathcal{H}'$ be the corresponding perpendicular bisecting hyperplane. If $\mathcal{H}'$ lies to the right of $q$, as we have already computed the cost of the clustering due to the partitioning at $q$, we only need to alter the cost over the remaining points in $M_R$ which is at most $m/2$ and can be done in time $O(m)$ by applying observation 3.2 at most $m/2$ times. If $\mathcal{H}'$ is to

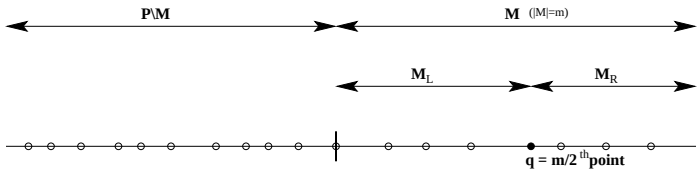


Figure 3.5: Sampling recursively by guessing the size of $|P_2|$

the left of $q$, let $\mathcal{H}$ be the optimal line bisecting the line joining the centroids $(c_1, c_2)$. In our current guess, $m/4 \leq |P_2| \leq m/2$. Therefore, the line $\mathcal{H}$ must lie to the right of $q$. Due to observation 3.1, the cost of the clustering due to the line $\mathcal{H}''$, passing through $q$ and parallel to $\mathcal{H}$, must be between that of $\mathcal{H}$ and $\mathcal{H}'$. Thus, if $\mathcal{H}'$ defines a $(1+\delta)$-*approximate* 2-*means clustering*, then so does $\mathcal{H}''$ for which we already know the cost. Hence we have restricted ourselves to altering the cost known at $q$ only over the set $M_R$ for each of the candidate centers. Hence the time required to compute the cost of the clustering in each iteration is $O(m)$. See Figure 3.5 for illustration.

As we are always recurring into the right half, the running time of this algorithm is characterized by the recurrence relation, $T(n) = T(n/2) + cn$, which is linear in $n$. Performing the same steps in the other direction in step 3 of the algorithm at most doubles the running time.

In step 4, we return the minimum of the costs of the clusterings due to the pairs formed from these centers with $c_1$ fixed. This gives us a $(1+\delta)$-approximation to the *center location on the line* problem, with constant probability. ∎

Putting everything together, we obtain the main result of the chapter:

**Theorem 3.11** *Algorithm* **2-means**, *using algorithm* **CenterLocation**, *determines a* $(1+\varepsilon)$-*approximate* 2-*means clustering of a point set* $P$ *in* $\Re^d$ *in time* $O((1/\varepsilon)^{O(1/\varepsilon)}(d/\varepsilon)^d n)$, *with constant probability.*

*Proof.* By Lemma 3.10, algorithm **CenterLocation** solves the $(1+\delta)$-*approximate center location on a line* problem in time $O((1/\delta)^{O(1/\delta)}n)$.

Therefore, applying Lemma 3.8 and using the fact that the algorithm **CenterLocation** is invoked with $\delta = \varepsilon/4$, we conclude that the algorithm **2-means** solves the *approximate 2-means clustering* problem in time $O((1/\varepsilon)^{O(1/\varepsilon)}(d/\varepsilon)^d n)$. ∎

## 3.4 Key Ideas

Some of the ideas used in this algorithm are the basis for the general algorithm for solving a large class of clustering problems in linear time that we present later (cf. Chapters 4 and 5). We list them as:

- The ability to progressively approximate the centers of the clusters in order of their sizes using random sampling.
- The ability to sample points from a cluster in the presence of extraneous points (that do not belong to the cluster) by iteratively eliminating a constant fraction of the remaining points.
- The ability to compute and compare the cost of $O(\log n)$ different clusterings in linear time.

# Chapter 4

# An Improved Algorithm for 2-means Clustering

In this chapter we present a simplified version of the general clustering algorithm described in the next chapter (c.f. Chapter 5), that works for the 2-*means clustering* problem. The 2-means clustering algorithm contains many of the ideas inherent in the more general algorithm and therefore makes it easier to understand the general algorithm.

The 2-*means clustering* algorithm presented in this chapter also attains linear running time with constant probability, but is an improvement over the 2-*means clustering* algorithm presented in the previous chapter (see Chapter 3) as the algorithm works for any dimensions (the expression for running time does not contain terms exponential in the number of dimensions).

In Section 4.1, we present some preliminary definitions and results. The algorithm and its proof of correctness are presented in Section 4.2. In Section 4.3, we abstract certain ideas and properties used in the 2-means algorithm. Our generalized algorithm (c.f. Chapter 5) will work for any clustering problems satisfying these properties.

## 4.1 Preliminaries

Recall our general definition of clustering problems. A clustering problem is defined by two parameters – an integer $k$ and a real-valued cost function $f(Q, x)$, where $Q$ is a set of points, and $x$ is a point in an Euclidean space. We shall denote this clustering problem as $\mathcal{C}(f, k)$. The input to $\mathcal{C}(f, k)$ is a set of points in a Euclidean space.

Given an instance $P$ of $n$ points, $\mathcal{C}(f,k)$ seeks to partition them into $k$ sets, which we shall denote as *clusters*. Let these clusters be $C_1,\ldots,C_k$. A solution also finds $k$ points, which we call *centers*, $c_1,\ldots,c_k$. We shall say that $c_i$ is the center of cluster $C_i$ (or the points in $C_i$ are assigned to $c_i$). The objective of the problem is to minimize the quantity $\sum_{i=1}^{k} f(C_i,c_i)$.

For the $k$-medians clustering problem, $f_1(Q,x)=\sum_{q\in Q}\mathrm{d}(q,x)$ and for the $k$-means clustering problem, $f_2(Q,x)=\sum_{q\in Q}\mathrm{d}(q,x)^2$. To handle discrete versions of these problems, i.e., cases where the centers have to be one of the points in $P$, we can make $f(Q,x)$ unbounded if $x\notin Q$.

**Definition** Given a point set $P$, let $\mathsf{OPT}_k(P)$ be the cost of the optimal solution to the clustering problem $\mathcal{C}(f,k)$ on input $P$.

**Definition** Given a set of points $P$ and a set of $k$ points $C$, let $\mathsf{OPT}_k(P,C)$ be the cost of the optimal solution to $\mathcal{C}(f,k)$ on $P$ when the set of centers is $C$.

We avoid parameterizing all our definitions by $f$ because the clustering problem will be clear from the context.

We now look at some properties of the 1-means problem.

For any point $x\in\Re^d$, it is easy to check that

$$f_2(P,x)=f_2(P,c(P))+|P|\cdot\mathrm{d}(c(P),x)^2. \tag{4.1}$$

We can deduce an important property of any optimal solution to the 2-means clustering problem. Suppose we are given an optimal solution to the 2-means clustering problem with respect to the input $P$. Let $C=\{c_1,c_2\}$ be the set of centers constructed by this solution. $C$ produces a partitioning of the point set $P$ into 2 clusters, namely, $P_1,P_2$. $P_i$ is the set of points for which the closest point in $C$ is $c_i$. In other words, the clusters correspond to the points in the Voronoi regions in $\Re^d$ with respect to $C$. Now, Fact 3.1 implies that $c_i$ must be the centroid of $P_i$ for $i=1,2$.

Recall from Lemma 3.2 that the centroid of a small random sample of points in $P$ can be a good approximation to $c(P)$.

Suppose $P'$ is a subset of $P$ and we want to get a good approximation to the optimal 1-mean for the point set $P'$. Following Lemma 3.2, we would like to sample from $P'$. But the problem is that $P'$ is not explicitly given to us. The following Lemma states that if the size of $P'$ is close to that of $P$, then we can sample a slightly larger set of points from $P$ and hopefully this sample would contain enough random samples from $P'$. Let us define things more formally first. Let $P$ be

a set of points and $P'$ be a subset of $P$ such that $|P'| \geq \theta|P|$, where $\theta$ is a constant between 0 and 1. Suppose we take a sample $S$ of size $\frac{4}{\theta\varepsilon}$ from $P$. Now we consider all possible subsets of size $\frac{2}{\varepsilon}$ of $S$. For each of these subsets $S'$, we compute its centroid $c(S')$, and consider this as a potential center for the 1-means problem instance on $P'$. In other words, we consider $f_2(P', c(S'))$ for all such subsets $S'$. The following Lemma shows that one of these subsets must give a close enough approximation to the optimal 1-means solution for $P'$.

**Lemma 4.1** *The following event happens with constant probability*

$$\min_{S':S'\subset S,|S'|=\frac{2}{\varepsilon}} f_2(P', c(S')) \leq (1+\varepsilon)\mathsf{OPT}_1(P')$$

*Proof.* With constant probability, $S$ contains at least $\frac{2}{\varepsilon}$ points from $P'$. The rest follows from Lemma 3.2. ∎

## 4.2 The Algorithm and Proof of Correctness

We now present the 2-means clustering algorithm. In the following proofs, we use the standard notation $\mathrm{b}(p, r)$ to denote the ball of radius $r$ around a point $p$.

**Theorem 4.2** *Given a point set $P$ of size $n$ in $\Re^d$, there exists an algorithm which produces a $(1+\varepsilon)$-approximation to the optimal 2-means solution on the point set $P$ with constant probability. Further, this algorithm runs in time $O(2^{(1/\varepsilon)^{O(1)}}dn)$.*

*Proof.* Let $\alpha = \varepsilon/64$. We can assume that $\mathsf{OPT}_2(P) > (1+\varepsilon/2)\mathsf{OPT}_1(P)$ otherwise the solution to the 1-mean problem for $P$ obtained by computing the centroid of $P$ in $O(nd)$ time that has cost at most $(1+\varepsilon/2)\mathsf{OPT}_2(P)$.

Consider an optimal 2-means solution for $P$. Let $c_1$ and $c_2$ be the two centers in this solution. Let $P_1$ be the points which are closer to $c_1$ than $c_2$ and $P_2$ be the points closer to $c_2$ than $c_1$. So $c_1$ is the centroid of $P_1$ and $c_2$ that of $P_2$. Without loss of generality, assume that $|P_1| \geq |P_2|$.

Since $|P_1| \geq |P|/2$, Lemma 4.1 implies that if we sample a set $S$ of size $O\left(\frac{1}{\varepsilon}\right)$ from $P$ and look at the set of centroids of all subsets of $S$ of size $\frac{2}{\varepsilon}$, then at least one of these centroids, call it $c_1'$ has the property that $f_2(P_1, c_1') \leq (1+\alpha)f_2(P_1, c_1)$. Since our algorithm is going to cycle through all such subsets of $S$, we can assume that we have found such a point $c_1'$.

Let the distance between $c_1$ and $c_2$ be $t$, i.e., $\mathrm{d}(c_1, c_2) = t$.

**Lemma 4.3** *$d(c_1, c_1') \leq t/4$.*

*Proof.* Suppose $\mathrm{d}(c_1, c_1') > t/4$. Equation (4.1) implies that

$$f_2(P_1, c_1') - f_2(P_1, c_1) = |P_1|\mathrm{d}(c_1, c_1')^2 \geq \frac{t^2|P_1|}{16}.$$

But we also know that left hand side is at most $\alpha f_2(P_1, c_1)$. Thus we get $t^2|P_1| \leq 16\alpha f_2(P_1, c_1)$.

Applying Equation (4.1) once again, we see that

$$f_2(P_1, c_2) = f_2(P_1, c_1) + t^2|P_1| \leq (1 + 16\alpha) f_2(P_1, c_1).$$

Therefore, $f_2(P, c_2) \leq (1+16\alpha) f_2(P_1, c_1) + f_2(P_2, c_2) \leq (1+16\alpha)\mathsf{OPT}_2(P)$. This contradicts the fact that $\mathsf{OPT}_1(P) > (1 + \varepsilon/2)\mathsf{OPT}_2(P)$.

This completes the proof of Lemma 4.3. ∎

Now consider the ball $\mathrm{b}(c_1', t/4)$. The previous Lemma implies that this ball is contained in the ball $\mathcal{B}(c_1, t/2)$ of radius $t/2$ centered at $c_1$. So $\mathcal{B}(c_1', t/4)$ only contains points in $P_1$. Since we are looking for the point $c_2$, we can delete the points in this ball and hope that the resulting point set has a good fraction of points from $P_2$.

This is what we prove next. Let $P_1'$ denote the point set $P_1 - \mathrm{b}(c_1', t/4)$. Let $P'$ denote $P_1' \cup P_2$. As we noted above $P_2$ is a subset of $P'$.

**Claim 4.2.1** *$|P_2| \geq \alpha |P_1'|$*

*Proof.* Suppose not, i.e., $|P_2| \leq \alpha |P_1'|$. Notice that

$$f_2(P_1, c_1') \geq f_2(P_1', c_1') \geq \frac{t^2|P_1'|}{16}.$$

Since $f_2(P_1, c_1') \leq (1 + \alpha) f_2(P_1, c_1)$, it follows that

$$t^2|P_1'| \leq 16(1 + \alpha) f_2(P_1, c_1) \tag{4.2}$$

So,

$$\begin{aligned}
f_2(P, c_1) &= f_2(P_1, c_1) + f_2(P_2, c_1) \\
&= f_2(P_1, c_1) + f_2(P_2, c_2) + t^2|P_2| \\
&= f_2(P_1, c_1) + f_2(P_2, c_2) + t^2\alpha|P_1'| \\
&\leq f_2(P_1, c_1) + f_2(P_2, c_2) \\
&\quad +16\alpha(1+\alpha)f_2(P_1, c_1) \\
&\leq (1+32\alpha)f_2(P_1, c_1) + f_2(P_2, c_2) \\
&\leq (1+32\alpha)\mathsf{OPT}_2(P),
\end{aligned}$$

where the second equation follows from Equation (4.1), while the fourth inequality follows from Inequality (4.2) and the fact that $|P_2| \leq \alpha|P_1'|$. But this contradicts the fact that $\mathsf{OPT}_1(P) > (1+\varepsilon/2)\mathsf{OPT}_2(P)$.

This completes the proof of Claim 4.2.1. ∎

The above claim combined with Lemma 3.2 implies that if we sample $O\left(\frac{1}{\alpha^2}\right)$ points from $P'$, and consider the centroids of all subsets of size $\frac{2}{\alpha}$ in this sample, then with constant probability we shall get a point $c_2'$ for which $f_2(P_2, c_2') \leq (1+\alpha)f_2(P_2, c_2)$. Thus, we get the centers $c_1'$ and $c_2'$ which satisfy the requirements of our Lemma.

The only problem is that we do not know the value of the parameter $t$. We will somehow need to guess this value and yet maintain the fact that our algorithm takes only linear amount of time.

We can assume that we have found $c_1'$ (this does not require any assumption on $t$). Now we need to sample from $P'$ (recall that $P'$ is the set of points obtained by removing the points in $P$ distant at most $t/4$ from $c_1'$). Suppose we know the parameter $i$ such that $\frac{n}{2^i} \leq |P'| \leq \frac{n}{2^{i-1}}$.

Consider the points of $P$ in descending order of distance from $c_1'$. Let $Q_i'$ be the first $\frac{n}{2^{i-1}}$ points in this sequence. Notice that $P'$ is a subset of $Q_i'$ and $|P'| \geq |Q_i'|/2$. Also we can find $Q_i'$ in linear time (because we can locate the point at position $\frac{n}{2^{i-1}}$ in linear time). Since $|P_2| \geq \alpha|P_1'|$, we see that $|P_2| \geq \alpha|Q_i'|/4$. Thus, Lemma 3.2 implies that it is enough to sample $O\left(\frac{1}{\alpha^2}\right)$ points from $Q_i'$ to locate $c_2'$ (with constant probability of course).

But the problem with this scheme is that we do not know the value $i$. One option is try all possible values of $i$, which will imply a running time of $O(n \log n)$ (treating the terms involving $\alpha$ and $d$ as constant). Also note that we cannot use approximate range searching because preprocessing takes $O(n\log n)$ time.

We somehow need to combine the sampling and the idea of guessing the value of $i$. Our algorithm proceeds as follows. It tries values of $i$ in the order $0, 1, 2, \ldots$. In iteration $i$, we find the set of points $Q'_i$. Note that $Q'_{i+1}$ is a subset of $Q'_i$. In fact $Q'_{i+1}$ is the half of $Q'_i$ which is farther from $c'_1$. So in iteration $(i+1)$, we can begin from the set of points $Q'_i$ (instead of $P'$). We can find the candidate point $c'_2$ by sampling from $Q'_{i+1}$. Thus we can find $Q'_{i+1}$ in time linear in $|Q'_{i+1}|$ only.

Further in iteration $i$, we also maintain the sum $f_2(P - Q'_i, c'_1)$. Since $f_2(P - Q'_{i+1}, c'_1) = f_2(P - Q'_i, c'_1) + f_2(Q'_i - Q'_{i+1}, c'_1)$, we can compute $f_2(P - Q'_{i+1}, c'_1)$ in iteration $i+1$ in time linear in $Q'_{i+1}$. This is needed because when we find a candidate $c'_2$ in iteration $i+1$, we need to compute the 2-means solution when all points in $P - Q'_i$ are assigned to $c'_1$ and the points in $Q'_i$ are assigned to the nearer of $c'_1$ and $c'_2$. We can do this in time linear in $|Q'_{i+1}|$ if we maintain the quantities $f_2(P - Q'_i, c'_1)$ for all $i$.

Thus, we see that iteration $i$ takes time linear in $|Q'_i|$. Since $|Q'_i|$'s decrease by a factor of 2, the overall running time for a given value of $c'_1$ is $O(2^{(1/\alpha)^{O(1)}} dn)$. Since the number of possible candidates for $c'_1$ is $O(2^{(1/\alpha)^{O(1)}})$, the running time is as stated.

**Claim 4.2.2** *The cost, $\Delta$, reported by the algorithm satisfies $\mathsf{OPT}_2(P) \leq \Delta \leq (1+\alpha)\mathsf{OPT}_2(P)$.*

*Proof.* $\mathsf{OPT}_2(P) \leq \Delta$ is obvious as we are associating each point with one of the 2 centers being reported and accumulating the corresponding cost. Now, consider the case when we have the candidate center set where each center is a $(1+\alpha)$-approximate centroid of it's respective cluster. As we are associating each point to the approximate centroid of the corresponding cluster or a center closer than it, it follows that $\Delta \leq (1+\alpha)\mathsf{OPT}_2(P)$. If we report the minimum cost clustering, $\mathcal{C}$, then since the actual cost of the clustering (due to the corresponding Voronoi partitioning) can only be better than the cost that we report (because we associate some points with approximate centroids of corresponding cluster rather than the closest center), we have $\mathsf{OPT}_2(P) \leq \Delta \leq (1+\alpha)\mathsf{OPT}_2(P)$. This proves Claim 4.2.2. ∎

This also completes the proof of Theorem 4.2. ∎

## 4.3 General Properties of Clustering Problems

In this section, we generalize certain ideas and properties that we used in the 2-means algorithm. Our generalized algorithm (c.f. Chapter 5) will work on any of the clustering problems defined in Section 1.2 provided these properties are satisfied.

The first idea that is used in the 2-means algorithm that we wish to generalize is that the 2-means clustering problem cannot be solved by simply solving the 1-means clustering problem at the expense of a small increase in the cost of the solution.

**Definition** We say that a point set $P$ is $(k, \alpha)$-*irreducible* if $\mathsf{OPT}_{k-1}(P) \geq (1 + \delta\alpha)\mathsf{OPT}_k(P)$, where $\delta$ is a constant determined by the nature of the clustering problem (e.g. $k$-median, $k$-means). Otherwise we say that the point set is $(k, \alpha)$-*reducible*.

Reducibility captures the fact that if $P$ is $(k, \alpha)$-reducible for a small constant $\alpha$, then the optimal solution for $\mathcal{C}(f, k-1)$ on $P$ is close to that for $\mathcal{C}(f, k)$ on $P$. So if we are solving the latter problem, it is enough to solve the former one. In fact, when solving the problem $\mathcal{C}(f, k)$ on the point set $P$, we can assume that $P$ is $(k, \alpha)$-irreducible, where $\alpha = \frac{\epsilon}{8\delta k}$. Indeed, suppose this is not the case. Let $i$ be the highest integer such that $P$ is $(i, \alpha)$-irreducible. Then, $\mathsf{OPT}_k(P) \leq (1 + \delta\alpha)^{k-i}\mathsf{OPT}_i(P) \leq (1 + \varepsilon/4)\mathsf{OPT}_i(P)$. Therefore, if we can get a $(1 + \varepsilon/4)$-approximation algorithm for $\mathcal{C}(f, i)$ on input $P$, then we have a $(1 + \varepsilon)$-approximation algorithm for $\mathcal{C}(f, k)$ on $P$. Thus it is enough to solve instances which are irreducible.

The first property that we want $\mathcal{C}(f, k)$ to satisfy is a fairly obvious one – it is always better to assign a point in $P$ to the nearest center. We state this more formally as follows :

> ***Closeness Property* :** Let $Q$ and $Q'$ be two disjoint set of points, and let $q \in Q$. Suppose $x$ and $x'$ are two points such that $\mathrm{d}(q, x) > \mathrm{d}(q, x')$. Then the cost function $f$ satisfies the following property
>
> $$f(Q, x) + f(Q', x') \geq f(Q - \{q\}, x) + f(Q' \cup \{q\}, x').$$

This is essentially saying that in order to find a solution, it is enough to find the set of $k$ centers. Once we have found the centers, the actual partitioning of $P$ is just the Voronoi partitioning with respect to these centers. It is easy to see that the $k$-means problem and the $k$-median problem (both the continuous and the discrete versions) satisfy this property.

We desire two more properties from $\mathcal{C}(f, k)$. The first property says that if we are solving $\mathcal{C}(f, 1)$, then there should be a simple random sampling algorithm. The second property says that suppose we have approximated the first $i$ centers of the optimal solution closely. Then we should be able to easily extract the points in $P$ which get assigned to these centers. We describe these properties in more detail below :

One of the key ingredients of the 2-means clustering algorithm described in the previous section (Lemma 3.2) is the ability to derive an approximate center for a set of points using a small (constant size) random sample from the point set. The generalization of this requirement is formally presented below.

> ***Random Sampling Procedure*** **:** There exists a procedure $\mathcal{A}$ that takes a set of points $Q \in \Re^d$ and a parameter $\alpha$ as input. $\mathcal{A}$ first randomly samples a small set $R$ of constant, $\lambda_\alpha$, points from $Q$. Starting from $R$, $\mathcal{A}$ produces another set of points, which we call core$(R)$, of constant size, $\beta_\alpha$. $\mathcal{A}$ satisfies the condition that with constant probability there is at least one point $c \in$ core$(R)$ such that $\mathsf{OPT}_1(Q, \{c\}) \leq (1+\alpha)\mathsf{OPT}_1(Q)$. Further the time taken by $\mathcal{A}$ to produce core$(R)$ from $R$ is at most $O(\eta_\alpha \cdot dn)$, where $n$ is the size of $Q$ and $\eta_\alpha$ is a constant.
>
> As described in the previous section, if we take a random sample $R$ of size $\lambda_\alpha = 2/\alpha$ points from the point set and compute its centroid, core$(R)$ of size $\beta_\alpha = 1$, then with constant probability, this centroid is a $(1+\alpha)$ approximate to the mean of point set.
>
> However, in the course of our algorithm, the set $Q$ will not be explicitly known - instead we sample from a superset $P \supseteq Q$. We will sample a slightly larger set of points from $P$ and then we isolate a $\lambda_\alpha$ subset that consists only of points in $Q$. Although we are not directly sampling from $Q$, our sampling/isolation procedure must ensure that all $\lambda_\alpha$ subsets of $Q$ are equally likely in the same way if we had directly sampled from $Q$.

We now generalize the result of the previous section describing how we can perform the random sampling procedure from a point set even when there exist extra points. Let $P$ be a set of points and $Q$ be a subset of $P$ such that $|Q| \geq \theta|P|$, where $\theta$ is a constant between 0 and 1. Suppose that we require a sample of $\lambda_\alpha$ points from $Q$ to generate the candidate center set that contains a $(1+\alpha)$-approximation to the 1-center. Suppose we take a sample $S$ of size $O(\frac{4\lambda_\alpha}{\theta})$ from $P$. Now we consider all possible subsets of size $\lambda_\alpha$ of $S$. For each of these subsets $S'$, we can generate a candidate centre set for the 1-center for the clustering problem using the *Random Sampling Procedure*, $\mathcal{A}$, and consider each one of these as a potential 1-center for the clustering problem instance on $Q$. The following Lemma states that one of these subsets must give a close enough approximation to the optimal 1-center solution for $Q$.

**Lemma 4.4** *(Superset Sampling Lemma) Let $\mathcal{CEN}(S')$ be the centre set generated using the* Random Sampling Procedure *on sampled subset $S'$. Then, the following event happens with constant*

*probability*

$$\min_{c' \in \mathcal{CEN}(S'):S' \subset S,|S'|=\lambda_\alpha} f(Q,c') \leq (1+\alpha)\mathsf{OPT}_1(Q).$$

*Proof.* With constant probability, $S$ contains at least $\lambda_\alpha$ points from $Q$, the required sample size. Clearly the set $S'$ of $\lambda_\alpha$ points is equiprobable amongst all the point sets of the same size in $Q$ (obtained with replacement). The rest follows from the *Random Sampling Procedure* for the clustering problem. ∎

We will later see that our generalized algorithm yields a running time of $O(2^{(k/\varepsilon)^{O(1)}}nd)$ when $\lambda_\alpha = O((\frac{1}{\alpha})^{O(1)})$, $\beta_\alpha = O(2^{(\frac{1}{\alpha})^{O(1)}})$ and $\eta_\alpha = O(2^{(\frac{1}{\alpha})^{O(1)}})$.

Another important property of the 2-means clustering algorithm described in the previous section is the ability to obtain a random sample of points from the smaller cluster after carefully removing some points and then performing random sampling. Once we have approximated the center of the larger cluster, we can remove enough of its points from around this center. This leaves us with a point set, such that the smaller cluster forms a constant fraction of this point set. Thus, we can then obtain a random sample of points from the smaller cluster of the required constant size by taking a constant size random sample from the remaining point set.

We now generalize this property. Consider the optimal $k$-clustering of a point set, $P$, where $i$ centers are fixed (these correspond to the centers that have already been approximated) and $k-i$ centers are free to be selected from the centers of the optimal solution (these correspond to the centers that are yet to be determined). We refer to the clusters centered at the fixed points as fixed clusters and the clusters centered at the free centers as free clusters (see Figure 4.1). There exists a value $r$ such that if we construct balls of radius $r$ around the $i$ fixed centers, then these balls contain enough points ($S$) of the fixed clusters, such that the free clusters form a constant fraction of the remaining points lying outside these balls ($P - S$). This allows us to obtain a random sample of points from the largest free cluster of the required constant size by taking a costant size random sample from the remaining point set. This generalization is formally stated below.

> ***Tightness Property*** **:** Let $P$ be a set of points which is $(k,\alpha)$-irreducible for some constant $\alpha$. Consider an optimal solution to $\mathcal{C}(f,k)$ on $P$ – let $C = \{c_1,\ldots,c_k\}$ be the centers in this solution. Suppose we have a set of $i$ points $C'_i = \{c'_1,\ldots,c'_i\}$, such that $\mathsf{OPT}_k(P,C') \leq (1+\alpha/k)^i\mathsf{OPT}_k(P)$, where $C' = \{c'_1,\ldots,c'_i,c_{i+1},\ldots,c_k\}$. Let $P'_1,\ldots,P'_k$ be the partitioning of $P$ if we choose $C'$ as the set of centers (in other words this is the Voronoi partitioning of $P$ with respect to $C'$). We assume w.l.o.g. that $P'_{i+1}$ is the largest cluster amongst $P'_{i+1},\ldots,P'_k$. Then there exists a set of points $S$ such that the following conditions hold :

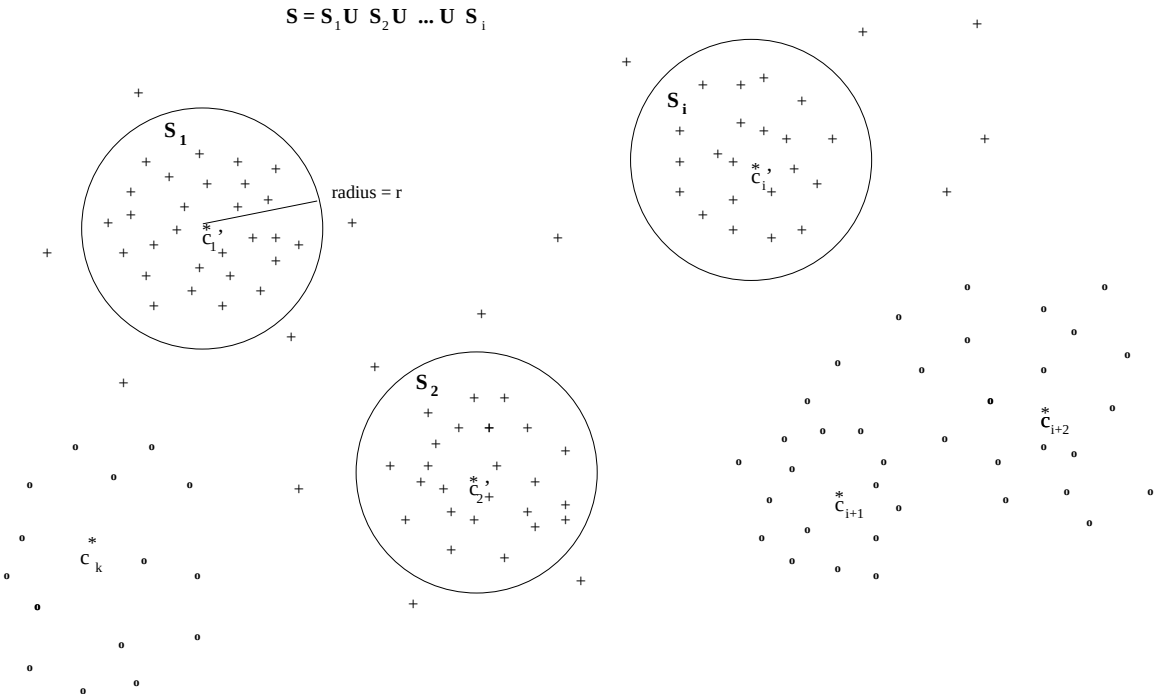


Figure 4.1: Illustration of Tightness Property

(a) $S$ is contained in $P'_1 \cup \ldots \cup P'_i$.

(b) Let $x \in S, x' \in P - S$. Then, $\mathrm{d}(x, \{c'_1, \ldots, c'_i\}) \leq \mathrm{d}(x', \{c'_1, \ldots, c'_i\})$.

(c) $P - S$ contains at most $\frac{|P'_{i+1}|}{\alpha^{\mathcal{O}(1)}}$ points of $P'_1 \cup \ldots \cup P'_i$.

We exemplify the existence of the tightness property for the $k$-means clustering problems.

**Lemma 4.5** *The* $k$-means *and* discrete $k$-means *clustering problems, having cost function* $f_2(Q, x) = \sum_{q \in Q} d(q, x)^2$ *satisfy the tightness property.*

*Proof.* We essentially need to show the existence of the desired set $S$, described in the definition above.

Recall that in the discrete version of the problem, $f_2(Q, x) = \infty$ when $x$ is not a point of the input point set. Consider the closest pair of centers between the sets $C' \backslash C'_i$ and $C'_i$ – let these centers be $c_l$ and $c'_r$ respectively. Let $t = \mathrm{d}(c_l, c'_r)$. Let $S$ be the set of points $\mathrm{b}(c'_1, t/4) \cup \cdots \cup \mathrm{b}(c'_i, t/4)$.

Clearly, $S$ is contained in $P'_1 \cup \cdots \cup P'_i$. This shows (a). Also, for any $x \in S, x' \in P - S$, $\mathsf{d}(x, \{c'_1, \ldots, c'_i\}) \leq \mathsf{d}(x', \{c'_1, \ldots, c'_i\})$. This proves (b).

Suppose $P - S$ contains more than $|P_l|/\alpha^2$ points of $P'_1 \cup \cdots \cup P'_i$. In that case, these points are assigned to centers at distance at least $t/4$. It follows that $\mathsf{OPT}_k(P, C')$ is at least $\frac{t^2|P_l|}{16\alpha^2}$. This implies that $t^2|P_l| \leq 16\alpha^2\mathsf{OPT}_k(P, C')$. Let $m_l$ be the centroid of $P_l$. Further, let $T_l$ be the average cost paid by $P_l$ when the center is its centroid, i.e., $T_l = \frac{\sum_{p \in P_l} \mathsf{d}(p,m_l)^2}{|P_l|}$. Observe that $f_2(P_l, c_l) = |P_l|(T_l + \mathsf{d}(c_l, m_l)^2)$. Therefore, if we assign the points in $P_l$ from $c_l$ to $c'_r$, the increase in cost is

$$\begin{aligned} |P_l|\left(\mathsf{d}(c'_r, m_l)^2 - \mathsf{d}(c_l, m_l)^2\right) &\leq |P_l|\left((\mathsf{d}(c'_r, c_l) + \mathsf{d}(c_l, m_l))^2 - \mathsf{d}(c_l, m_l)^2\right) \\ &\leq |P_l|\left(t^2 + 2t\mathsf{d}(c_l, m_l)\right) \end{aligned}$$

We know that the first term above, i.e., $|P_l|t^2$ is at most $16\alpha^2\mathsf{OPT}_k(P, C')$. We now need to bound the second term only. We consider two cases

- $t \leq \alpha\mathsf{d}(c_l, c_m)$ : In this case, $|P_l| \cdot 2t\mathsf{d}(c_l, m_l) \leq 2\alpha\mathsf{d}(c_l, m_l)^2|P_l| \leq 2\alpha f_2(P_l, c_l) \leq 2\alpha\mathsf{OPT}_k(P, C')$.
- $t > \alpha\mathsf{d}(c_l, c_m)$ : In this case, $|P_l| \cdot 2t\mathsf{d}(c_l, m_l) \leq \frac{2t^2|P_l|}{\alpha} \leq 32\alpha\mathsf{OPT}_k(P, C')$.

Thus, in either case, the cost increases by at most

$$48\alpha\mathsf{OPT}_k(P, C') \leq 48\alpha(1 + \alpha/k)^i\mathsf{OPT}_k(P) \leq 48\alpha(1 + \alpha/k)^k\mathsf{OPT}_k(P) \leq 144\alpha\mathsf{OPT}_k(P).$$

But this contradicts the fact that $P$ is $(k, \alpha)$-irreducible for $\delta = 144$. This proves the tightness property. ∎

We now make some important observations about the *Tightness Property* in the case when there are multiset (coincident) points. These observations are important in solving the weighted versions of the clustering problems efficiently (c.f. Section 5.6).

**Observation 4.1** *In a multiset clustering problem, given a set of $k$ centers, there always exists an optimal clustering (for these $k$ centers) in which all the points that share the same co-ordinates are assigned to the same center.*

By the *Closeness Property*, every point is assigned to its closest center. Now, if there are two centers equidistant from a point, it does not matter which center they get assigned to as this does

not change the cost of the solution. Therefore we can always modify an optimal clustering to get another clustering satisfying the above condition which has the same cost.

**Observation 4.2** *In a multiset clustering problem, the* Tightness Property *can be extended to ensure that any pair of coincident points either belong to $S$ or to $P - S$, i.e., they are not split between $S$ and $P - S$.*

This follows from the fact that the optimal clustering is determined by the Voronoi partitioning of the point set (*Closeness Property* ) and Observation 4.1 above. Let $S$ be a set satisfying the conditions of the tightness property. Consider the points of $P - S$ that are coincident with some point of $S$. Let this set of points be $X$. Then it can be easily verified that the set $S \cup X$ also satisfies the conditions of the tightness property and any pair of coincident points either belong to $S \cup X$ or $P - (S \cup X)$.

# Chapter 5

# A General Algorithm for Clustering

In this chapter, we present a general approach for designing approximation algorithms for a class of geometric clustering problems in arbitrary dimensions. More specifically, our approach leads to simple randomized algorithms for the $k$-means, $k$-medians and discrete $k$-means problems that yield $(1+\varepsilon)$ approximations with probability $\geq 1/2$ and running time $\in O(2^{(k/\varepsilon)^{O(1)}}dn)$. Our method is general in the sense that it is applicable to clustering problems satisfying certain simple properties and is likely to have further applications. Parts of these results have appeared in [45] and [46].

We can show that if a clustering problem $\mathcal{C}(f,k)$ satisfies the conditions stated in the previous chapter (see Section 4.3), then there is an algorithm which with constant probability produces a solution within $(1+\varepsilon)$ factor of the optimal cost. Further the running time of this algorithm is $O(2^{\left(\frac{k}{\varepsilon}\right)^{O(1)}} \cdot dn)$.

Fix a clustering problem $\mathcal{C}(f,k)$. Fix an instance consisting of a set $P$ of $n$ points in $\Re^d$. Suppose we are given a constant $\varepsilon > 0$. We would like to construct a solution for the clustering problem whose cost is within $(1+\varepsilon)$ of the optimal cost.

In Section 5.1, we present a brief outline of our algorithm. In Section 5.2, we describe a general approach for solving clustering problems efficiently. In subsequent sections we give applications of the general method by showing that this class of problems includes the $k$-means, $k$-medians and discrete $k$-means problems. In Section 5.4.3, we also describe an efficient approximation algorithm for the 1-median problem. In Section 5.6, we extend our algorithms for efficiently handling weighted point sets.

## 5.1 Outline

We present a brief outline of the algorithm.

We can assume that the solution is irreducible, i.e., removing one of the centers does not create a solution which has cost within a small factor of the optimal solution.

We start with $k$ optimal (unknown) centers. In each iteration, we will consider the optimal clustering formed by $i$ currently known centers and $k - i$ optimal (unknown) centers. Call this the optimal clustering of the current iteration. Our goal will be to approximate the next largest cluster, so that the resulting clustering is an approximation to the optimal clustering of the current iteration. We will bound the overall approximation factor to within a factor of $(1+\varepsilon)$ of the optimal clustering.

One of the main challenges is to accurately estimate the centers of the (optimal) clustering that may have widely varying sizes [9, 53]. In order to obtain a linear time algorithm, this must be done efficiently.

Suppose we have found centers $c'_1, \ldots, c'_i$. As the clustering problem satisfies the tightness property, we know that there exists a value $r$ such that the points at a distance of at most $r$ from $\{c'_1, \ldots, c'_i\}$ (set $S$ of the tightness property) get assigned to $c_1, \ldots, c_i$ by the optimal solution induced by these $k$ centers. So, we can delete these points. Without loss of generality, we assume that the largest cluster from amongst those centered around the unknown clusters $\{c_{i+1}, \ldots, c_k\}$ is centered around $c_{i+1}$. Let $P'_{i+1}$ be this clustering. Now we can show that among the remaining points, the size of $P'_{i+1}$ is significant. Therefore, we can use random sampling to obtain a center $c'_{i+1}$ which is a pretty good estimate of $c_{i+1}$. Of course we do not know the value of $r$, and so a naive implementation of this idea gives an $O(n(\log n)^k)$ time algorithm.

But now we want to modify it to a linear time algorithm. This is where the algorithm gets more involved. As mentioned above, we can not guess the parameter $r$. So we try to guess the size of the point set obtained by removing the points in the balls of radius $r$ around $\{c_1, \ldots, c_i\}$ ($P - S$). So we work with the remaining point set with the hope that the time taken for this remaining point set will also be small and so the overall time will be linear. Now, we describe the actual clustering algorithm.

## 5.2 The Algorithm

The algorithm is described in Figures 5.1 and 5.2. Figure 5.1 is the main algorithm. The inputs are the point set $P$, $k$ and an approximation factor $\varepsilon$. Let $\alpha$ denote $\frac{\varepsilon}{8\delta k}$, where $\delta$ is a suitable

**Algorithm** $k$**-clustering**$(P, k, \varepsilon)$
**Inputs :** Point set $P$, number of clusters $k$, approximation ratio $\varepsilon$.
**Output :** $k$-clustering of $P$.

1. For $i = 1$ to $k$ do
   Obtain the clustering **Irred-**$k$**-clustering**$(P, i, i, \phi, \varepsilon/(8\delta k), 0)$.
2. Return the clustering which has minimum cost.

Figure 5.1: The $k$-clustering Algorithm

constant depending on the nature of the clustering problem, as determined by the irreducibility of the clustering problem (see Definition 4.3). The algorithm $k$-**clustering**$(P, k, \varepsilon)$ tries to find the highest $i$ such that $P$ is $(i, \alpha)$-irreducible. Essentially we are saying that it is enough to find $i$ centers only. Since we do not know this value of $i$, the algorithm tries all possible values of $i$.

We now describe the algorithm **Irred-**$k$**-clustering**$(Q, m, k, C, \alpha, \text{Sum})$. We have found a set $C$ of $k - m$ centers already. The points in $P - Q$ have been assigned to $C$. We need to assign the remaining points in $Q$. The case $m = 0$ is clear. In step 2, we try to find a new center that is a $(1 + \alpha/k)$-approximation to the 1-center of the next largest cluster by the *Random Sampling Procedure*, $\mathcal{A}$, for the clustering problem. This will work provided a good fraction of the points in $Q$ do not get assigned to $C$. If this is not the case then in step 3, we assign half of the points in $Q$ to $C$ and call the algorithm recursively with this reduced point set. For the base case, when $|C| = 0$, as $P_1$ is the largest cluster, we require to sample only $O(k\lambda_{\alpha/k})$ points, where $\lambda_{\alpha/k}$ is the sample size required by the *Random Sampling Procedure*, $\mathcal{A}$. This is tackled in Step 2. Step 3 is not performed in this case, as there are no centers.

## 5.3 Analysis and Proof of Correctness

We can now show that if we have the *Random Sampling Procedure* described above, then we can get a $(1 + \varepsilon)$-approximation algorithm for the clustering problem with constant probability. Further the running time of the algorithm is $O(2^{(\frac{k}{\varepsilon})^{O(1)}} dn)$.

**Theorem 5.1** *Suppose a point set $P$ is $(k, \alpha)$-irreducible. Then the algorithm*
**Irred-**$k$**-clustering**$(P, k, k, \emptyset, \alpha, 0)$ *returns a solution to the clustering problem $\mathcal{C}(f, k)$ on input $P$*

**Algorithm Irred-$k$-clustering**$(Q, m, k, C, \alpha, \text{Sum})$
**Inputs :** $Q$: Remaining point set, $m$: number of cluster centers yet to be found, $k$: total number of clusters,
$C$: set of $k - m$ cluster centers found so far,
$\alpha$: approximation factor, $Sum$: the cost of assigning points in $P - Q$ to the centers in $C$
**Output :** The clustering of the points in $Q$ in $k$ clusters.

1. If $m = 0$
   Assign the points in $Q$ to the nearest centers in $C$.
   Sum = Sum + The clustering cost of $Q$.
   Return the clustering.
2. (a) Sample a set $S$ of size $O(\lambda_{\alpha/k} k/\alpha^{O(1)})$ from $Q$.
   (b) For each subset $S'$ of $S$ of size $\lambda_{\alpha/k}$ do
   Compute the candidate centre set from $S'$.
   For each center, c, in the candidate center set, obtain the clustering **Irred-$k$-clustering**$(Q, m-1, k, C \cup \{c\}, \alpha, \text{Sum})$.
3. (a) Consider the points in $Q$ in ascending order of distance from $C$.
   (b) Let $U$ be the first $|Q|/2$ points in this sequence.
   (c) Assign the points in $U$ to the nearest centers in $C$.
   (d) Sum = Sum + The clustering cost of $U$.
   (e) Compute the clustering **Irred-$k$-clustering**$(Q - U, m, k, C, \alpha, \text{Sum})$.
4. Return the clustering which has minimum cost.

Figure 5.2: The irreducible $k$-clustering algorithm

*of cost at most* $(1+\alpha)\mathsf{OPT}_k(P)$ *with probability* $\gamma^k$*, where* $\gamma$ *is a constant.*

*Proof.* Consider an optimal solution to $\mathcal{C}(f, k)$ on input $P$. Let the centers be $\mathcal{K} = \{c_1, \ldots, c_k\}$ and let these partition the point set $P$ into clusters $P_1, \ldots, P_k$ respectively. The only source of randomization in our algorithm is the invocations to the superset sampling Lemma (Lemma 4.4). Recall that the desired event in the superset sampling Lemma happens with constant probability. For ease of exposition, we shall assume that this desired event in fact always happens when we invoke this Lemma. At the end of this proof, we will compute the actual probability with which our algorithm succeeds. Thus, unless otherwise stated, we assume that the desired event in the superset sampling Lemma always happens.

Observe that when we call **Irred-$k$-clustering** with input $(P, k, k, \emptyset, \alpha, 0)$, it gets called recursively again several times (although with different parameters). Let $\mathcal{W}$ be the set of all calls to **Irred-$k$-clustering** when we start it with input $(P, k, k, \emptyset, \alpha, 0)$. Let $\mathcal{W}_i$ be those calls in $\mathcal{W}$ in which the parameter $C$ (i.e., the set of centers already found) has size $i$.

For all values of $i$, our algorithm shall maintain the following invariant :

**Invariant :** The set $\mathcal{W}_i$ contains a call in which the list of parameters $(Q, m, k, C, \alpha, \text{Sum})$ has the following properties :

(1) If the optimal solution, $\mathcal{K}$, is $(k, \alpha)$-irreducible and $C'_i = \{c'_1, \ldots, c'_i\}$ is a set of $i$ known centers then there exists a set $C''_i = \{c_{i+1}, \ldots, c_k\}$ of $k - i$ unknown centers, $C''_i \subseteq \mathcal{K}$, such that $\mathsf{OPT}_k(P, C') \leq (1 + \alpha/k)^i \mathsf{OPT}_k(P)$, where $C' = C'_i \cup C''_i$.

(2) Let $P'_1, \ldots, P'_k$ be the partitioning of $P$ if we choose $C'$ as the set of centers (in other words this is the Voronoi partitioning of $P$ with respect to $C'$), where $C'$ is as defined above. Then the set $P - Q$ is a subset of $P'_1 \cup \cdots \cup P'_i$.

Clearly, if we show that the invariant holds for $i = k$, then we are done. It holds trivially for $i = 0$. Suppose the invariant holds for some fixed $i$. We shall show that the invariant holds for $(i + 1)$ as well. We assume w.l.o.g. that $P'_{i+1}$ is the largest cluster amongst $P'_{i+1} \cup \cdots \cup P'_k$. Our algorithm approximates the center of $P'_{i+1}$ and we show that this center when added to $C'_i$ forms the required set of centers $C'_{i+1}$ for the next iteration.

As the invariant holds for $i$, there exist parameter lists in $\mathcal{W}_i$ which satisfy the invariant properties mentioned above. Consider any such parameter list. As the conditions of the *Tightness Property* are met, there exists a set $S$ contained in $P'_1 \cup \cdots \cup P'_i$ such that $P - S$ contains at most $|P'_{i+1}|/\alpha$ points of $P'_1 \cup \cdots \cup P'_i$. Let $\bar{P}$ denote $P - S$.

We show in Claim 5.3.1 that the invariant properties imply that $Q$ contains all the points of $P'_{i+1} \cup \cdots \cup P'_k$. Further, in Claim 5.3.2, we show that $P'_{i+1}$ forms a constant fraction of the points of $\bar{P}$, i.e., $|P'_{i+1}| \geq \frac{\alpha^{O(1)}}{k}|\bar{P}|$. It follows that $|P'_{i+1}| \geq \frac{\alpha^{O(1)}}{k}|\bar{P} \cap Q|$. So, if we knew $\bar{P}$, then we could get a point $c'_{i+1}$ which is a $(1+\alpha/k)$ approximation to $c_{i+1}$ (as the 1-center of the cluster $P'_{i+1}$) by sampling $O((\lambda_{\alpha/k} k/\alpha^{O(1)}))$ points from $\bar{P} \cap Q$, and generating the candidate center set of size $O(2^{(\frac{k}{\alpha}\lambda_{\alpha/k})^{O(1)}} \beta_{\alpha/k})$ as described by the *Random Sampling Procedure* and the *Superset Sampling Lemma*. But of course we do not know $\bar{P}$.

Note that we actually only require a constant factor approximation to $\bar{P} \cap Q$. Amongst the parameter lists in $\mathcal{W}_i$ satisfying the invariant conditions mentioned above, choose a list $(Q, m, k, C, \alpha,$

Sum) for which $|Q|$ is smallest.

Note that condition (b) of the tightness property suggests that when we consider the points in $P$ (and therefore $Q$) in order of distance from the centers in $C'_i$, then the points of $S$ are closer than the points of $\bar{P}$. Therefore, we can eliminate half the remaining points of $Q$ in order of distance from these centers until we get to a factor 2 approximation to $\bar{P} \cap Q$. This is done recursively in Step 3 of the algorithm. In fact, a call to the algorithm **Irred-$k$-clustering** that corresponds to $Q$ being a factor 2 approximation to $\bar{P} \cap Q$ corresponds to a parameter list $(Q, m, k, C, \alpha, \text{Sum})$ in $\mathcal{W}_i$, satisfying the invariant conditions, for which $|Q|$ is smallest. We prove this in Lemma 5.2.

Note that this is similar to the process of eliminating points in order of distance from the approximate center of the larger cluster till we get to a constant fraction of the points in the second cluster in the 2-means clustering algorithm described in Section 4.2.

We now formally prove the above claims.

**Claim 5.3.1** *$P'_{i+1} \cup \cdots \cup P'_k$ is contained in $\bar{P} \cap Q$.*

*Proof.* We already know that $S$ is contained in $P'_1 \cup \cdots \cup P'_i$. Therefore, $P'_{i+1} \cup \cdots \cup P'_k$ is contained in $\bar{P}$. We only require to show that $P'_{i+1} \cup \cdots \cup P'_k \subseteq Q$.

Suppose that the claim is not true. Then there exists a point $x \in P'_{i+1} \cup \cdots \cup P'_k$ such that $x \in P - Q$. This means that $x$ was eliminated in a previous iteration. Let that iteration be $j < i$. Then there exist centers $c'_1, \ldots, c'_j \in C'_i$, such that in iteration $j$, $\mathrm{d}(x, \{c'_1, \ldots, c'_j\}) \leq \mathrm{d}(x, \{c_{i+1}, \ldots, c_k\})$. This implies that $\mathrm{d}(x, C'_i) \leq \mathrm{d}(x, C''_i)$. This contradicts that $x \in P'_{i+1} \cup \cdots \cup P'_k$ by the *closeness property* of the clustering problem.

This proves Claim 5.3.1. ∎

**Claim 5.3.2** $|P'_{i+1}| \geq \frac{\alpha^{O(1)}}{k} |\bar{P}|$.

*Proof.* By the tightness property, we know that there are at most $|P'_{i+1}|/\alpha$ elements of $P'_1 \cup \ldots \cup P'_i$ in $\bar{P}$. Therefore, since $P'_{i+1}, \ldots, P'_k$ are the clusters associated with the centers in $C''_i$ and $P'_{i+1}$ is the largest of these clusters, we have $|\bar{P}| \leq |P'_{i+1}|/\alpha^{O(1)} + |P'_{i+1}| + \ldots + |P'_k| \leq |P'_{i+1}|/\alpha^{O(1)} + k|P'_{i+1}| \leq \frac{k}{\alpha^{O(1)}} |P'_{i+1}|$.

This proves Claim 5.3.2. ∎

Recall that we are considering the parameter list $(Q, m, k, C, \alpha, \text{Sum})$ in $\mathcal{W}_i$, satisfying the invariant conditions, for which $|Q|$ is smallest.

**Lemma 5.2** $|\bar{P} \cap Q| \geq |Q|/2$.

*Proof.* Suppose not, i.e., $|\bar{P} \cap Q| \leq |Q|/2$.

**Claim 5.3.3** *Consider the points in $Q$ sorted in ascending order of the distance from $C$. Let $U$ be the first $|Q|/2$ points in this order. Then $U$ does not contain a point of $\bar{P} \cap Q$.*

*Proof.* Follows from condition (b) of the *Tightness Property* for the clustering problem and the assumption that $|\bar{P} \cap Q| \leq |Q|/2$. ∎

So, if $U$ is as defined in the claim above, then $\bar{P} \cap Q$ is a subset of $Q - U$. Since $P'_{i+1} \cup \cdots \cup P'_k$ is contained in $\bar{P} \cap Q$ (because of Claim 5.3.1 and the fact that $Q$ is in the parameter list which satisfies the invariant for $i$), it follows that $P'_{i+1} \cup \cdots \cup P'_k$ is a subset of $Q - U$. Thus, the parameter list $(Q - U, C, k, m, \alpha, \text{Sum})$ which is formed in Step 3(e) of the algorithm satisfies the invariant for $i$ as well, i.e., it is in $\mathcal{C}_i$. But this violates the fact that $(Q, C, k, m, \alpha, \text{Sum})$ was the parameter list satisfying the invariant for $i$ in $\mathcal{C}_i$ for which $|Q|$ is smallest.

This proves Lemma 5.2. ∎

The Lemma above implies that $|\bar{P} \cap Q| \geq |Q|/2$. Combined with Claim 5.3.2, we get $|P'_{i+1}| \geq \frac{\alpha^{O(1)}|Q|}{4k}$. The superset sampling Lemma combined with the claim above imply that by sampling $O(\lambda_{\alpha/k} k/\alpha^{O(1)})$ points from $Q$ and generating the candidate center set as described by the *Random Sampling Procedure*, $\mathcal{A}$, for the clustering problem, we shall get a point $c'_{i+1}$ such that $f(P'_{i+1}, c'_{i+1}) \leq (1 + \alpha/k) f(P'_{i+1}, c_{i+1})$, where $c_{i+1} \in C''_i$ is the center of $P'_{i+1}$ in the optimal clustering induced by $C'$. This is the case handled by the step 2(b) in the algorithm **Irred-$k$-clustering**. In this case the algorithm is called again with parameters $(Q, m - 1, k, C \cup \{c'_{i+1}\}, \alpha, \text{Sum})$. It is easy to see now that this parameter list satisfies the invariant for $i + 1$. The set of known centers $C'_{i+1}$ for the next iteration is $C'_i \cup \{c'_{i+1}\}$ and the set of unknown centers $C''_{i+1}$ is $C''_i \backslash \{c_{i+1}\}$. Since $f(P'_{i+1}, c'_{i+1}) \leq (1+\alpha/k) f(P'_{i+1}, c_{i+1})$ and the clustering problem satisfies the closeness property, it follows that $\mathsf{OPT}_k(P, C'_{i+1} \cup C''_{i+1}) \leq (1 + \alpha/k)\mathsf{OPT}_k(P, C') \leq (1 + \alpha/k)^{i+1}\mathsf{OPT}_k(P)$. Thus we have shown that the invariant holds for all values of $i$.

As we mentioned earlier, a parameter list $(Q, m, k, C, \alpha, \text{Sum})$ which satisfies the invariant for $i = k$ has the desired centers in $C$.

It is easy to verify that the cost reported by the algorithm $\mathsf{OPT}_k(P, C)$ satisfies

$$\mathsf{OPT}_k(P) \leq \mathsf{OPT}_k(P, C) \leq (1 + \alpha/k)^k \mathsf{OPT}_k(P) \leq (1 + 2\alpha)\mathsf{OPT}_k(P) \leq (1 + \varepsilon/4)\mathsf{OPT}_k(P).$$

This proves the correctness of our algorithm. We just need to calculate the probability with which the algorithm is called with such a parameter list.

Note that the only source of randomness in **Irred-$k$-clustering** is in the Step 2(a). The sampling gives the desired result with constant probability (according to Lemma 4.4). Further each time we execute Step 2, we decrease $m$ by 1. So, in any sequence of successive recursive calls, there can be at most $k$ invocations of Step 2. Now, we have just shown that there is a parameter list in $\mathcal{W}_k$ for which $C$ contains a set of centers close to the optimal clusters. Let us look at the sequence of recursive calls which have resulted in this parameter list. In these sequence of calls, as we mentioned above, there are $k$ invocations of the random sampling. Each of these work correctly with constant probability. Therefore, the probability that we actually see this parameter list during the execution of this algorithm is $\gamma^k$ for some constant $\gamma$.

This completes the proof of Theorem 5.1 ∎

Now we establish the running time of our algorithm.

**Theorem 5.3** *The algorithm* **Irred-$k$-clustering** *when called with parameters* $(P, k, k, \emptyset, \alpha, 0)$ *runs in time* $O(2^{(k/\alpha)^{O(1)}}dn)$*, where* $n = |P|$.

*Proof.* Let $T(n, m)$ be the running time of our algorithm on input $(Q, m, k, C, \alpha, \text{Sum})$ where $n = |Q|$. Then in Step 2(b), we have $u(k, \alpha)$ subsets of the sample, where $u(k, \alpha) = O(2^{(\lambda_{\alpha/k}\frac{k}{\alpha})^{O(1)}})$. Computation of the candidate center set from any set $S'$ in Step 2(b) takes $O(\eta_{\alpha/k} \cdot nd)$ time. Steps 3(a)-(d) take $O(nd)$ time. Therefore we get the recurrence

$$T(n, m) = O(u(k, \alpha) \cdot \beta_{\alpha/k})T(n, m-1) + T(n/2, m) + O(u(k, \alpha) \cdot \eta_{\alpha/k} \cdot nd).$$

Let $\lambda_\alpha = O(1/\alpha^{O(1)})$, $\beta_\alpha = O(2^{(1/\alpha)^{O(1)}})$ and $\eta_\alpha = O(2^{(1/\alpha)^{O(1)}})$. Choose $c = O(2^{(k/\alpha)^\gamma})$ to be large enough, for a suitable constant $\gamma$, such that

$$T(n, m) \leq c \cdot T(n, m-1) + T(n/2, m) + c \cdot nd.$$

We claim that $T(n, m) \leq c^m \cdot 2^{3m^2} \cdot nd$. The proof is by induction. Consider the base cases. $T(n, 0) = knd$ as there are no more centers to be determined and the points only need to be assigned to the closest center. Also, $T(0, m) = 0$ as there are no points and therefore it holds vacuously. For the inductive step, suppose that the claim holds for $T(n', m')\ \forall n', \forall m' < m$ and it holds for $T(n', m')\ \forall n' < n, \forall m'$. Then, we are required to show that

$$c^m \cdot 2^{3m^2} \cdot nd \geq c \cdot c^{m-1} \cdot 2^{3(m-1)^2} \cdot nd + c^m \cdot 2^{3m^2} \cdot \frac{n}{2}d + c \cdot nd.$$

For this, it suffices to show that $2^{3m^2} \geq 2^{3(m-1)^2} + 2^{3m^2-1} + 1$ which clearly holds for $m \geq 1$.

It follows that $T(n,k)$ is $O(2^{(k/\alpha)^{O(1)}}dn)$ when $\lambda_\alpha = O(1/\alpha^{O(1)})$, $\beta_\alpha = O(2^{(1/\alpha)^{O(1)}})$ and $\eta_\alpha = O(2^{(1/\alpha)^{O(1)}})$. ∎

We can now state our main Theorem.

**Theorem 5.4** *For a clustering problem satisfying the* Closeness Property, Tightness Property *and for which there exists a* Random Sampling Procedure, *a $(1+\varepsilon)$-approximate solution for a point set $P$ in $\Re^d$ can be found in time $O(2^{(k/\varepsilon)^{O(1)}}dn)$, with constant probability.*

*Proof.* We can run the algorithm **Irred-$k$-clustering** $c^k$ times for some constant $c$ to ensure that it yields the desired result with constant probability. This still keeps the running time $O(2^{(k/\alpha)^{O(1)}}dn)$. So let us assume this algorithm gives the desired solution with constant probability.

Notice that the running time of our main algorithm in Figure 5.1 is also $O(2^{(k/\alpha)^{O(1)}}dn)$. We just have to show that it is correct.

Let $i$ be the highest index for which $P$ is $(i,\alpha)$-irreducible. So, it follows that

$$\mathsf{OPT}_i(P) \leq (1+\delta k\alpha)\mathsf{OPT}_{i+1}(P) \leq \cdots \leq (1+\delta k\alpha)^{k-i}\mathsf{OPT}_k(P) \leq (1+\varepsilon/4)\mathsf{OPT}_k(P).$$

Further, we know that the algorithm **Irred-$k$-clustering** on input $(P,i,i,\emptyset,\alpha,0)$ yields a set of $i$ centers $C$ for which $\mathsf{OPT}_k(P,C) \leq (1+\varepsilon/4)\mathsf{OPT}_i(P)$. Therefore, we get a solution of cost at most $(1+\varepsilon/4)(1+\varepsilon/4)\mathsf{OPT}_k(P) \leq (1+\varepsilon)\mathsf{OPT}_k(P)$. This proves the Theorem. ∎

We now give applications to various clustering problems. We show that these clustering problems satisfy the tightness property and admit a random sampling procedure as described in the previous section.

For the $k$-means clustering problem, the random sampling property follows from Lemma 3.2 shown by Inaba et al [37], and the tightness property follows from Lemma 4.5. This leads to the following Corollary to Theorem 5.4.

**Corollary 5.5** *Given a point set $P$ of $n$ points in $\Re^d$, a $(1+\varepsilon)$-approximate solution to the $k$-means clustering problem can be found in time $O(2^{(k/\varepsilon)^{O(1)}}dn)$, with constant probability.*

## 5.4 $k$-medians Clustering

As described earlier, the clustering problem $\mathcal{C}(f,k)$ is said to be the $k$-median problem if $f(Q,x) = \sum_{q\in Q} \mathrm{d}(q,x)$. We now exhibit the Random Sampling Procedure and the Tightness Property for this

problem leading to the following Corollary to Theorem 5.4.

**Corollary 5.6** *Given a point set $P$ of $n$ points in $\Re^d$, a $(1+\varepsilon)$-approximate solution to the $k$-medians clustering problem can be found in time $O(2^{(k/\varepsilon)^{O(1)}}dn)$, with constant probability.*

### 5.4.1 Random Sampling Procedure

Badoiu et al. [9] showed that a small random sample can be used to get a close approximation to the optimal 1-median solution. Given a set of points $P$, let AvgMed$(P)$ denote $\frac{\mathsf{OPT}_1(P)}{|P|}$, i.e., the average cost paid by a point towards the optimal 1-median solution.

**Lemma 5.7** *[9] Let $P$ be a set of points in $\Re^d$, and $\varepsilon$ be a constant between 0 and 1. Let $X$ be a random sample of $O(1/\varepsilon^3 \log 1/\varepsilon)$ points from $P$. Then with constant probability, the following two events happen: (i) The flat $span(X)$ contains a point $x$ such that $\mathsf{OPT}_1(P,\{x\}) \leq (1+\varepsilon)\mathsf{OPT}_1(P)$. and (ii) $X$ contains a point $y$ at distance at most 2AvgMed$(P)$ from $x$.*

We now show that if we can upper and lower bound AvgMed$(P)$ up to constant factors, then we can construct a small set of points such that at least one of these is a good approximation to the optimal center for the 1-median problem on $P$.

**Lemma 5.8** *Let $P$ be a set of points in $\Re^d$ and $X$ be a random sample of size $O(1/\varepsilon^3 \log 1/\varepsilon)$ from $P$. Suppose we happen to know numbers $a$ and $b$ such that $a \leq$ AvgMed$(P) \leq b$. Then, we can construct a set $Y$ of $O(2^{(1/\varepsilon)^{O(1)}}\log(b/\varepsilon a))$ points such that with constant probability there is at least one point $z \in X \cup Y$ satisfying $\mathsf{OPT}_1(P,\{z\}) \leq (1+2\varepsilon)\mathsf{OPT}_1(P)$. Further, the time taken to construct $Y$ from $X$ is is $O(2^{(1/\varepsilon)^{O(1)}}d)$.*

*Proof.* Our construction is similar to that of Badoiu et al. [9]. We can assume that the result stated in Lemma 5.7 holds (because this happens with constant probability). Let $x$ and $y$ be as in Lemma 5.7.

We will carefully construct candidate points around the points of $X$ in $span(X)$ in an effort to get within close distance of $x$.

For each point $p \in X$, and each integer $i$ in the range $[\lfloor \log \frac{\varepsilon}{4} a \rfloor, \lceil \log b \rceil]$ we do the following – let $t = 2^i$. Consider the grid $G_p(t)$ of side length $\varepsilon t/(4|X|) = O(t\varepsilon^4 \log(1/\varepsilon))$ in $span(X)$ centered at $p$. We add all the vertices of this grid lying within distance at most $2t$ from $p$ to our candidate set $Y$. This completes the construction of $Y$. It is easy to see that the time taken to construct $Y$ from $X$ is $O(2^{(1/\varepsilon)^{O(1)}}d)$.

We now show the existence of the desired point $z \in X \cup Y$. Consider the following cases:

1. $\mathrm{d}(y,x) \leq \varepsilon \mathrm{AvgMed}(P)$ : Using triangle inequality, we see that

$$f(P,y) \leq f(P,x) + |P|\mathrm{d}(y,x) \leq (1+2\varepsilon)\mathsf{OPT}_1(P).$$

   Therefore $y$ itself is the required point.

2. $\mathrm{d}(y,x) > \varepsilon \mathrm{AvgMed}(P)$ : Consider the value of $i$ such that $2^{i-1} \leq \mathrm{AvgMed}(P) \leq 2^i$ – while constructing $Y$, we must have considered this value of $i$ for all points in $X$. Let $t = 2^i$. Clearly, $t/2 \leq \mathrm{AvgMed}(P) \leq t$.

   Observe that $\mathrm{d}(y,x) \leq 2\mathrm{AvgMed}(P) \leq 2t$. Therefore, in the manner by which we have constructed $G_y(t)$, there must be a point $p \in G_y(t)$ for which $\mathrm{d}(p,x) \leq \varepsilon t/2 \leq \varepsilon \mathrm{AvgMed}(P)$. This implies that

$$f(P,p) \leq f(P,x) + |P|\mathrm{d}(x,p) \leq (1+2\varepsilon)\mathsf{OPT}_1(P).$$

   Hence $p$ is the required point.

This completes the proof of Lemma 5.8. ∎

We now show the existence of the random sampling procedure.

**Theorem 5.9** *Let $P$ be a set of $n$ points in $\Re^d$, and let $\varepsilon$ be a constant, $0 < \varepsilon < 1/12$. There exists an algorithm which randomly samples a set $R$ of $O((\frac{1}{\varepsilon})^{O(1)})$ points from $P$. Using this sample only, it constructs a set of points core$(R)$ such that with constant probability there is a point $x \in$ core$(R)$ satisfying $f(P,x) \leq (1+O(\varepsilon))\mathsf{OPT}_1(P)$. Further, the time taken to construct core$(R)$ from $R$ is $O(2^{(1/\varepsilon)^{O(1)}}d)$.*

*Proof.* Consider the optimal 1-median solution for $P$ – let $c$ be the center in this solution. Let $T$ denote AvgMed$(P)$. Consider the ball $B_1$ of radius $T/\varepsilon^2$ around $c$. Let $P'$ be the points of $P$ contained in $B_1$. It is easy to see that $|P'| \geq (1-\varepsilon^2)n$.

Sample a point $p$ at random from $P$. With constant probability, it lies in $P'$. Randomly sample a set $Q$ of $1/\varepsilon$ points from $P$. Again, with constant probability, these points lie in $P'$. So we assume that these two events happen. Let $v = \sum_{q \in Q} \mathrm{d}(q,p)$. We want to show that $v$ is actually close to AvgMed$(P)$.

Let $B_2$ denote the ball of radius $\varepsilon T$ centered at $p$. One of the following two cases must happen :

- There are at least $2\varepsilon|P'|$ points of $P'$ outside $B_2$ : In this case, with constant probability, the sample $Q$ contains a point outside $B_2$. Therefore, $v \geq \varepsilon T$. Also notice that any two points in $B_1$ are at distance at most $2T/\varepsilon^2$ from each other. So, $v \leq 2T|Q|/\varepsilon^2$. We choose $a = v/\varepsilon$ and $b = \frac{v\varepsilon^2}{2|Q|}$. Notice that $b/a$ is $O(\varepsilon^{O(1)})$. We can now use the Lemma 5.8 to construct the desired core set.

- There are at most $2\varepsilon|P'|$ points of $P'$ outside $B_2$ : Suppose $\mathrm{d}(p,c) \leq 4\varepsilon T$. In this case $f(P,p) \leq (1+O(\varepsilon))\mathsf{OPT}_1(P)$ and we are done. So assume this is not the case. Note that the number of points outside $B_2$ is at most $|P-P'| + 2\varepsilon|P'| \leq \varepsilon^2 n + 2\varepsilon(1-\varepsilon^2)n \leq 3\varepsilon n$. Now suppose we assign all points of $P$ from $c$ to $p$. Let us see the change in cost. The distance the points in $B_2$ have to travel decreases by at least $\mathrm{d}(c,p) - 2\varepsilon R$. The increase in the distance for points outside $B_2$ is at most $\mathrm{d}(c,p)$. So the overall decrease in cost is at least

$$|B_2|(\mathrm{d}(c,p) - 2\varepsilon R) - (n - |B_2|)\mathrm{d}(c,p) > 0$$

  if we use $|B_2| \geq n(1-3\varepsilon)$ and $\mathrm{d}(c,p) \geq 4\varepsilon R$. This yields a contradiction because $c$ is the optimal center. Thus we are done in this case as well.

This proves Theorem 5.9. ∎

Thus we have shown the existence of the random sampling procedure.

### 5.4.2 Tightness Property

We now show the existence of tightness property.

**Lemma 5.10** *The $k$-medians clustering problem, having cost function $f_1(Q,x) = \sum_{q\in Q} d(q,x)$ satisfies the tightness property.*

*Proof.* We need to show the existence of the desired set $S$.

Consider the closest pair of centers between the sets $C'\backslash C'_i$ and $C'_i$ – let these centers be $c_l$ and $c'_r$ respectively. Let $t = \mathrm{d}(c_l, c'_r)$. Let $S$ be the set of points $\mathrm{b}(c'_1, t/4) \cup \cdots \cup \mathrm{b}(c'_i, t/4)$, i.e., the points which are distant at most $t/4$ from $C'_i = \{c'_1, \ldots, c'_i\}$.

Clearly, $S$ is contained in $P'_1 \cup \cdots \cup P'_i$. This shows (a). Also, for any $x \in S, x' \in P - S$, $\mathrm{d}(x, \{c'_1, \ldots, c'_i\}) \leq \mathrm{d}(x', \{c'_1, \ldots, c'_i\})$. This proves (b).

Suppose $P - S$ contains more than $|P_l|/\alpha$ points of $P'_1 \cup \cdots \cup P'_i$. In that case, these points are assigned to centers at distance at least $t/4$. It follows that $\mathsf{OPT}_k(P, C')$ is at least $\frac{t|P_l|}{4\alpha}$. This implies

that $t|P_l| \leq 4\alpha\mathsf{OPT}_k(P, C')$. But then if we assign all the points in $P_l$ to $c'_r$, the cost increases by at most

$$|P_l|t \leq 4\alpha\mathsf{OPT}_k(P, C') \leq 4\alpha(1+\alpha/k)^i\mathsf{OPT}_k(P) \leq 4\alpha(1+\alpha/k)^k\mathsf{OPT}_k(P) \leq 12\alpha\mathsf{OPT}_k(P).$$

But this contradicts the fact that $P$ is $(k, \alpha)$-irreducible for $\delta = 12$. This proves the tightness property. ∎

### 5.4.3 Applications to the 1-median Problem

In this section, we present an algorithm for the 1-median problem. Given a set of $n$ points in $\Re^d$, the algorithm with constant probability produces a solution of cost at most $(1+\varepsilon)$ of the optimal cost for any constant $\varepsilon > 0$. The running time of the algorithm is $O(2^{1/\varepsilon^{O(1)}}d)$. Here, it is assumed that the input is provided in some standard representation (for instance as an array by passing its pointer) and the algorithm does not have to read the entire input but it may determine the output by checking only a subset of the input elements. Here, we require that it is possible to randomly sample a point in constant time.

Our algorithm is based on the following idea presented by Indyk [38].

**Lemma 5.11** *[38] Let $X$ be a set of $n$ points in $\Re^d$. For a point $a \in \Re^d$ and a subset $Q \subseteq X$, define $S_Q(a) = \sum_{x\in Q} d(a, x)$ and $S(a) = S_X(a)$. Let $\varepsilon$ be a constant, $0 \leq \varepsilon \leq 1$. Suppose $a$ and $b$ are two points such that $S(b) > (1+\varepsilon)S(a)$. Then,*

$$Pr\left(\sum_{x\in Q} d(a, x) \geq \sum_{x\in Q} d(b, x)\right) < e^{-\varepsilon^2|Q|/64}.$$

We now show the existence of a fast algorithm for approximating the optimal 1-median solution.

**Theorem 5.12** *Let $P$ be a set of $n$ points in $\Re^d$, and let $\varepsilon$ be a constant, $0 < \varepsilon < 1$. There exists an algorithm which randomly samples a set $R$ of $O((\frac{1}{\varepsilon})^{O(1)})$ points from $P$. Using this sample only, it finds a point $p$ such that $f(P, x) \leq (1+O(\varepsilon))\mathsf{OPT}_1(P)$ with constant probability (independent of $\varepsilon$). The time taken by the algorithm to find such a point $p$ from $R$ is $O(2^{(1/\varepsilon)^{O(1)}}d)$.*

*Proof.* We first randomly sample a set $R_1$ of $O((\frac{1}{\varepsilon})^{O(1)})$ points from $P$ and using Theorem 5.9, construct a set core$(R_1)$ of $O(2^{(1/\varepsilon)^{O(1)}})$ points such that with constant probability, there is a point $x \in$ core$(R_1)$ satisfying $f(P, x) \leq (1+O(\varepsilon))\mathsf{OPT}_1(P)$.

Now we randomly sample a set $R_2$ of $O((1/\varepsilon)^{O(1)})$ points and find the point $p \in \text{core}(R_1)$ for which $S_{R_2}(p) = f_1(R_2, p)$ is minimum. By Lemma 5.11, $p$ is with constant probability a $(1 + O(\varepsilon))$-approximate median of $P$.

Clearly, the time taken by the algorithm is $O(2^{(1/\varepsilon)^{O(1)}} d)$. ∎

Also note that we can boost the success probability to an arbitrarily small constant by selecting a large enough (yet constant) sample $R$.

## 5.5 Discrete $k$-means Clustering

This is the same as the $k$-means problem with the additional constraint that the centers must be chosen from the input point set only.

The tightness property follows from Lemma 4.5. We now exhibit the Random Sampling Procedure for this problem leading to the following Corollary to Theorem 5.4.

**Corollary 5.13** *Given a point set $P$ of $n$ points in $\Re^d$, a $(1+\varepsilon)$-approximate solution to the discrete $k$-means clustering problem can be found in time $O(2^{(k/\varepsilon)^{O(1)}} dn)$, with constant probability.*

### 5.5.1 Random Sampling Procedure

We first show that given a good approximation to the center of the optimal (continuous) 1-means problem, we can get a good approximation to the center of the optimal discrete 1-means problem. We require some additional notations. Let $P$ be a set of $n$ points in $\Re^d$. Let $c$ be the center of the optimal solution to the (continuous) 1-means problem on $P$.

**Lemma 5.14** *Let $\alpha$ be a constant, $0 < \alpha < 1$, and $c'$ be a point in $\Re^d$ such that $\sum_{p\in P} d(p, c')^2 \leq (1 + \alpha) \sum_{p\in P} d(p, c)^2$. Let $x'$ be the point of $P$ closest to $c'$. Then $\mathsf{OPT}_1(P, \{x'\}) \leq (1 + O(\sqrt{\alpha}))\ \mathsf{OPT}_1(P)$.*

*Proof.* Let $x$ be the center of the optimal discrete 1-mean solution, i.e., $\mathsf{OPT}_1(P, \{x\}) = \mathsf{OPT}_1(P)$. Let $T$ be the average cost paid by the points of $P$ in the optimal 1-means solution, i.e., $T = \frac{\sum_{p\in P} \mathrm{d}(p,c)^2}{|P|}$.

Then $\mathsf{OPT}_1(P) = |P|(T + \mathrm{d}(c, x)^2)$ and $\mathsf{OPT}_1(P, \{x'\}) = |P|(T + \mathrm{d}(c, x')^2)$. From the definition of $c'$, we know that $\mathrm{d}(c, c')^2 \leq \alpha T$.

Notice that

$$\mathrm{d}(c,x')\leq \mathrm{d}(c,c')+\mathrm{d}(c',x')\leq \mathrm{d}(c,c')+\mathrm{d}(c',x)\leq 2\mathrm{d}(c,c')+\mathrm{d}(c,x).$$

We know that $f_2(P,x)=|P|(T+\mathrm{d}(c,x)^2)$ and $f_2(P,x')=|P|(T+\mathrm{d}(c,x')^2)$. So

$$\begin{aligned} f_2(P,x')-f_2(P,x) &= |P|(\mathrm{d}(c,x')^2-\mathrm{d}(c,x)^2) \\ &\leq |P|\left((2\mathrm{d}(c,c')+\mathrm{d}(c,x))^2-\mathrm{d}(c,x)^2\right) \\ &\leq 4|P|\left(\mathrm{d}(c,c')^2+\mathrm{d}(c,c')\mathrm{d}(c,x)\right) \\ &\leq 4|P|\left(\alpha T+\sqrt{\alpha T}\mathrm{d}(c,x)\right) \\ &\leq 4|P|\left(\alpha T+\sqrt{\alpha}(T+\mathrm{d}(c,x)^2)\right) \\ &\leq O(\sqrt{\alpha})\mathsf{OPT}_1(P). \end{aligned}$$

∎

We now show the existence of the random sampling procedure.

**Theorem 5.15** *Let $\alpha$ be a constant, $0<\alpha<1$. There exists an algorithm which randomly samples a set $R$ of $O\left(\frac{1}{\alpha}\right)$ points from $P$. Using this sample, it finds a singleton set core$(R)$ such that with constant probability the point $x\in$ core$(R)$ satisfies $f(P,x)\leq(1+O(\sqrt{\alpha}))\mathsf{OPT}_1(P)$. Further, the time taken to construct core$(R)$ from $R$ is $O((\frac{1}{\alpha}+n)d)$.*

*Proof.* Using Lemma 3.2, we can get a point $c'$ such that $\sum_{p\in P}\mathrm{d}(p,c')^2\leq(1+\alpha)\sum_{p\in P}\mathrm{d}(p,c)^2$. As mentioned in the Lemma, we do this by by taking the centroid of a random sample of $O(1/\alpha)$ points of $P$. This takes time $O(\frac{1}{\alpha}\cdot d)$.

The rest follows from the previous Lemma. ∎

## 5.6 Weighted Clustering

In this section, we consider the situation in which each point $p$ has an integral weight $w_p$ associated with it. Let $W$ be the total sum of all the weights and $n$ be the number of distinct points.

Let us look at the cost function for some weighted clustering problems.

- Weighted $k$-median : $f_1(Q,x)=\sum_{q\in Q}w_q\cdot\mathrm{d}(q,x)$.
- Weighted $k$-means : $f_2(Q,x)=\sum_{q\in Q}w_q\cdot\mathrm{d}(q,x)^2$.

For a set of points $S$, let $W_S = \sum_{s \in S} w_s$ be the weighted sum of the points in $S$.

An important observation is that the solution to the above weighted problems is the same as the solution to the corresponding unweighted version of the problems where a point $p$ with weight $w$ is replaced with $w$ points (of unit weight). The algorithm and proofs of the unweighted version extend with little or no change by virtue of this observation. We outline these extensions below.

The *Irreducibility* definition and the *Closeness Property* extend to the weighted version without any change. The *Random Sampling Procedure* is modified as follows.

> ***Weighted Random Sampling Procedure*** **:** There exists a procedure $\mathcal{A}$ that takes a set of weighted points $Q \in \Re^d$ and a parameter $\alpha$ as input. $\mathcal{A}$ first randomly samples a small set $R$ of constant, $\lambda_\alpha$, points from $Q$. Starting from $R$, $\mathcal{A}$ produces another set of points, which we call core$(R)$, of constant size, $\beta_\alpha$. $\mathcal{A}$ satisfies the condition that with constant probability there is at least one point $c \in$ core$(R)$ such that $\mathsf{OPT}_1(Q, \{c\}) \leq (1+\alpha)\mathsf{OPT}_1(Q)$. Further the time taken by $\mathcal{A}$ to produce core$(R)$ from $R$ is at most $O(\eta_\alpha \cdot dn)$, where $n$ is the size (number of distinct points) of $Q$ and $\eta_\alpha$ is a constant.
>
> Note that, as in the unweighted case, when we actually apply the *Weighted Random Sampling Procedure*, the set $Q$ will not be explicitly known - instead a superset $P \supseteq Q$ will be given. We will sample a slightly larger set of points from $P$ and then isolate a subset of points of $Q$. Our sampling/isolation procedure must additionally satisfy the condition that any point set obtained from the underlying set $Q$ after sampling/isolation must be equiprobable amongst all the point sets of the same size in $Q_m$ (obtained with replacement) where $Q_m$ is the multiset obtained by replacing each weighted point $p \in Q$ by $w_p$ points.

We extend the *Superset Sampling Lemma* to the weighted problem, provided that the total weight of the subset $Q$ is a constant fraction of the total weight of $P$. To do this we perform weighted sampling, i.e., while sampling every point, a point $p$ is picked with probability $w_p/W$. We then isolate a subset of $Q$ by enumerating all subsets of the sample.

Let $P$ be a set of points and $Q$ be a subset of $P$ such that $W_Q \geq \theta W_P$, where $\theta$ is a constant between 0 and 1. Suppose that we require a sample of $\lambda_\alpha$ points from $Q$ to generate the candidate center set that contains a $(1+\alpha)$-approximation to the 1-center. Suppose we take a weighted sample $S$ of size $O(\frac{4\lambda_\alpha}{\theta})$ from $P$. Now we consider all possible subsets of size $\lambda_\alpha$ of $S$. For each of these subsets $S'$, we can generate a candidate centre set for the 1-center for the clustering problem using the *Weighted Random Sampling Procedure*, $\mathcal{A}$, and consider each one of these as a potential 1-

center for the clustering problem instance on $Q$. The following Lemma states that one of these subsets must give a close enough approximation to the optimal 1-center solution for $Q$.

**Lemma 5.16** *(Weighted Superset Sampling Lemma) Let $\mathcal{CEN}(S')$ be the centre set generated using the* Weighted Random Sampling Procedure *on sampled subset $S'$. Then, the following event happens with constant probability*

$$\min_{c' \in \mathcal{CEN}(S'): S' \subset S, |S'| = \lambda_\alpha} f(Q, c') \leq (1+\alpha)\mathsf{OPT}_1(Q).$$

*Proof.* With constant probability, $S$ contains at least $\lambda_\alpha$ points from $Q$, the required sample size. Moreover this set $S'$ of $\lambda_\alpha$ points is equiprobable amongst all the point sets of the same size in $Q_m$ (obtained with replacement) where $Q_m$ is the corresponding multiset obtained by replacing a weighted point $p \in Q$ with $w_p$ points. The rest follows from the *Weighted Random Sampling Procedure* for the clustering problem. ∎

For the *Tightness Property*, the size of the set is substituted with the weight of the set. We restate this property below.

> ***Weighted Tightness Property*:** Let $P$ be a set of points which is $(k, \alpha)$-irreducible for some constant $\alpha$. Consider an optimal solution to $\mathcal{C}(f, k)$ on $P$ – let $C = \{c_1, \ldots, c_k\}$ be the centers in this solution. Suppose we have a set of $i$ points $C'_i = \{c'_1, \ldots, c'_i\}$, such that $\mathsf{OPT}_k(P, C') \leq (1+\alpha/k)^i \mathsf{OPT}_k(P)$, where $C' = \{c'_1, \ldots, c'_i, c_{i+1}, \ldots, c_k\}$. Let $P'_1, \ldots, P'_k$ be the partitioning of $P$ if we choose $C'$ as the set of centers (in other words this is the Voronoi partitioning of $P$ with respect to $C'$). We assume w.l.o.g. that $P'_{i+1}$ be the largest cluster amongst $P'_{i+1}, \ldots, P'_k$. Then there exists a set of points $S$ such that the following conditions hold :
>
> (a) $S$ is contained in $P'_1 \cup \ldots \cup P'_i$.
>
> (b) Let $x \in S, x' \in P - S$. Then, $\mathrm{d}(x, \{c'_1, \ldots, c'_i\}) \leq \mathrm{d}(x', \{c'_1, \ldots, c'_i\})$.
>
> (c) $P - S$ contains at most $\frac{W_{P'_{i+1}}}{\alpha^{O(1)}}$ points of $P'_1 \cup \ldots \cup P'_i$.

It is important to note here that given a *Random Sampling Procedure* for an unweighted clustering problem, the corresponding *Weighted Random Sampling Procedure* for the weighted version of the problem can be simply obtained by performing weighted sampling as described above. Similarly, it is easy to see that the *Weighted Superset Sampling Lemma* and the *Weighted Tightness Prop-*

**Algorithm Weighted-Irred-$k$-clustering**$(Q, m, k, C, \alpha, \text{Sum})$
**Inputs :** $Q$: Remaining point set, $m$: number of cluster centers yet to be found, $k$: total number of clusters,
$C$: set of $k - m$ cluster centers found so far,
$\alpha$: approximation factor, $Sum$: the cost of assigning points in $P - Q$ to the centers in $C$
**Output :** The clustering of the points in $Q$ in $k$ clusters.

1. If $m = 0$
   Assign the points in $Q$ to the nearest centers in $C$.
   Sum = Sum + The clustering cost of $Q$.
   Return the clustering.
2. (a) Sample (weighted) a set $S$ of size $O(\lambda_{\alpha/k} k/\alpha^{O(1)})$ from $Q$.
   (b) For each subset $S'$ of $S$ of size $\lambda_{\alpha/k}$ do
   Compute the candidate centre set from $S'$.
   For each center, c, in the candidate center set, obtain the clustering **Weighted-Irred-$k$-clustering**$(Q, m-1, k, C \cup \{c\}, \alpha, \text{Sum})$.
3. (a) Consider the points in $Q$ in ascending order of distance from $C$.
   (b) Let $U$ be the first $j$ points in this order such that $W_U \geq W_Q/2 > W_U - w_{p_j}$, where $p_j$ is the $j^{th}$ point in this sequence
   (c) Assign the points in $U$ to the nearest centers in $C$.
   (d) Sum = Sum + The clustering cost of $U$.
   (e) Compute the clustering **Weighted-Irred-$k$-clustering**$(Q - U, m, k, C, \alpha, \text{Sum})$.
4. Return the clustering which has minimum cost.

Figure 5.3: The weighted irreducible $k$-clustering algorithm

*erty* translate into the *Superset Sampling Lemma* and the *Tightness Property* for the corresponding unweighted version of the problem.

### 5.6.1 The Weighted Algorithm

The algorithm extends to the weighted version as follows. The algorithm *$k$-clustering* remains unchanged. The algorithm *Irred-$k$-clustering* is modified as shown in Figure 5.3. We call the weighted version of this algorithm, the **Weighted-Irred-$k$-clustering** algorithm.

Step 2 is modified to perform weighted random sampling. Based on the *Weighted Superset Sampling Lemma*, now sampling can be performed for a set only if the remaining points have at most double the weight (instead of double the number of points). Therefore, in step $3(b)$, we only

eliminate points constituting half the remaining weight (instead of half the remaining points). We assign these points to the nearest centers in $C$ and recursively compute *Irred-$k$-clustering* on the remaining set.

### 5.6.2 Analysis and Proof of Correctness

We will show that the weighted algorithm mimics the steps performed by the unweighted algorithm on the unweighted version of the problem, with the exception of certain optimizations that lead to improved running time.

We now prove correctness of the weighted algorithm.

**Theorem 5.17** *Suppose a weighted point set $P$ is $(k, \alpha)$-irreducible. Then the algorithm* **Weighted-Irred-$k$-clustering**$(P, k, k, \emptyset, \alpha, 0)$ *returns a solution to the clustering problem $\mathcal{C}(f, k)$ on input $P$ of cost at most $(1 + \alpha)\mathsf{OPT}_k(P)$ with probability $\gamma^k$, where $\gamma$ is a constant.*

*Proof.* Note that the modifications in Step 2, namely weighted sampling for application of the *Weighted Superset Sampling Lemma* and the *Weighted Random Sampling Procedure* correspond to unweighted sampling for application of the *Superset Sampling Lemma* and the *Random Sampling Procedure* on the unweighted problem. Therefore the weighted algorithm mimics this step of the unweighted algorithm exactly.

In Step 3, as the *Weighted Tightness Property* implies *Tightness Property* in the corresponding unweighted problem, we can remove half the remaining weight in order to get a factor 2 approximation to the set $\bar{P} \cap Q$ as was done in the unweighted version of the problem.

However, a careful look at Step $3(b)$ seems to suggest that we may be removing extra points in the corresponding unweighted algorithm since we are removing all the points that have the same co-ordinates as the point that divides $Q$ into two equal sized partitions in order of distances from the known centers. We prove below that we are justified in removing these extra points, in the corresponding unweighted version of the problem.

Let $X_p$ denote the set of (multiset) points in the unweighted problem corresponding to the point $p$ in the weighted problem. Note that $w_p = |X_p|$. Consider $X_{p_j}$ (of Step $3(b)$) in the unweighted clustering problem. Then by Observation 4.2, either $X_{p_j} \subseteq S$ or $X_{p_j} \cap S = \phi$. If $X_{p_j} \subseteq S$, then $Q$ is already a factor 2 approximation of $\bar{P} \cap Q$, since $W_U - w_{p_j} < W_Q/2$. Hence, the next center is discovered using random sampling in step 2 of the algorithm. Otherwise, $X_{p_j} \cap S = \phi$. In this case, clearly by eliminating the whole of $X_{p_j}$ in step $3(b)$ of the algorithm, we do not remove any point of $\bar{P} \cap Q$.

Therefore the correctness of the weighted algorithm on a weighted point set follows from the correctness of the corresponding unweighted algorithm when run on the corresponding unweighted instance of the problem. ∎

We now establish the running time of the weighted algorithm.

**Theorem 5.18** *The algorithm* **Weighted-Irred-$k$-clustering** *when called with parameters* $(P, k, k, \emptyset, \alpha, 0)$ *runs in time* $O(2^{(k/\alpha)^{O(1)}} dn \log^k W)$*, where* $n =$ *number of distinct points in* $P$ *and* $W =$ *total weight of the point set* $P$*.*

*Proof.* Let $T(n, m, W)$ be the running time of the weighted algorithm on input $(Q, m, k, C, \alpha,$ Sum$)$ where $n =$ number of distinct points in $Q$ and $W =$ total weight of the point set $Q$. Note that we can perform the weighted random sampling in Step 2(a), in time $O(\lambda_{\alpha/k} \cdot \frac{k}{\alpha^{O(1)}} \cdot n)$. This can be done by computing the cumulative weights of the weighted points, $Q$, in an array of size $n$, then generating a random number in the range 1 to $W_Q$ and picking the corresponding point based on the cumulative weights array. Then in Step 2(b), we have $u(k, \alpha)$ subsets of the sample, where $u(k, \alpha) = O(2^{(\lambda_{\alpha/k} \frac{k}{\alpha})^{O(1)}})$. Computation of the candidate center set from any set $S'$ in Step 2(b) takes $O(\eta_{\alpha/k} \cdot nd)$ time. Steps 3(a)-(d) take $O(nd)$ time. Also note that in step 3(b), at least one distinct point is removed. Therefore we get the recurrence

$$T(n, m, W) = O(u(k, \alpha) \cdot \beta_{\alpha/k}) T(n, m-1, W) + T(n-1, m, W/2) + O(u(k, \alpha) \cdot \eta_{\alpha/k} \cdot nd).$$

Let $\lambda_\alpha = O(1/\alpha^{O(1)})$, $\beta_\alpha = O(2^{(1/\alpha)^{O(1)}})$ and $\eta_\alpha = O(2^{(1/\alpha)^{O(1)}})$. Choose $c = O(2^{(k/\alpha)^\gamma})$ to be large enough, for a suitable constant $\gamma$, such that

$$T(n, m, W) \leq c \cdot T(n, m-1, W) + T(n-1, m, W/2) + c \cdot nd.$$

We claim that $T(n, m, W) \leq c^m \cdot 2^{3m^2} \cdot nd \cdot \log^m W$. The proof is by induction. We show the inductive step here. Suppose that the claim holds for $T(n', m', W')\ \forall n' < n, \forall m', \forall W'$, it holds for $T(n', m', W')\ \forall n', \forall m' < m, \forall W'$ and it holds for $T(n', m', W')\ \forall n', \forall m', \forall W' < W$. Then, we are required to show that

$$c^m \cdot 2^{3m^2} \cdot nd \cdot \log^m W \geq c \cdot c^{m-1} \cdot 2^{3(m-1)^2} \cdot nd \cdot \log^{m-1} W + c^m \cdot 2^{3m^2} \cdot (n-1)d \cdot (\log W - 1)^m + c \cdot nd.$$

For this, it suffices to show that

$$2^{3m^2}\log^m W \geq 2^{3(m-1)^2}\log^{m-1} W + 2^{3m^2-1}(\log W - 1)^m + 1.$$

We know that

$$a^k \geq a^{k-1} + (a-1)^k$$

(follows from the identity $a^k - b^k = (a-b)(a^{k-1} + a^{k-2}b + ... + b^{k-1})$ by setting $b = a-1$). Therefore, we get that

$$\begin{aligned} & 2^{3m^2}\log^m W \geq 2^{3m^2}\log^{m-1}(W) + 2^{3m^2}(\log W - 1)^m \\ \geq \quad & 2^{3(m-1)^2}\log^{m-1} W + 2^{3m^2-1}(\log W - 1)^m + 1 \qquad \text{for } m \geq 1 \end{aligned}$$

It follows that $T(n,k,W)$ is $O(2^{(k/\alpha)^{O(1)}} dn\log^k W)$ when $\lambda_\alpha = O(1/\alpha^{O(1)})$, $\beta_\alpha = O(2^{(1/\alpha)^{O(1)}})$ and $\eta_\alpha = O(2^{(1/\alpha)^{O(1)}})$. ∎

Using these theorems, we get the final result for the weighted problem.

**Theorem 5.19** *For a clustering problem satisfying the* Closeness Property, Weighted Tightness Property *and for which there exists a* Weighted Random Sampling Procedure, *a $(1+\varepsilon)$-approximate solution for a weighted point set $P$ in $\Re^d$ can be found in time $O(2^{(k/\alpha)^{O(1)}} dn\log^k W)$, with constant probability.*

*Proof.* The proof is along the same lines as Theorem 5.4 based on Theorems 5.17 and 5.18. ∎

The following corollary follows from our arguments for the extensions to the general properties for weighted clustering problems.

**Corollary 5.20** *Given a point set $P$ of $n$ points in $\Re^d$ with total weight $W$, $(1+\varepsilon)$-approximate solutions to the weighted $k$-means clustering, weighted $k$-medians clustering and the weighted discrete $k$-means clustering problems can be found in time $O(2^{(k/\alpha)^{O(1)}} dn\log^k W)$, with constant probability.*

## 5.7 Open Problems

It remains open whether or not the discrete $k$-medians clustering problem belongs to this class of clustering problems. In particular, the existence of a *Random Sampling procedure* for the discrete

$k$-medians clustering problem is not yet known.

We illustrate below by means of an example, that the strategy used in the discrete $k$-means clustering, of first finding an approximate (continuous) center and then selecting the closest discrete point to this approximate center does not work for the case of the discrete $k$-medians clustering problem. More precisely, we show that for a point set $P$, the discrete point closest to the non-discrete 1-median may have cost arbitrary larger than the cost of the optimal discrete 1-median.

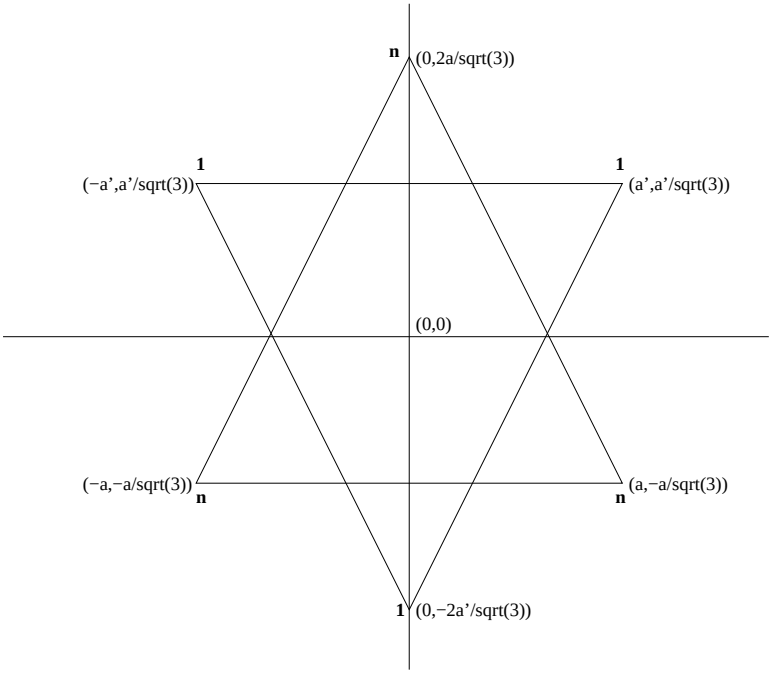


Figure 5.4: Bad example for discrete 1-median clustering

Consider a set $P_1$ consisting of $3n$ points placed on the vertices of an equilateral triangle with $n$ points on each vertex. The co-ordinates of the vertices of the triangle are $(0, 2a/\sqrt{3})$, $(a, -a/\sqrt{3})$ and $(-a, -a/\sqrt{3})$. Consider another set $P_2$ consisting of 3 points placed on the vertices of another equilateral triangle with 1 point on each vertex. The co-ordinates of this triangle are $(0, -2a'/\sqrt{3})$, $(a', a'/\sqrt{3})$ and $(-a', a'/\sqrt{3})$ (see figure 5.4). The optimal 1-median for both $P_1$ and $P_2$ happens to be the origin $(0, 0)$. Therefore the optimal 1-median for $P = P_1 \cup P_2$ is also the origin. Let $a' = a - \delta$, where $\delta$ is an infinitely small number. For ease of computation, we will consider $a' = a$. The reason for having a different value of $a'$ is only to ensure that the points of $P_2$ are

closer to the origin than the points of $P_1$. Now, the optimal (non-discrete) 1-median cost of $P$ is $2\sqrt{3}a(n+1)$. The cost of the 1-median solution when the center is at a vertex corresponding to $P_1$ is $4an + 8a/\sqrt{3}$ and the cost of the 1-median solution when the center is at a vertex corresponding to $P_2$ is $4a + 8an/\sqrt{3}$. Note that the discrete points of $P_2$ are closer to the optimal (non-discrete) 1-median than the discrete points of $P_1$. Therefore if we pick the discrete point closest to the non-discrete 1-median, we can get a solution that has cost arbitrarily larger than the cost of the optimal discrete 1-median cost.

Note that the optimal 1-mean of the points set also happens to be the origin. However, the cost of the 1-means solution when we pick any center from $P_1$ is $4(2n+1)a^2 + 4(a-\delta)^2$ whereas for a center from $P_2$ it is $4a^2n + 4(n+2)(a-\delta)^2$. Thus, the cost is always lower for a point that is closer to the optimal 1-mean of the point set.

Thus, the strategy used in approximating the discrete 1-mean center, of approximating the (non-discrete) 1-center and then selecting the discrete point closest to it, does not work for the 1-median problem.

In a recent development, Ke Chen [19] showed that we can obtain small coresets for $k$-median clustering in metric spaces as well as in Euclidean spaces. Specifically, in $\Re^d$, the coresets are of size with only polynomial dependence in $d$. Ke Chen shows that combining these coresets with our result leads to a $(1+\varepsilon)$-approximation algorithm for $k$-median clustering in $\Re^d$, with running time $O(ndk + 2^{(k/\varepsilon)^{O(1)}} d^2 n^\sigma)$, for any $\sigma > 0$. An interesting open problem is whether there exist coresets for the $k$-median or $k$-means clustering problems of size independent of $n$ and having only polynomial dependence in $d$.

Another interesting open problem is to find a PTAS for the $k$-means clustering problem, even for fixed dimensions.

# Chapter 6

# $K$-median Problem with Priorities in Metric Spaces

In the *priority $k$-median* problem, we are given a set of demands, $\mathcal{D}$ and a set of facilities $\mathcal{F}$ in a metric space. Each demand has a weight $d_j$ associated with it, denoting the quantity of demand to be assigned to an open facility. There are $m$ *types* of demands, with the type indicating the priority of the demand. Thus $\mathcal{D}$ is the disjoint union of $\mathcal{D}_1, \ldots, \mathcal{D}_m$, where we say that $\mathcal{D}_k$ are demands of *type* $k$. Similarly, there are $m$ *types* of facilities, i.e., $\mathcal{F}$ is a disjoint union of $\mathcal{F}_1, \ldots, \mathcal{F}_m$, where we say that $\mathcal{F}_k$ are facilities of *type* $k$. The *type* of a facility specifies its capability in serving the demands – a facility of *type* $k$ can serve demands of *type* $\geq k$, i.e., a demand of *type* $k$ can be served by a facility of *type* $\leq k$. Let $c_{ij}$ denote distance between $i$ and $j$ where $i, j$ can be demands or facilities. A feasible solution opens a set of facilities $F$, and assigns each demand to an open facility. We are given bounds $k_r$ on the number of facilities that can be opened from $\mathcal{F}_r$. As mentioned above, a demand $j$ can only be assigned to an open facility of its type or lower. Let $i(j)$ denote the facility that a demand $j$ is assigned to. Then the cost of the solution is defined as $\sum_{j \in \mathcal{D}} d_j \cdot c_{ji(j)}$. The goal of the *priority $k$-median* problem is to obtain a solution of minimum cost.

A closely related problem is the *facility location problem* where we wish to locate facilities to service a set of demands. This problem has been widely studied in computer science and operations research communities [52, 51]. The tradeoff involved in such problems is the following – we would like to spend as little in opening new facilities as possible, but the clients should not be located too far from the nearest facility. The *facility location problem* balances the two costs by minimizing (weighted) average of the cost of opening facilities and the distances demands have to travel to the

nearest open facility. More concretely, an instance of the facility location problem consists of a set $\mathcal{D}$ of demand points and a set $\mathcal{F}$ of potential facility points. Each facility in $\mathcal{F}$ has an opening cost. We are also given the distance between each demand and facility points. The facility location problem seeks to open some facilities in $\mathcal{F}$ so that the sum of the average distance traveled by a demand in $\mathcal{D}$ to the nearest open facility and the total opening costs of facilities opened is minimized. The variant of the *facility location* problem where the facilities and demands have priorities has already been studied before. Shmoys et al. [59] presented a $6$-approximation algorithm for this problem.

In this chapter, we present a natural integer program for the *priority $k$-median* problem. We show that this integer program has an unbounded integrality gap when there are four priorities. We then give a polynomial time constant factor approximation algorithm for the case when there are only two priorities. Our techniques are based on rounding the natural linear programming relaxation for this problem. Our algorithm involves deeply analyzing the structure of the fractional solution and simplifying it through a sequence of carefully formulated steps. We then formulate another linear program for the simplified problem and show that the solution to this LP must be half-integral. We finally round the half-integral solution to an integral solution.

In Section 6.1 we present the natural integer program for the *priority $k$-median* problem and also show the integrality gap for the case of $4$ priorities. In Section 6.2, we present the constant factor approximation algorithm for the case of $2$ priorities.

## 6.1 Integer Programming Formulation

For a demand $j$, let $type(j) = r$ if $j \in \mathcal{D}_r$.

Fix an instance $\mathcal{I}$ of the problem as described above. We give a natural integer programming formulation for this problem.

$$\min \quad \sum_{j \in \mathcal{D}, i \in \mathcal{F}} d_j \cdot c_{ij} \cdot x_{ij} \tag{6.1}$$

$$\sum_{i \in \mathcal{F}_r} y_i \leq k_r \quad \text{for } r = 1, \ldots, m \tag{6.2}$$

$$\sum_{i \in \mathcal{F}_1 \cup \ldots \cup \mathcal{F}_{type(j)}} x_{ij} = 1 \quad \text{for all demands } j \tag{6.3}$$

$$x_{ij} \leq y_i \quad \text{for all demands } j \text{ and facilities } i \tag{6.4}$$

$$x_{ij} \in \{0, 1\} \quad \text{for all demands } j \text{ and facilities } i \tag{6.5}$$

$$y_i \in \{0, 1\} \quad \text{for all facilities } i \tag{6.6}$$

The integer program can be relaxed to a linear program by relaxing the constraints (6.5) and (6.6) to

$$x_{ij} \geq 0 \quad \text{for all demands } j \text{ and facilities } i \tag{6.7}$$

$$y_i \geq 0 \quad \text{for all facilities } i \tag{6.8}$$

We solve this relaxed linear program and let $x^*, y^*$ be an optimal solution to the linear program. Let $OPT$ denote the cost of this solution.

### 6.1.1 Integrality Gap

We show that the integrality gap of the above relaxation is unbounded when there are four priorities.

Consider the example in which the set of demands is $\mathcal{D} = \{j_2, j_2', j_3, j_3', j_4, j_4'\}$ where $j_2, j_2' \in \mathcal{D}_2$, $j_3, j_3' \in \mathcal{D}_3$ and $j_4, j_4' \in \mathcal{D}_4$. and the set of facilities is $\mathcal{F} = \{i_1, i_1', i_2, i_2', i_3, i_3', i_4, i_4'\}$ where $i_1, i_1' \in \mathcal{F}_1$, $i_2, i_2' \in \mathcal{F}_2$, $i_3, i_3' \in \mathcal{F}_3$ and $i_4, i_4' \in \mathcal{F}_4$. Let $k_1 = k_2 = k_3 = k_4 = 1$ so that at most one facility of each type can be opened. The points $\{i_1, i_2, i_3, i_4, j_2, j_3, j_4\}$ are very far away from $\{i_1', i_2', i_3', i_4', j_2', j_3', j_4'\}$ and so we refer to them as two scenarios.

The distances between these locations are as follows (see Figure 6.1 for an illustration) :

$c_{st} = 1$ for $s, t \in \{i_1, i_4, j_4\}$ and $s \neq t$

$c_{st} = 1$ for $s, t \in \{i_2, i_3, j_3\}$ and $s \neq t$

$c_{st} = d$ for $s = j_2$ and $t \in \{i_1, i_2, i_3, i_4, j_3, j_4\}$

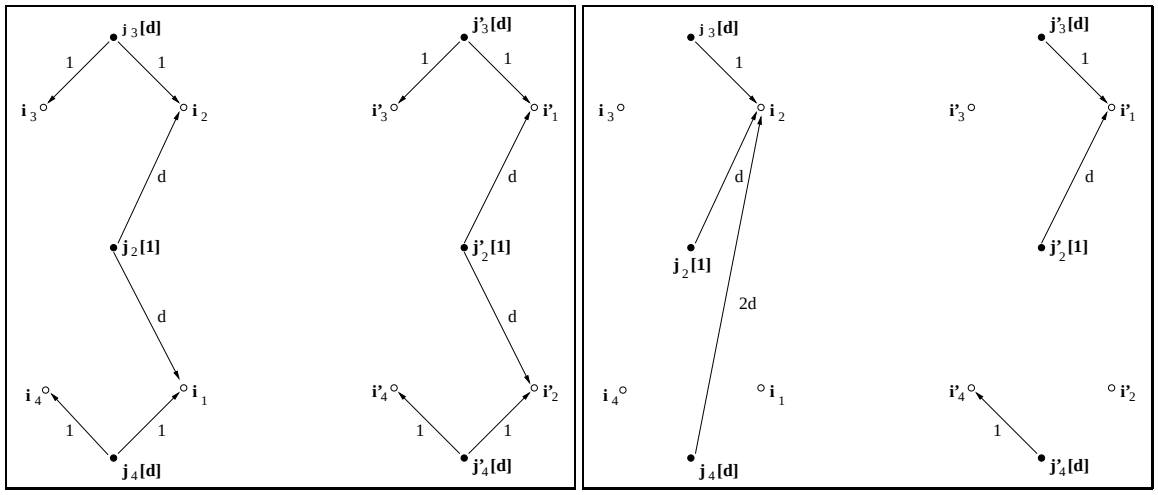


Figure 6.1: Optimal fractional and integral solutions

$c_{st} = 2d$ for $s \in \{i_1, i_4, j_4\}$ and $t \in \{i_2, i_3, j_3\}$

$c_{st} = 1$ for $s, t \in \{i'_1, i'_3, j'_3\}$ and $s \neq t$

$c_{st} = 1$ for $s, t \in \{i'_2, i'_4, j'_4\}$ and $s \neq t$

$c_{st} = d$ for $s = j'_2$ and $t \in \{i'_1, i'_2, i'_3, i'_4, j'_3, j'_4\}$

$c_{st} = 2d$ for $s \in \{i'_1, i'_3, j'_3\}$ and $t \in \{i'_2, i'_4, j'_4\}$

$c_{st} = D$ for $s \in \{i_1, i_2, i_3, i_4, j_2, j_3, j_4\}$ and $t \in \{i'_1, i'_2, i'_3, i'_4, j'_2, j'_3, j'_4\}$

The demands are as follows.

$d_s = 1$ for $s \in \{j_2, j'_2\}$

$d_s = d$ for $s \in \{j_3, j'_3, j_4, j'_4\}$

Let $D >> d$ so that assignments from demands to facilities across the scenarios are discouraged.

The optimal fractional solutions are illustrated in Figure 6.1. In this optimal solution, all the facilities are half-open and each demand is assigned to 2 facilities. Therefore the cost of the optimal fractional solution is $6d$.

Now in the optimal integral solution, note that $j_2$ can only be assigned to $i_1$ or $i_2$. Therefore one of these facilities must be open. Similarly, $j'_2$ can only be assigned to $i'_1$ or $i'_2$. Therefore one of these facilities must also be open. But since $k_1 = k_2 = 1$, we must open a $type(1)$ facility in one scenario and a $type(2)$ facility in the other scenario. Suppose we open the facilities $i'_1$ and $i_2$. Now since $k_4 = 1$ we can only open one of the facilities $i_4$ or $i'_4$. Therefore only one of $j_4$ or $j'_4$ can be assigned to a facility close to it and the other demand will have to be assigned to a facility that is very far from it.

A similar problem is encountered if we open the facilities $i_1$ and $i'_2$. In this case, one of the two demands $j_3$ and $j'_3$ has to be assigned to a facility that is very far from it.

An optimal integral solution is illustrated in Figure 6.1. Note that it does not help to open a $type(3)$ facility. Therefore the cost of the integral solution is $2d^2 + 5d$.

These observations lead to the following result for the LP relaxation of the natural integer programming formulation specified in Section 6.1.

**Theorem 6.1** *For the* priority $k$-median *problem, the LP relaxation of its natural integer programming formulation has an unbounded integrality gap (even in terms of the input number of demands/facilities) when there are four priorities.*

## 6.2 Two Priorities

In this section, we restrict our attention to the case of two priorities, i.e., demands $j \in \mathcal{D}_1 \cup \mathcal{D}_2$ and facilities $i \in \mathcal{F}_1 \cup \mathcal{F}_2$. A demand in $\mathcal{D}_1$ can only use facilities in $\mathcal{F}_1$ whereas a demand in $\mathcal{D}_2$ can use facilities in $\mathcal{F}_1 \cup \mathcal{F}_2$.

We start by solving the relaxed LP for the natural integer program of this problem. We then simplify the problem structure by reducing the number of demands and facilities. First we consolidate demands wherever it is feasible for a demand to use the facilities of another demand. This is done in a manner so that an integral solution to the modified instance with fewer demands results in an integral solution for the original problem instance with at most a constant factor increase in the approximation. We then define a hierarchical structure on the demands called *scenarios*. This is followed by careful reassignment of the demands so that they use the facilities which are either associated with their own scenario or the scenario associated with one other demand. We then show that the assignments can be modified further so that the solution satisfies a nice property (*Structure Property*) that greatly limits the number of open facilities. We then formulate a modified LP for

this nicer instance for which the modified assignments form a feasible solution. We argue that the solution to this modified LP is half-integral. Finally we show that this half-integral solution can be modified to an integral solution by suffering at most a constant factor loss in our approximation.

### 6.2.1 Consolidating demands

For a demand $j$, let $C_j^*$ denote the cost of shipping one unit of demand from $j$ in the optimal LP solution. In other words, $C_j^* = \sum_i x_{ij}^* c_{ij}$. Note that $OPT = \sum_j d_j \cdot C_j^*$. In this step, we consolidate demands at nearby locations by moving demands from one location to another. This step consists of two further substeps.

- *Substep 1 :* We reassign demands to each location $j$ as follows. Initially, we set $d'_j = d_j$ for all locations $j$. Consider the locations in ascending order of $C_j^*$ values. When we consider a location $j$, we check if there is another location $j'$ which has been considered already and which satisfies the conditions : $d'_{j'} > 0$ and $c_{jj'} \leq 32C_j^*$ and $type(j') = type(j)$. If there is such a location $j'$, then the entire demand of $j$ is transferred to $j'$, i.e., set $d'_{j'}$ to $d'_{j'} + d'_j$, and $d'_j$ to $0$.

- *Substep 2 :* We reassign demands to each location $j$ as follows. Initially, we set $d''_j = d'_j$ for all locations $j$. We consider all $type(2)$ demands. When we consider a location $j \in \mathcal{D}_2$, we check if there is another location $j' \in \mathcal{D}_1$ which satisfies the conditions : $d''_{j'} > 0$ and $c_{jj'} \leq 32C_j^*$ and $C_{j'}^* \leq c_{jj'}$. If there is such a location $j'$, then the entire demand of $j$ is transferred to $j'$, i.e., set $d''_{j'}$ to $d''_{j'} + d''_j$, and $d''_j$ to $0$.

Note that the condition on the type of $j$ and $j'$ ensures us that the demand $j$ can use the facilities that $j'$ is assigned to.

Let $\mathcal{I}'$ be the new instance obtained thus. Let $\mathcal{D}'$ denote the locations $j$ for which $d''_j > 0$. It is easy to see that $x^*, y^*$ is still a feasible fractional solution for the modified instance. It is also easy to see that an integral solution to the original LP can be obtained from an integral solution to this modified instance with at most a constant factor loss in approximation.

**Observation 6.1** *For any two demands $j, j'$ of the same type, $c_{jj'} > 32 \cdot max\{C_j^*, C_{j'}^*\}$.*

**Observation 6.2** *For any two demands $j \in \mathcal{D}'_2$ and $j' \in \mathcal{D}'_1$, if $C_{j'}^* \leq c_{jj'}$, then $c_{jj'} > 32 \cdot C_j^*$.*

### 6.2.2 Scenarios

**Definition** *Ball(p,r) :* For a location $p$, let $Ball(p, r)$ denote the set of all the locations at a distance of at most $r$ from $p$.

**Definition** *Nearest Assignable Demand, Critical Radius, Critical Ball :* With every demand $j \in \mathcal{D}'$, we associate another demand from $\mathcal{D}'$, denoted by $n(j)$ which we call its nearest assignable demand. The critical radius is denoted by $r_j$ and is defined to be $c_{jn(j)}/16$. The critical ball, denoted by $\mathcal{B}_j$, is defined to be $Ball(j, r_j)$ The nearest assignable demand is determined as follows:

- If $j \in \mathcal{D}'_1$, $n(j) = \text{argmin}_{j' \in \mathcal{D}'_1 \setminus \{j\}}\{c_{jj'}\}$
- If $j \in \mathcal{D}'_2$, let $\mathcal{V} = \{j' \in \mathcal{D}'_1 | c_{jj'} \leq \frac{3}{2} r_{j'}\} \cup \{j\}$. Then $n(j) = \text{argmin}_{j' \in \mathcal{D}'_1 \cup \mathcal{D}'_2 \setminus \mathcal{V}}\{c_{jj'}\}$.

If there is no demand that is a candidate to be a nearest assignable demand to $j$, then we set $n(j) = \Gamma$ and $r_j$ to be twice the distance to the furthest location (demand or facility). Thus, in this case, all facilities lie in $\mathcal{B}_j$.

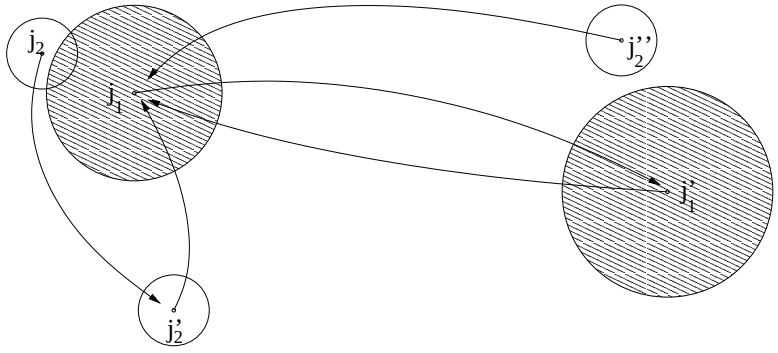


Figure 6.2: Nearest assignable demands

Figure 6.2 shows an example of nearest assignable demands for different demands under different situations. Let $j_1, j'_1 \in \mathcal{D}'_1$ and $j_2, j'_2, j''_2 \in \mathcal{D}'_2$. Note that for a $type(1)$ demand, the nearest assignable demand is always the closest $type(1)$ demand. Hence $j_1$ and $j'_1$ are nearest assignable demands of each other. For the demand $j_2$, $j_1$ is not a candidate as it is not sufficiently far, i.e., $c_{j_1 j_2} \leq \frac{3}{2} r_{j_1}$, and hence the nearest assignable demand is the next closest demand, i.e., $j'_2$. For $j'_2$, the closest demand is $j_1$ and it is sufficiently far and therefore $j_1$ is the nearest assignable demand

for $j_2'$. $j_2''$ is again too close to $j_1'$ and therefore its nearest assignable demand is the next closest demand $j_1$.

As we will see later, we can modify the solution by suffering a constant factor loss in our approximation, so that any demand $j$ uses facilities which are either close to itself or are close to its nearest assignable demand $n(j)$.

**Observation 6.3** *For a demand $j$, such that $n(j) = \Gamma$, there is only one demand of type $type(j)$, i.e., $j$.*

**Observation 6.4** *For a demand $j$, such that $n(j) = \Gamma$, for all $j' \in \mathcal{D}'$, $n(j') \neq j$.*

This is straightforward to see for a $type(2)$ demand. For a $type(1)$ demand, this must be the only $type(1)$ demand. The critical radius is set so large that all demands lie in the critical ball, and therefore this demand is not a candidate to be the nearest assignable demand for any $type(2)$ demand.

**Lemma 6.2** *If there exists a demand $j \in \mathcal{D}_2'$, such that $n(j) = \Gamma$, then there is at most one $type(1)$ demand. If there is indeed such a $type(1)$ demand, say $j_1$, then $n(j_1) = \Gamma$.*

*Proof.* The proof is by contradiction. Suppose there is a demand $j_1 \in \mathcal{D}_1'$ such that $n(j_1) \neq \Gamma$. Let $j_2 = n(j_1)$, $j_3 = n(j_2)$ and so on. This sequence must end in a cycle formed by two demands, say $j'$ and $j''$ such that $n(j') = j''$ and $n(j'') = j'$. Note that it must be that $type(j') = type(j'') = 1$.

We will show that either $j'$ or $j''$ is a candidate for being the nearest assignable demand of $j$ and therefore $n(j)$ cannot be $\Gamma$. It must be true that $c_{jj'} \leq \frac{3}{2} r_{j'}$ and $c_{jj''} \leq \frac{3}{2} r_{j''}$, because otherwise one of them is a candidate nearest assignable demand of $j$. Also $r_{j'} = r_{j''} = c_{j'j''}/16$. Using these facts, we get that

$$c_{j'j''} \leq c_{jj'} + c_{jj''} \leq \frac{3}{2}(r_{j'} + r_{j''}) = \frac{3}{16} c_{j'j''} < c_{j'j''}.$$

Contradiction. ∎

**Observation 6.5** *For any demand $j$, such that $n(j) \neq \Gamma$, $type(n(j)) \leq type(j)$.*

**Observation 6.6** *For a demand $j$, such that $n(j) \neq \Gamma$, $c_{jn(j)} > \frac{3}{2} r_{n(j)}$.*

**Lemma 6.3** *For any demand $j$, such that $n(j) \neq \Gamma$, $r_j + r_{n(j)} < c_{jn(j)}$.*

*Proof.* By Observation 6.6 and the fact that $r_j = c_{jn(j)}/16$, it follows that

$$r_j + r_{n(j)} = \frac{1}{16} \cdot c_{jn(j)} + r_{n(j)} < \frac{1}{16} \cdot c_{jn(j)} + \frac{2}{3} \cdot c_{jn(j)} < c_{jn(j)}.$$

■

**Corollary 6.4** *For any demand $j$, such that $n(j) \neq \Gamma$, $\mathcal{B}_j \cap \mathcal{B}_{n(j)} = \phi$.*

We now show that at least a half fraction of any demand is assigned to facilities that lie within its *critical ball*.

**Lemma 6.5** *For any demand $j \in \mathcal{D}'$, $Ball(j, 2 \cdot C^*_j) \subseteq \mathcal{B}_j$.*

*Proof.* If $n(j) = \Gamma$, then the claim clearly holds true. Suppose not.

First consider $j \in \mathcal{D}'_1$. Then by Observation 6.1,

$$r_j = \frac{1}{16} \cdot c_{jn(j)} > \frac{1}{16} \cdot 32C^*_j = 2C^*_j.$$

Now consider $j \in \mathcal{D}'_2$. If $type(n(j)) = 2$, then by Observation 6.1,

$$r_j = \frac{1}{16} \cdot c_{jn(j)} > \frac{1}{16} \cdot 32C^*_j = 2C^*_j.$$

Otherwise $type(n(j)) = 1$. In this case, consider the demand $n(j)$. By the discussion above and Lemma 6.3,

$$C^*_{n(j)} < r_{n(j)} < c_{jn(j)}.$$

Therefore, by observation 6.2,

$$r_j = \frac{1}{16} \cdot c_{jn(j)} > \frac{32}{16} \cdot C^*_j = 2C^*_j.$$

This completes the proof of the Lemma. ■

We now define scenarios. For this, we construct graphs on the set of demands, which we call *intersection graphs*.

**Definition** *Intersection graphs :* The *level 1 intersection graph* is the graph $G_1 = (V, E_1)$, where $V = \mathcal{D}'$ and $(j_1, j_2) \in E_1$ iff $\mathcal{B}_{j_1} \cap \mathcal{B}_{j_2} \neq \phi$ and $j_1, j_2 \in \mathcal{D}'_1 \cup \mathcal{D}'_2$. We will later see that any connected component in this graph has at most one type(1) demand.

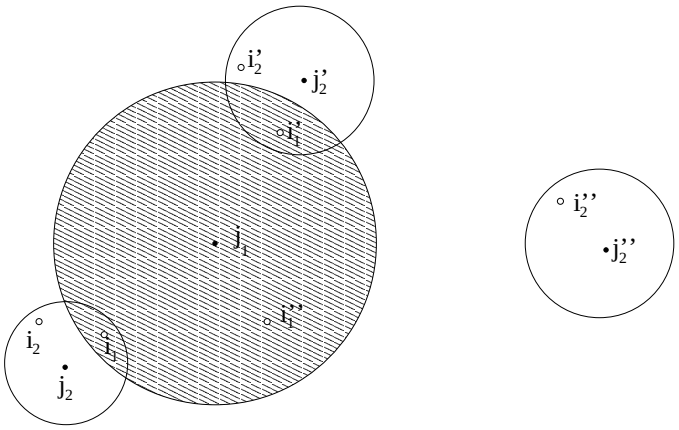


Figure 6.3: Example of Level 1 and Level 2 Scenarios

The *level 2 intersection graph* is the graph $G_2 = (V, \phi)$, where $V = \mathcal{D}'$. Therefore, there are no edges in a *level 2 intersection graph*. All connected components in this graph are isolated demands.

Let $\mathcal{CC}_G(j)$ denote the set of demands that are in the same connected component of the graph $G$ as $j$.

**Definition** *Scenarios :* We define a *level 1* scenario for every connected component in $G_1$. Let $\mathcal{J}$ be the set of demands in the connected component. Then the corresponding *level 1* scenario is defined by the set of facilities $\cup_{j \in \mathcal{J}}(\mathcal{B}_j \cap \mathcal{F})$.

Similarly, we define a *level 2* scenario for every connected component in $G_2$. Let $j$ be a demand in $G_2$.

- If $j \in \mathcal{D}'_2$, then the *level 2* scenario is defined by the set of facilities $\mathcal{B}_j \cap \mathcal{F}$
- If $j \in \mathcal{D}'_1$, then the *level 2* scenario is defined by the set of facilities $\{i \in \mathcal{B}_j \cap \mathcal{F} | i \notin \mathcal{B}_{j'}$ for any $j' \in \mathcal{CC}_{G_1}(j) \setminus \{j\}\}$.

Thus every *level 1* scenario is the union of some *level 2* scenarios. We denote the *level k* scenario to which a demand $j$ belongs by $\mathcal{S}_k(j)$. Also, if $k = type(j)$, then we simply denote the scenario as $\mathcal{S}(j)$, i.e. $\mathcal{S}(j) = \mathcal{S}_{type(j)}(j)$.

Figure 6.3 illustrates facilities in *level 1* and *level 2* scenarios. In this example, $\mathcal{S}_1(j_1) = \{i_1, i'_1, i''_1, i_2, i'_2\}$, $\mathcal{S}_2(j_2) = \{i_1, i_2\}$, $\mathcal{S}_2(j'_2) = \{i'_1, i'_2\}$ and $\mathcal{S}_2(j_1) = \{i''_1\}$. Also, $\mathcal{S}_1(j''_2) = \mathcal{S}_2(j''_2) = \{i''_2\}$

**Lemma 6.6** *Let $j_1 \in \mathcal{D}'_1$ and $j_2 \in \mathcal{D}'_2$ such that $n(j_1) \neq \Gamma$, $n(j_2) \neq \Gamma$ and $\mathcal{B}_{j_1} \cap \mathcal{B}_{j_2} \neq \phi$. Then*

1. $r_{j_2} < 1.1 r_{j_1}$
2. *For any facility $i \in \mathcal{B}_{j_2}$, $c_{ij_1} < 4 \cdot r_{j_1}$.*

*Proof.* If $c_{j_1 j_2} > \frac{3}{2} r_{j_1}$, then $j_1$ is a candidate to be the nearest assignable demand of $j_2$ and therefore by Lemma 6.3, $\mathcal{B}_{j_1} \cap \mathcal{B}_{j_2} = \phi$. Hence, it must be that

$$c_{j_1 j_2} \quad \leq \quad \frac{3}{2} r_{j_1} \tag{6.9}$$

Let $j'_1 = n(j_1)$. Then by inequality (6.9), it follows that

$$c_{j'_1 j_2} \geq c_{j_1 j'_1} - c_{j_1 j_2} \geq c_{j_1 j'_1} - \frac{3}{2} r_{j_1} = c_{j_1 j'_1} - \frac{3}{2} \cdot \frac{1}{16} \cdot c_{j_1 j'_1} = \frac{29}{32} \cdot c_{j_1 j'_1} \geq \frac{29}{32} \cdot 16 r_{j'_1} > \frac{3}{2} \cdot r_{j'_1}.$$

Hence $j'_1$ is a candidate to be the nearest assignable demand of $j_2$. This implies that

$$r_{j_2} \quad \leq \quad \frac{1}{16} \cdot c_{j'_1 j_2} \tag{6.10}$$

By inequality (6.9) and the fact that $j'_1 = n(j_1)$, it follows that

$$c_{j'_1 j_2} \leq c_{j_1 j'_1} + c_{j_1 j_2} = 16 r_{j_1} + c_{j_1 j_2} \leq 16 r_{j_1} + \frac{3}{2} \cdot r_{j_1} = 17.5 r_{j_1} \tag{6.11}$$

Also, by inequalities (6.10) and (6.11), we get that

$$r_{j_2} \leq \frac{c_{j'_1 j_2}}{16} \leq \frac{17.5}{16} r_{j_1} < 1.1 r_{j_1} \tag{6.12}$$

This proves claim 1 of the Lemma.

Now, let $i$ be any facility in $\mathcal{B}_{j_2}$. By the fact that $\mathcal{B}_{j_1} \cap \mathcal{B}_{j_2} \neq \phi$ and inequality (6.12), we get that

$$c_{ij_1} \leq c_{j_1 j_2} + c_{ij_2} \leq (r_{j_1} + r_{j_2}) + c_{ij_2} \leq r_{j_1} + 2 \cdot r_{j_2} < r_{j_1} + 2.2 r_{j_1} < 4 r_{j_1}$$

This completes the proof of Claim 2 of the Lemma as well. ∎

We now show that for any demand $j$ of type $k$ $(= 1, 2)$, the distance from the demand to any facility in $\mathcal{S}_k(j)$ cannot be more than $4 \cdot r_j$.

**Lemma 6.7** *Let $j$ be any demand. Let $k = type(j)$. Consider the level $k$ intersection graph, $G_k$. Let $\mathcal{C}$ be the connected component in $G_k$ spanning the demands constituting the scenario $\mathcal{S}(j)$. Then*

1. *There is only one type(k) demand in $\mathcal{C}$, i.e., $j$.*
2. *For any facility $i \in \mathcal{S}(j)$, $c_{ij} \leq 4 \cdot r_j$.*

*Proof.* Note that $G_2$ consists of disconnected points, i.e., it does not contain any edges. If $j \in \mathcal{D}'_2$, then clearly $j$ is the only *level 2* demand in its connected component in $G_2$. Moreover, $\mathcal{S}(j)$ only consists of the facilities in $\mathcal{B}_j$ and therefore for any facility $i \in \mathcal{S}(j)$, $c_{ij} \leq r_j < 4 \cdot r_j$.

Now suppose that $j \in \mathcal{D}'_1$. Suppose there are 2 facilities of $type(1)$ in C. Let $j_1$ and $j'_1$ be these facilities. Note that $j_1$ and $j'_1$ are mutual candidates for being the nearest assignable demands of each other. Therefore $r_{j_1}, r_{j'_1} \leq c_{j_1 j'_1}/16$. By observation 6.1, we know that the critical balls of two demands of the same type cannot overlap. Therefore, it must be that there is a demand $j_2$ of $type(2)$ such that $\mathcal{B}_{j_1} \cap \mathcal{B}_{j_2} \neq \phi$ and $\mathcal{B}_{j'_1} \cap \mathcal{B}_{j_2} \neq \phi$. Let $i$ be any facility in $\mathcal{B}_{j_1} \cap \mathcal{B}_{j_2}$. Then, since $j'_1$ is a candidate to be $n(j_1)$, it follows that

$$c_{ij_1} \leq r_{j_1} \leq \frac{1}{16} \cdot c_{j_1 j'_1} \tag{6.13}$$

Also, by Lemma 6.6 and the fact that $j_1$ is a candidate to be $n(j'_1)$, we get that

$$c_{ij'_1} \leq 4 \cdot r_{j'_1} \leq \frac{1}{4} \cdot c_{j_1 j'_1} \tag{6.14}$$

Using inequalities (6.13) and (6.14), we finally get that

$$c_{j_1 j'_1} \leq c_{ij_1} + c_{ij'_1} \leq \frac{1}{16} \cdot c_{j_1 j'_1} + \frac{1}{4} \cdot c_{j_1 j'_1} < c_{j_1 j'_1}$$

Hence, there cannot be two facilities of $type(1)$ in $\mathcal{C}$. This proves Claim 1 of the Lemma. Given this, Claim 2 of the Lemma follows from Lemma 6.6. ∎

### 6.2.3 Changing the assignments

We now modify the assignments so that every demand either uses facilities in its own scenario or the scenario of its nearest assignable demand.

**Lemma 6.8** *By suffering a loss of at most a constant factor in approximation, we can modify the assignments such that for any demand $j$, it either uses the facilities in $\mathcal{S}(j)$ or facilities in $\mathcal{S}(n(j))$.*

*Proof.* If $n(j) = \Gamma$, then trivially, the demand is assigned to facilities within its scenario. Suppose otherwise.

Note that at least a $\frac{1}{2}$ fraction of any demand $j$ can be assigned to facilities in $\mathcal{B}_j$ and therefore $\mathcal{S}(j)$. Also note that $\frac{1}{2}$ fraction of any demand $j$ can be assigned to facilities in $\mathcal{B}_{n(j)}$ and therefore $\mathcal{S}(n(j))$.

We now assign $j$ to facilities (that can be used by $j$) in order of distance from $j$. Note that this can only reduce the overall cost. Also, now at least $\frac{1}{2}$ fraction of demand $j$ is assigned to facilities in $\mathcal{B}_j$ and therefore at most $\frac{1}{2}$ fraction is assigned to facilities outside $\mathcal{B}_j$.

Let $i \notin \mathcal{B}_j$ be any facility such that $x^*_{ij} > 0$. Then

$$c_{ij} > r_j = \frac{1}{16} \cdot c_{jn(j)} \tag{6.15}$$

Let $i'$ be any facility in $\mathcal{S}(n(j))$. Then by Lemma 6.7, Observation 6.6 and inequality (6.15), it follows that

$$\begin{aligned} c_{i'j} \quad &\leq c_{jn(j)} + c_{i'n(j)} \leq c_{jn(j)} + 4r_{n(j)} \\ &\leq c_{jn(j)} + 4 \cdot \frac{2}{3} \cdot c_{jn(j)} = \frac{11}{3} \cdot c_{jn(j)} \leq \frac{176}{3} \cdot c_{ij} \end{aligned} \tag{6.16}$$

Hence we can reassign $j$ to a usable facility in $\mathcal{S}(n(j))$ without increasing the cost by more than a constant factor. ∎

We now modify the assignment so that the cost paid by a demand to any facility in the scenario of its nearest assignable demand is the same. Therefore, it does not matter which facility it uses in that scenario. We will denote the (new) distance of demand $j$ to facilities in the scenario of its nearest assignable demand by $\bar{c}_j$.

**Lemma 6.9** *For any demand $j$ and facility $i \in \mathcal{S}(n(j))$, we set $c_{ij} = \frac{11}{3} c_{jn(j)} (= \bar{c}_j)$. This increases the cost of the solution by at most a constant factor.*

*Proof.* Follows from the fact that for any facility $i \in \mathcal{S}(n(j))$, $c_{ij} \geq \frac{1}{16} c_{jn(j)}$ and $c_{ij} \leq \frac{11}{3} c_{jn(j)}$. ∎

Therefore, we can now modify the assignments (going across to facilities that belong to the scenario of nearest assignable demands) so that a demand uses the same facilities from the nearest

assignable demand that are used by the nearest assignable demand itself.

We now modify the assignments so as to reduce the number of open facilities in *level 2* scenarios.

We first describe a simple structure that we desire the solution to exhibit. If a solution exhibits this structure, we say that it satisfies the *Structure Property*.

**Definition** A solution $(x, y)$ to the LP defined above is said to satisfy the *Structure Property* if

1. For every demand $j \in \mathcal{D}'_1$,
   (a) There is no $type(2)$ facility in $\mathcal{S}_2(j)$.
   (b) There is at most one facility of $type(1)$ in $\mathcal{S}_2(j)$.
   (c) If there is a $type(1)$ facility in $\mathcal{S}_2(j)$, then this facility is closer to $j$ than any other $type(1)$ facility in $\mathcal{S}_1(j)$.
2. For every demand $j \in \mathcal{D}'_2$,
   (a) There is at most one facility of $type(2)$ in $\mathcal{S}_2(j)$.
   (b) If there is no $type(1)$ demand in $\mathcal{CC}_{G_1}(j)$, then there is at most one facility of $type(1)$ in $\mathcal{S}_2(j)$.
   (c) If $j_1 \in \mathcal{D}'_1 \cap \mathcal{CC}_{G_1}(j)$ and $r_j \leq c_{jj_1}/4$, then there is at most one facility of $type(1)$ in $\mathcal{S}_2(j)$.
   (d) If $j_1 \in \mathcal{D}'_1 \cap \mathcal{CC}_{G_1}(j)$ and $r_j > c_{jj_1}/4$, then there are at most two facilities of $type(1)$ in $\mathcal{S}_2(j)$. Moreover, If there are indeed two facilities, say $i$ and $i'$, then for one of these facilities, say $i$,
      * $x_{ij_1} = y_i = 1$,
      * $0 < x_{ij} < y_i$ ; and
      * $i$ is the closest $type(1)$ facility to $j_1$.

**Lemma 6.10** *By increasing the cost of the solution by at most a constant factor, we can modify the solution, so that it satisfies the* Structure Property.

*Proof.* Let $j \in \mathcal{D}'_1$. Suppose that there is a $type(2)$ facility $i_2 \in \mathcal{S}_2(j)$. Note that all $type(2)$ demands in $\mathcal{CC}_{G_1}(j)$ are assigned to facilities in their own scenario or in the scenario of their nearest assignable demand. Therefore facilities in $\mathcal{S}_2(j)$ are not used by these demands. Moreover, demands

from other *level 1* scenarios that use $i_2$ can be reassigned to use other $type(1)$ demands in $\mathcal{S}_1(j)$ without increasing the cost. Therefore there are no $type(2)$ facilities in $\mathcal{S}_2(j)$. This proves Claim 1 (a) of the *Structure Property*. Note that all the $type(1)$ facilities in $\mathcal{S}_2(j)$ are only used by $j$ or demands from other *level 1* scenarios. So, we can simply move them to the $type(1)$ facility that is closest to $j$, say $i$. This completes the proof for Claim 1 (b) of the *Structure Property*. Now, if there is another facility $i' \in \mathcal{D}'_1 \cap \mathcal{S}_1(j)$ such that $c_{i'j} \leq c_{ij}$, then we can simply relocate $i$ to $i'$ and alter the assignments accordingly, without increasing the cost. This completes the proof for Claim 1 (c) of the *Structure Property*.

Now, consider $j \in \mathcal{D}'_2$. Note that $\mathcal{S}_2(j) = \mathcal{S}(j)$. If there are multiple facilities of $type(2)$, then we simply relocate all the open $type(2)$ facilities to the $type(2)$ facility that is closest to $j$, and alter the assignments accordingly. This completes the proof for Claim 2 (a) of the *Structure Property*.

If there is no $type(1)$ demand in $\mathcal{CC}_{G_1}(j)$, then we can simply relocate all the $type(1)$ facilities in $\mathcal{S}(j)$ to the $type(1)$ facility that is closest to $j$ without increasing the cost. This completes the proof of Claim 2 (b) of the *Structure Property*.

Otherwise, let $j_1 \in \mathcal{D}'_1 \cap \mathcal{CC}_{G_1}(j)$. If there is only one $type(1)$ facility in $\mathcal{S}(j)$ then there is nothing to do. Suppose that there are multiple facilities of $type(1)$ in $\mathcal{S}_2(j)$. Consider the following cases:

- *If* $r_j \leq c_{jj_1}/4$ *:*
  Then all the facilities in $\mathcal{B}_j$ can be considered to be roughly at the same distance from $j_1$ (within a constant factor). We can therefore relocate all the $type(1)$ facilities to the $type(1)$ facility that is closest to $j$ without increasing the cost by more than a factor of $\frac{5}{3}$, leaving us with only one $type(1)$ facility in the scenario.

  This completes the proof of Claim 2 (c) of the *Structure Property*.

- *If* $r_j > c_{jj_1}/4$ *:* We make the following claim.

  **Claim 6.2.1** *If there is any other demand* $j_2 \in \mathcal{D}'_2 \cap \mathcal{CC}_{G_1}(j)$*, then* $c_{ij_1} \leq c_{i'j_1}$ *for any two facilities* $i \in \mathcal{B}_j$ *and* $i' \in \mathcal{B}_{j_2}$*.*

  *Proof.* Note that since $j$ and $j_2$ are mutual candidates to be the nearest assignable demands of each other,

$$r_j, r_{j_2} \quad \leq \quad \frac{1}{16} \cdot c_{jj_2} \tag{6.17}$$

Now let $i$ be any facility in $\mathcal{B}_j$. Then by inequality (6.17), it follows that

$$c_{ij_1} \leq c_{ij} + c_{jj_1} \leq r_j + c_{jj_1} < r_j + 4r_j = 5r_j \leq \frac{5}{16} \cdot c_{jj_2} \tag{6.18}$$

Moreover, let $i'$ be any facility in $\mathcal{B}_{j_2}$. Then using inequality (6.17), we get that

$$\begin{aligned} c_{i'j_1} &\geq c_{j_1j_2} - c_{i'j_2} \geq c_{j_1j_2} - r_{j_2} \geq c_{j_1j_2} - \frac{1}{16} \cdot c_{jj_2} \geq (c_{jj_2} - c_{jj_1}) - \frac{1}{16} \cdot c_{jj_2} \\ &= \frac{15}{16} \cdot c_{jj_2} - c_{jj_1} > \frac{15}{16} \cdot c_{jj_2} - 4r_j \geq \frac{15}{16} \cdot c_{jj_2} - \frac{4}{16} \cdot c_{jj_2} = \frac{11}{16} \cdot c_{jj_2} \end{aligned} \tag{6.19}$$

Thus, equations (6.18) and (6.19) together imply that $c_{ij_1} \leq c_{i'j_1}$.

This completes the proof of Claim 6.2.1. ∎

Now, note that for facilities in $\mathcal{S}(j)$ that are used both by $j_1$ and $j$, the cost of relocating a small fraction from one facility to another is equal and opposite in cost to that of relocating in the opposite direction. We can therefore relocate in the direction that reduces the cost. We can repeat this till we are left with only one facility which is used by both $j_1$ and $j$. Also, we can relocate all the $type(1)$ facilities in $\mathcal{S}(j)$ that are not used by $j_1$ to the $type(1)$ facility that is closest to $j$ without increasing the cost.

Now if we have only one $type(1)$ facility left in $\mathcal{S}(j)$, then we are done.

Suppose not. Then, for any $type(1)$ facility $i \in \mathcal{S}(j)$ such that $0 < x_{ij} < y_i$, we can relocate $(y_i - x_{ij})$ fraction from $i$ to the $type(1)$ facility that is closest to $j_1$ without increasing the cost. Therefore, there is only at most one $type(1)$ facility $i \in \mathcal{S}(j)$ such that $x_{ij} < y_i$. If there is indeed such a facility, then amongst the facilities in $\mathcal{S}(j)$, this facility must be closest to $j_1$. Also, by Claim 6.2.1, this facility does not belong to $\mathcal{S}_2(j_2)$ for any $j_2 \in \mathcal{D}'_2 \cap \mathcal{CC}_{G_1}(j)$. If $x_{ij} = 0$, we move $i$ from $\mathcal{S}(j)$ and consider it to be a part of $\mathcal{S}_2(j_1)$.

If there is only one $type(1)$ facility remaining in $\mathcal{S}(j)$, then we are done. If there is more than one facility, then note that $x_{ij} > 0$ for all $i \in \mathcal{F}_1 \cap \mathcal{S}(j)$. Also, $j_1$ uses only one of these facilities since there can only be one facility that is used by both $j$ and $j_1$ (by adjustments above). Hence, it must be that there are two $type(1)$ facilities in $\mathcal{S}_2(j)$ such that, one of the facilities, say $i_1$, is used by both $j_1$ and $j$ and the other, say $i'_1$, is only used by $j$. Also, $c_{i'_1j} \leq c_{i_1j}$ otherwise we can just relocate $i'_1$ to $i_1$ decreasing the cost.

Moreover, it must be that $x_{i_1 j_1} = y_{i_1}$, otherwise we can relocate some fraction from $i_1$ to $i'_1$, thereby reducing the cost.

Now consider the following cases:

- If $c_{i_1 j} \leq c_{i_1 j_1}$
  Then we relocate $i_1$ to $i'_1$. Note that this reduces the cost paid by $j$ and increases the cost paid by $j_1$ at most by a factor of 3 since

$$c_{i'_1 j_1} \leq c_{i'_1 j} + c_{i_1 j} + c_{i_1 j_1} \leq 2c_{i_1 j} + c_{i_1 j_1} \leq 3c_{i_1 j_1}$$

  This leaves us with only one facility in $\mathcal{S}(j)$ and therefore we are done.

- If $c_{i_1 j_1} < c_{i_1 j}$
  We again consider the following cases:

  * *If there is another facility in $\mathcal{B}_{j_1}$ (say $i''_1$) such that $c_{i''_1 j_1} \leq c_{i_1 j_1}$ :*
    Then we can move from $i_1$ to $i''_1$ until either the sum of the facilities in $\mathcal{B}_j$ becomes $\frac{1}{2}$ or $i_1$ is shut down. Note that this only increases the cost paid by $j$ by a factor of at most 3 since

$$c_{i''_1 j} \leq c_{i''_1 j_1} + c_{i_1 j_1} + c_{i_1 j} \leq 2c_{i_1 j_1} + c_{i_1 j} \leq 3c_{i_1 j}$$

    If $i_1$ is shut down, then we are left with only one facility in $\mathcal{S}_2(j_2)$ and therefore we are done.
    If sum of facilities in $\mathcal{S}_2(j_2)$ becomes $\frac{1}{2}$, then note that $j$ is already paying a cost of $\frac{1}{2}r_j$. Therefore, moving from $i'_1$ to $i_1$ can only at most double the cost paid by $j$. Therefore we are again left with only one $type(1)$ facility in $\mathcal{B}_j$.

  * *If there is no other closer facility in $\mathcal{B}_{j_1}$ :*
    By Claim 6.2.1, we know that any other $type(1)$ facility must be at least as far from $j_1$ as $i'_1$. Since $j_1$ does not use $i'_1$, it must be that $x_{i_1 j_1} = y_{i_1} = 1$. Also, $x_{i_1 j} < y_{i_1}$.

  This completes the proof of Claim 2 (d) of the *Structure Property*.

∎

Therefore a $type(2)$ demand, say $j$, may have one or two $type(1)$ facilities in its scenario. These settings are illustrated in Figure 6.4.

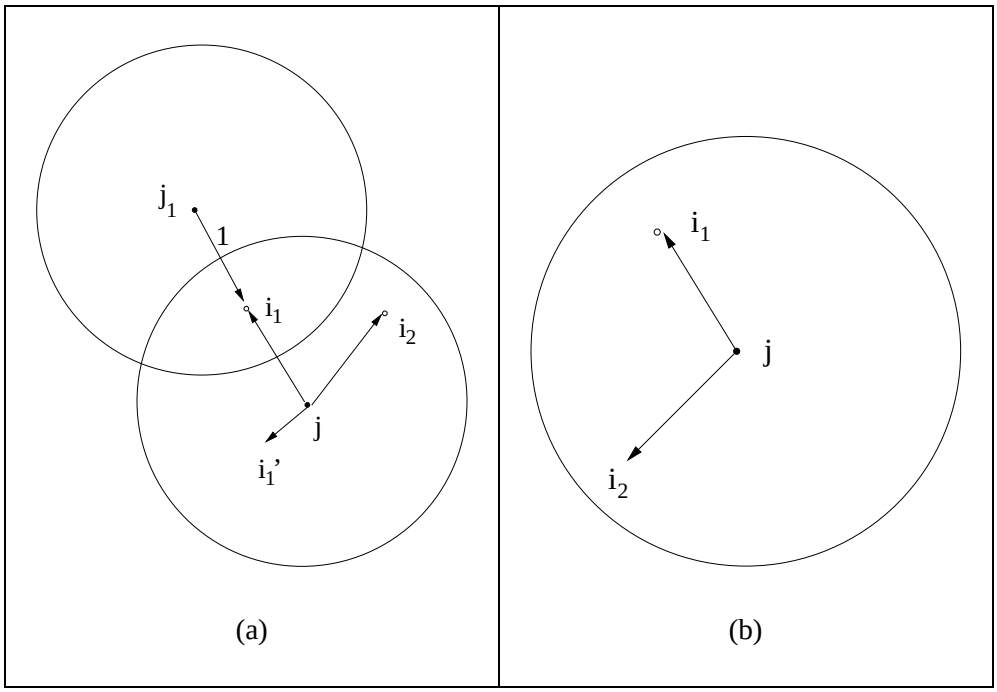


Figure 6.4: Different settings for $type(2)$ demands

### 6.2.4 Modified LP

We now present an LP for this modified problem instance. Let $\mathcal{F}'(j)$ denote the set of facilities used by demand $j$, i.e., $\{i \in \mathcal{F} | x_{ij} > 0\}$. Let $\mathcal{F}'$ denote the set of facilities used by at least one demand. Let $\mathcal{F}'_1$ and $\mathcal{F}'_2$ denote the set of facilities in $\mathcal{F}'$ of $type(1)$ and $type(2)$ respectively.

We consider the modified LP described as follows. We replace the variables $y_i$'s with $\bar{y}_i$'s and

$x_{ij}$'s with $\bar{x}_{ij}$'s when solving the modified LP.

$$\min \quad \sum_{j \in \mathcal{D}', i \in \mathcal{F}'} d_j \cdot c_{ij} \cdot \bar{x}_{ij} \tag{6.20}$$

$$\sum_{i \in \mathcal{F}'_r} \bar{y}_i \leq k_r \qquad \text{for } r = 1, \ldots, m \tag{6.21}$$

$$\sum_{i \in \mathcal{F}'(j)} \bar{x}_{ij} = 1 \qquad \text{for all demands } j \tag{6.22}$$

$$\sum_{i \in \mathcal{F}'(j) \cap \mathcal{S}(j)} \bar{x}_{ij} \geq \frac{1}{2} \qquad \text{for all demands } j \tag{6.23}$$

$$\bar{x}_{ij} = 1 \qquad \text{if } x_{ij} = 1 \tag{6.24}$$

$$\bar{y}_i = 1 \qquad \text{if } y_i = 1 \tag{6.25}$$

$$\bar{x}_{ij} \leq \bar{y}_i \qquad \text{for all demands } j \text{ and facilities } i \in \mathcal{F}' \tag{6.26}$$

$$\bar{x}_{ij} \geq 0 \qquad \text{for all demands } j \text{ and facilities } i \in \mathcal{F}' \tag{6.27}$$

$$\bar{y}_i \geq 0 \qquad \text{for all facilities } i \in \mathcal{F}' \tag{6.28}$$

Note that the modified solution obtained by changing the assignments described above is a feasible solution to this new LP. Therefore the optimal solution to this LP can only have cost at most as much as the above solution.

**Lemma 6.11** *The assignments in the optimal solution of the modified LP,* $(\bar{x}, \bar{y})$ *continue to satisfy the* Structure Property.

*Proof.* For Claim 2 (d) of the *Structure Property*, the first part follows from constraints (6.24) and (6.25). If $\bar{x}_{ij} = 0$, then it can be moved out to $\mathcal{S}_2(j_1)$. If $\bar{x}_{ij} = \bar{y}_i = 1$, then $i'$ can be shutdown. This shows the second part.

All the remaining claims follow from the fact that we do not introduce any new facilities. ∎

### 6.2.5 Half-Integrality of a Vertex Solution

We now show that the solution to the modified LP is half integral.

**Definition** *Facility relocation cost :*
For a *level 1* scenario, $\mathcal{S}_1(j)$, let $\mathcal{D}(\mathcal{S}_1(j)) = \mathcal{D}' \cap \mathcal{CC}_{G_1}(j)$. For a *level 2* scenario, $\mathcal{S}_2(j)$, let

$\mathcal{D}(\mathcal{S}_2(j)) = \mathcal{D}'_2 \cap \mathcal{CC}_{G_2}(j)$. Note that for a $type(1)$ demand $j_1$, $\mathcal{D}(\mathcal{S}_2(j_1)) = \phi$ and for a $type(2)$ demand $j_2$, $\mathcal{D}(\mathcal{S}_2(j_2)) = \{j_2\}$.

Let $\mathcal{C}$ denote a chain of facilities $< i_1, i_2, .., i_s >$. We define $\mathcal{C} + \delta$ to be the operation where the extents to which the facilities are open are modified as follows:

- $\bar{y}_{i_t} = \bar{y}_{i_t} + \delta$, $t$ is odd
- $\bar{y}_{i_t} = \bar{y}_{i_t} - \delta$, $t$ is even

Let $\Delta(\mathcal{D}(\mathcal{S}), \mathcal{C} + \delta)$ denote the change in the cost paid by the demands $\mathcal{D}(\mathcal{S})$ when the operation $\mathcal{C} + \delta$ is performed.

Similarly, we define $\mathcal{C} - \delta$ to be the operation where the extents to which the facilities are open are modified as follows:

- $\bar{y}_{i_t} = \bar{y}_{i_t} - \delta$, $t$ is odd
- $\bar{y}_{i_t} = \bar{y}_{i_t} + \delta$, $t$ is even

Let $\Delta(\mathcal{D}(\mathcal{S}), \mathcal{C} - \delta)$ denote the change in the cost paid by the demands $\mathcal{D}(\mathcal{S})$ when the operation $\mathcal{C} - \delta$ is performed.

Our choice of the chain and the choice of $\delta$ (small enough quantity) will be such that all constraints of the modified LP will be satisfied even after the change except possibly constraint (6.21). Moreover, we will operate on multiple such (suitably selected) chains together so that even this constraint is not violated.

**Definition** Let $\mathcal{Y}_r(\mathcal{S}(j))$ denote the sum of all the $type(r)$ facilities in the scenario $\mathcal{S}(j)$ associated with demand $j$, i.e., $\mathcal{Y}_r(\mathcal{S}(j)) = \sum_{i \in \mathcal{F}'_r \cap \mathcal{S}(j)} \bar{y}_i$.

Let $\mathcal{Y}(\mathcal{S}(j))$ denote the sum of all the facilities in the scenario $\mathcal{S}(j)$ associated with demand $j$, i.e., $\mathcal{Y}(\mathcal{S}(j)) = \sum_{i \in \mathcal{F}' \cap \mathcal{S}(j)} \bar{y}_i = \mathcal{Y}_1(\mathcal{S}(j)) + \mathcal{Y}_2(\mathcal{S}(j))$.

The idea is to find a chain of facilities, $\mathcal{C}$, such that the cost of performing the operation $\mathcal{C} + \delta$ is equal but opposite the cost of performing the operation $\mathcal{C} - \delta$, i.e., $\Delta(\mathcal{D}', \mathcal{C} + \delta) = -\Delta(\mathcal{D}', \mathcal{C} - \delta)$. Moreover, none of these facilities should be half-integral and also the total sum of all the $type(1)$ facilities and $type(2)$ facilities should not be disturbed.

We show that if the solution is not half-integral, it is always possible to find such a chain. We can then essentially adjust the chain along the direction that does not increase the cost of the solution.

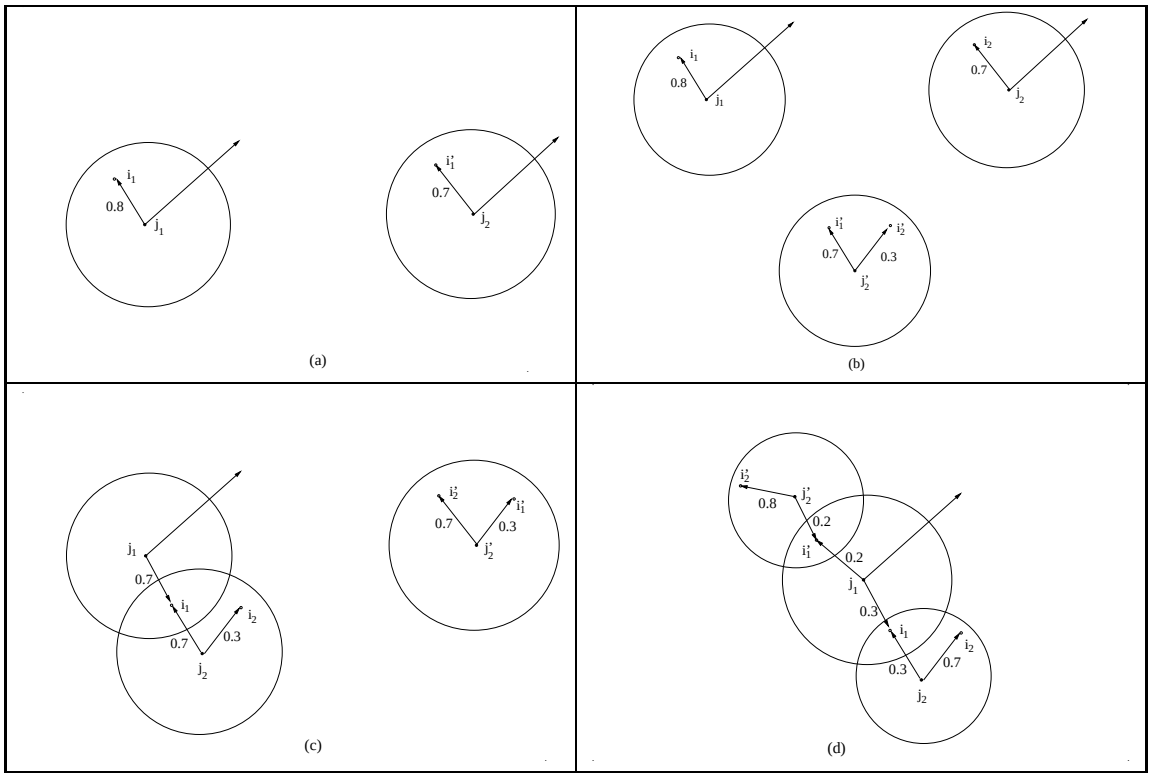


Figure 6.5: Examples of facility adjustments

The ability to do this implies that the solution to the LP is either not an optimal solution or is a non-vertex solution. This is a contradiction, implying that such a chain cannot exist and therefore the solution must be half integral.

For a demand $j$ that is assigned to facilities in $n(j)$, let $\bar{c}_j$ denote the distance to the facilities of $n(j)$.

Before getting into the technical details of proving the above claim, we illustrate the idea by means of some examples (see Figure 6.5).

We start with a simple example in Figure 6.5 (a). There are 2 demands $j_1$ and $j_2$ assigned to facilities of the $type(1)$ within their critical balls. Therefore $type(i_1) = type(i'_1) = 1$. In this example $x_{i_1 j_1} = y_{i_1} = 0.8$ and $x_{i'_1 j_2} = y_{i'_1} = 0.7$. Note that both the demands are also assigned to

facilities of their nearest assignable demands. Now consider the chain $\mathcal{C} = \{i_1, i'_1\}$. Then,

$$\Delta(\mathcal{D}', \mathcal{C} + \delta) = -\Delta(\mathcal{D}', \mathcal{C} - \delta) = \delta \cdot d_{j_1} \cdot (c_{i_1 j_1} - \bar{c}_{j_1}) + \delta \cdot d_{j_2} \cdot (\bar{c}_{j_2} - c_{i'_1 j_2}).$$

Therefore $\mathcal{C}$ is the required chain along which we can perform adjustments.

Things get more involved due to the existence of multiple types of facilities. This is illustrated in Figure 6.5 (b). In this example $type(j_1) = 1$, $type(j_2) = type(j'_2) = 2$, $type(i_1) = type(i'_1) = 1$ and $type(i_2) = type(i'_2) = 2$. Also, $x_{i_1 j_1} = y_{i_1} = 0.8$, $x_{i_2 j_2} = y_{i_2} = 0.7$, $x_{i'_1 j'_2} = y_{i'_1} = 0.7$ and $x_{i'_2 j'_2} = y_{i'_2} = 0.3$. Note that $\mathcal{Y}(\mathcal{S}(j'_2)) = 1$ and is already integral. Therefore, decreasing only one of the facilities will cause $j'_2$ to travel to facilities of its nearest assignable demand, which may incur a large cost. Therefore, we must include $i'_1$ and $i'_2$ together in any chain that we form. Consider the chain $\mathcal{C} = \{i_1, i'_1, i'_2, i_2\}$. Then,

$$\Delta(\mathcal{D}', \mathcal{C} + \delta) = -\Delta(\mathcal{D}', \mathcal{C} - \delta) = \delta \cdot (d_{j_1} \cdot (c_{i_1 j_1} - \bar{c}_{j_1}) + d_{j'_2} \cdot (c_{i'_2 j'_2} - c_{i'_1 j'_2}) + d_{j_2} \cdot (\bar{c}_{j_2} - c_{i'_2 j_2})).$$

Therefore $\mathcal{C}$ is the required chain along which we can perform adjustments.

The situation gets more complicated when demands of different types use the same facility lying in the intersection of their critical balls. This is illustrated in Figure 6.5 (c). In this example $j_1$ and $j_2$ use the same facility $i_1$. Moreover, $type(j_1) = 1$, $type(j_2) = type(j'_2) = 2$, $type(i_1) = type(i'_1) = 1$ and $type(i_2) = type(i'_2) = 2$. Also, $x_{i_1 j_1} = x_{i_1 j_2} = y_{i_1} = 0.7$, $x_{i_2 j_2} = y_{i_2} = 0.3$, $x_{i'_1 j'_2} = y_{i'_1} = 0.3$ and $x_{i'_2 j'_2} = y_{i'_2} = 0.7$. Note that $\mathcal{Y}(\mathcal{S}(j'_2)) = 1$ is already integral. Therefore, decreasing only one of the facilities will cause $j'_2$ to travel to facilities of its nearest assignable demand, which may incur a large cost. Moreover, $\mathcal{Y}(\mathcal{S}_2(j_2)) = 1$ is also integral. Therefore, decreasing only one of the facilities will cause $j_2$ to travel to facilities of its nearest assignable demand, which may incur a large cost. Therefore, we must include $i_1$ and $i_2$ together as well as $i'_1$ and $i'_2$ together in any chain that we form. Consider the chain $\mathcal{C} = \{i_1, i_2, i'_2, i'_1\}$. Then,

$$\Delta(\mathcal{D}', \mathcal{C} + \delta) = -\Delta(\mathcal{D}', \mathcal{C} - \delta) = \delta \cdot (d_{j_1} \cdot (c_{i_1 j_1} - \bar{c}_{j_1}) + d_{j_2} \cdot (c_{i_1 j_2} - c_{i_2 j_2}) + d_{j'_2} \cdot (c_{i'_2 j'_2} - c_{i'_1 j'_2})).$$

Therefore $\mathcal{C}$ is the required chain along which we can perform adjustments. Note that when adjusting $i_1$ it is important to ensure that increasing/decreasing it a small amount has equal and opposite impact on both $j_1$ as well as $j_2$ simultaneously.

Finally in Figure 6.5 (d), we show a *level 1* scenario for which the sum of both $type(1)$ and $type(2)$ facilities is half-integral. However, the facilities within the *level 2* scenarios are not half-

integral. In this example $type(j_1) = 1$, $type(j_2) = type(j_2') = 2$, $type(i_1) = type(i_1') = 1$ and $type(i_2) = type(i_2') = 2$. Also, $x_{i_1 j_1} = x_{i_1 j_2} = y_{i_1} = 0.3$, $x_{i_1' j_1} = x_{i_1' j_2'} = y_{i_1'} = 0.2$, $x_{i_2 j_2} = y_{i_2} = 0.7$ and $x_{i_2' j_2'} = y_{i_2'} = 0.8$. Note that $\mathcal{Y}_1(\mathcal{S}_1(j_1)) = 0.5$. Therefore, we cannot decrease only one of $i_1$ and $i_1'$ without increasing the other, as some other $type(1)$ demand may be using these facilities to the sum extent of 0.5. Also, since $\mathcal{Y}(\mathcal{S}(j_2)) = 1$ and $\mathcal{Y}(\mathcal{S}(j_2')) = 1$, the pairs of facilities $i_1$, $i_2$ and $i_1'$, $i_2'$ must occur together in any chain that is formed. Consider the chain $\mathcal{C} = \{i_1, i_2, i_2', i_1'\}$. Then,

$$\Delta(\mathcal{D}', \mathcal{C} + \delta) = -\Delta(\mathcal{D}', \mathcal{C} - \delta) = \delta \cdot (d_{j_1} \cdot (c_{i_1 j_1} - c_{i_1' j_1}) + d_{j_2} \cdot (c_{i_1 j_2} - c_{i_2 j_2}) + d_{j_2'} \cdot (c_{i_2' j_2'} - c_{i_1' j_2'})).$$

Therefore $\mathcal{C}$ is the required chain along which we can perform adjustments.

We discover the chain in parts. Any *level 2* scenario for which the sum of facilities of some type is not half-integral can form a part of such a chain. For a *level 1* scenario, we concatenate chains of some *level 2* scenarios to form a longer chain. Similarly, we concatenate such chains from *level 1* scenarios to form a chain that satisfies the required properties. Note that though individual chains may violate constraint (6.21) of the LP, the final chain that we form by concatenating these chains will not violate this constraint.

**Adjusting a *level 2* scenario with non-$\frac{1}{2}$-integral $type(1)$ facilities**

The main result of this section is as follows:

**Lemma 6.12** *Let $\mathcal{S}_2(j)$ be a* level 2 *scenario such that $\mathcal{Y}_1(\mathcal{S}_2(j))$ is not $\frac{1}{2}$-integral. Then there exists a chain of facilities $\mathcal{C}$ such that $\Delta(\mathcal{D}(\mathcal{S}_2(j)), \mathcal{C} + \delta) = -\Delta(\mathcal{D}(\mathcal{S}_2(j)), \mathcal{C} - \delta)$. Moreover $\mathcal{C}$ is of one of the following forms:*

1. *$< i_1 >$. Moreover, in this case $\bar{y}_{i_1} \neq 1$.*
2. *$< i_1, i_2 >$. Moreover, in this case $\mathcal{Y}_2(\mathcal{S}_2(j))$ is not $\frac{1}{2}$-integral and $\bar{y}_{i_1} \neq 1$.*

*Proof.* First, consider the case when $j \in \mathcal{D}_1'$. Then note that $\mathcal{D}(\mathcal{S}_2(j)) = \phi$. Consider the facility $i \in \mathcal{F}_1' \cap \mathcal{S}_2(j)$. By virtue of the fact that $\mathcal{Y}_1(\mathcal{S}_2(j))$ is not $\frac{1}{2}$-integral, we know that $\bar{y}_i \neq 1$. Then, $\mathcal{C} =< i >$ is the required chain since $\Delta(\mathcal{D}(\mathcal{S}(j_2)), \mathcal{C} + \delta) = -\Delta(\mathcal{D}(\mathcal{S}(j_2)), \mathcal{C} - \delta) = 0$.

Now consider the case when $j \in \mathcal{D}_2'$. Note that $\mathcal{S}_2(j) = \mathcal{S}(j)$. Also, note that $\mathcal{D}(\mathcal{S}(j)) = \{j\}$. Thus we are just interested in analyzing the change in the cost for demand $j$. Consider the following cases:

- *There are two $type(1)$ facilities in $\mathcal{S}(j)$ as shown in Figure 6.4 (a):*
  Consider $\mathcal{C} =< i'_1 >$. Then by Lemma 6.11, $\Delta(j, \mathcal{C} + \delta) = -\Delta(j, \mathcal{C} - \delta) = \delta \cdot d_j \cdot (c_{i'_1 j} - c_{i_1 j})$ and thus $< i'_1 >$ is the required chain.

- *There is only one $type(1)$ facility in $\mathcal{S}(j)$ as shown in Figure 6.4 (b):*
  By virtue of the fact that $\mathcal{Y}_1(\mathcal{S}_2(j))$ is not $\frac{1}{2}$-integral, we know that $\bar{y}_{i_1} \neq 1$.

  Note that in this case, by Lemma 6.11, there is at most one $type(2)$ facility in $\mathcal{S}(j)$.

  Consider the following sub-cases:

  - $\bar{x}_{i_1 j} < \bar{y}_{i_1}$ *:*
    Then $\mathcal{C} =< i_1 >$ is the required chain as $\Delta(j, \mathcal{C} + \delta) = -\Delta(j, \mathcal{C} - \delta) = 0$.

  - $\bar{x}_{i_1 j} = \bar{y}_{i_1}$ *and $\mathcal{Y}(\mathcal{S}(j))$ is $\frac{1}{2}$-integral :*
    In this case $\bar{y}_{i_1} + \bar{y}_{i_2}$ is $\frac{1}{2}$-integral. But, $\bar{y}_{i_1}$ is not $\frac{1}{2}$-integral implying that $\bar{y}_{i_2}$ is also not $\frac{1}{2}$-integral. Also, $\bar{x}_{i_1 j} > 0$ and $\bar{x}_{i_2 j} > 0$. Then $\mathcal{C} =< i_1, i_2 >$ is the required chain since $\Delta(j, \mathcal{C} + \delta) = -\Delta(j, \mathcal{C} - \delta) = \delta \cdot d_j \cdot (c_{i_1 j} - c_{i_2 j})$.

  - $\bar{x}_{i_1 j} = \bar{y}_{i_1}$ *and $\mathcal{Y}(\mathcal{S}(j))$ is not $\frac{1}{2}$-integral :*
    If there is a $type(2)$ facility, say $i_2$, in $\mathcal{S}(j)$, then $\bar{x}_{i_2 j} = \bar{y}_{i_2}$. Therefore $\frac{1}{2} < \mathcal{Y}(\mathcal{S}(j)) < 1$. Then $\mathcal{C} =< i_1 >$ is the required chain since $\Delta(j, \mathcal{C} + \delta) = -\Delta(j, \mathcal{C} - \delta) = \delta \cdot d_j \cdot (c_{i_1 j} - \bar{c}_j)$.

∎

**Adjusting a *level 2* scenario with non-$\frac{1}{2}$-integral $type(2)$ facilities**

The main result of this section is as follows:

**Lemma 6.13** *Let $\mathcal{S}_2(j)$ be a* level 2 *scenario such that $\mathcal{Y}_2(\mathcal{S}_2(j))$ is not $\frac{1}{2}$-integral. Then there exists a chain of facilities $\mathcal{C}$ such that $\Delta(\mathcal{D}(\mathcal{S}_2(j)), \mathcal{C} + \delta) = -\Delta(\mathcal{D}(\mathcal{S}_2(j)), \mathcal{C} - \delta)$. Moreover $\mathcal{C}$ is of one of the following forms:*

1. *$< i_2 >$, where $i_2 \in \mathcal{D}_2$.*

2. $< i_2, i_1 >$, *where* $i_1 \in \mathcal{D}_1, i_2 \in \mathcal{D}_2$. *Moreover, in this case* $\mathcal{Y}_1(\mathcal{S}_2(j))$ *is not* $\frac{1}{2}$*-integral and* $\bar{y}_{i_1} \neq 1$.

*Proof.* Clearly $j \in \mathcal{D}_2'$ because $\mathcal{F}_2' \cap \mathcal{S}_2(j) = \phi$ for $j \in \mathcal{D}_1'$ and therefore $\mathcal{Y}_2(\mathcal{S}_2(j))$ cannot be non-$\frac{1}{2}$-integral. Note that $\mathcal{S}_2(j) = \mathcal{S}(j)$. Also, Note that $\mathcal{D}(\mathcal{S}(j)) = \{j\}$. Thus we are just interested in analyzing the change in the cost for demand $j$.

Consider the following cases:

- *There are two* $type(1)$ *facilities in* $\mathcal{S}(j)$ *as shown in Figure 6.4 (a):*
  Consider $\mathcal{C} =< i_2 >$. Then applying Lemma 6.11, there exists a $type(1)$ demand, say $j_1 \in \mathcal{CC}_{G_1}(j)$ and we get that $\Delta(j, \mathcal{C} + \delta) = -\Delta(j, \mathcal{C} - \delta) = \delta \cdot d_j \cdot (c_{i_2 j} - c_{i_1 j})$.

- *There is at most one* $type(1)$ *facility in* $\mathcal{S}(j)$ *as shown in Figure 6.4 (b):*
  There is only one $type(2)$ facility in $\mathcal{S}(j)$, i.e., $i_2$.

  Consider the following sub-cases:

  – *There is a* $type(1)$ *facility, say* $i_1$, *in* $\mathcal{S}(j)$ *and* $\bar{x}_{i_1 j} < \bar{y}_{i_1}$ *:*
  Clearly, in this case, $\bar{x}_{i_1 j} > 0$. Then $\mathcal{C} =< i_2 >$ is the required chain since $\Delta(j, \mathcal{C} + \delta) = -\Delta(j, \mathcal{C} - \delta) = \delta \cdot d_j \cdot (c_{i_2 j} - c_{i_1 j})$.

  – *There is a* $type(1)$ *facility, say* $i_1$, *in* $\mathcal{S}(j)$ *and* $\bar{x}_{i_1 j} = \bar{y}_{i_1}$ *and* $\mathcal{Y}(\mathcal{S}_2(j))$ *is* $\frac{1}{2}$*-integral :*
  In this case $\bar{y}_{i_1} + \bar{y}_{i_2}$ is $\frac{1}{2}$-integral. Moreover, we know that $\bar{y}_{i_2}$ is not $\frac{1}{2}$-integral implying that $\bar{y}_{i_1}$ is also not $\frac{1}{2}$-integral. Also, $\bar{x}_{i_1 j} > 0$ and $\bar{x}_{i_2 j} > 0$. Then, $\mathcal{C} =< i_2, i_1 >$ is the required chain since $\Delta(j, \mathcal{C} + \delta) = -\Delta(j, \mathcal{C} - \delta) = \delta \cdot d_j \cdot (c_{i_2 j} - c_{i_1 j})$.

  – $\mathcal{Y}(\mathcal{S}_2(j))$ *is not* $\frac{1}{2}$*-integral and either* $\bar{x}_{i_1 j} = \bar{y}_{i_1}$ *or there is no* $type(1)$ *facility in* $\mathcal{S}(j)$ *:*
  In this case $\frac{1}{2} < \mathcal{Y}(\mathcal{S}_2(j)) < 1$. Some fraction of the demand $j$ is assigned to facilities in the scenario of its nearest assignable demand. Then $\mathcal{C} =< i_2 >$ is the required chain since $\Delta(j, \mathcal{C} + \delta) = -\Delta(j, \mathcal{C} - \delta) = \delta \cdot d_j \cdot (c_{i_2 j} - \bar{c}_j)$.

∎

**Adjusting a *level 1* scenario with non-$\frac{1}{2}$-integral $type(1)$ facilities**

The main result of this section is as follows:

**Lemma 6.14** *Let $\mathcal{S}(j)$ be a* level 1 *scenario such that $\mathcal{Y}_1(\mathcal{S}_1(j))$ is not $\frac{1}{2}$-integral. Then there exists a chain of facilities $\mathcal{C}$ such that $\Delta(\mathcal{D}(\mathcal{S}_1(j)), \mathcal{C}+\delta) = -\Delta(\mathcal{D}(\mathcal{S}_1(j)), \mathcal{C}-\delta)$. Moreover $\mathcal{C}$ is of one of the following forms:*

1. $< i_1 >$.
2. $< i_1, i_2 >$. *Moreover, in this case $\mathcal{Y}_2(\mathcal{S}_1(j))$ is not $\frac{1}{2}$-integral.*
3. $< i_1, i_2, i_2' >$. *Moreover, in this case $\mathcal{Y}_2(\mathcal{S}_1(j))$ is $\frac{1}{2}$-integral.*
4. $< i_1, i_1', i_2, i_2', i_1'' >$. *In this case also, $\mathcal{Y}_2(\mathcal{S}_1(j))$ is $\frac{1}{2}$-integral.*

*Proof.* The proof follows from Lemmas 6.15, 6.16 and 6.17. ∎

**Lemma 6.15** *Let $j_2$ be a $type(2)$ demand such that there is no $type(1)$ demand in $\mathcal{CC}_{G_1}(j_2)$ and $\mathcal{Y}_1(\mathcal{S}(j_2))$ is not $\frac{1}{2}$-integral. Then there exists a chain of facilities $\mathcal{C}$ such that $\Delta(\mathcal{D}(\mathcal{S}(j_2)), \mathcal{C}+\delta) = -\Delta(\mathcal{D}(\mathcal{S}(j_2)), \mathcal{C}-\delta)$. Moreover $\mathcal{C}$ is of one of the following forms:*

1. $< i_1 >$.
2. $< i_1, i_2 >$. *Moreover, in this case $\mathcal{Y}_2(\mathcal{S}(j_2))$ is not $\frac{1}{2}$-integral.*

*Proof.* Note that the *level 1* and *level 2* scenarios to which $j_2$ belongs are the same, i.e., $\mathcal{S}_1(j_2) = \mathcal{S}_2(j_2) = \mathcal{S}(j_2)$. Therefore the proof follows from Lemma 6.12. ∎

**Lemma 6.16** *Let $j_1$ be a $type(1)$ demand such that $\mathcal{Y}_1(\mathcal{S}_1(j_1)) > 1$ and not $\frac{1}{2}$-integral. Then there exists a chain of facilities $\mathcal{C}$ such that $\Delta(\mathcal{D}(\mathcal{S}_1(j_1)), \mathcal{C}+\delta) = -\Delta(\mathcal{D}(\mathcal{S}_1(j_1)), \mathcal{C}-\delta)$. Moreover $\mathcal{C}$ is of one of the following forms:*

1. $< i_1 >$.
2. $< i_1, i_2 >$. *Moreover, in this case $\mathcal{Y}_2(\mathcal{S}_1(j_1))$ is not $\frac{1}{2}$-integral.*
3. $< i_1, i_2, i_2' >$. *Moreover, in this case $\mathcal{Y}_2(\mathcal{S}_1(j_1))$ is $\frac{1}{2}$-integral.*
4. $< i_1, i_1', i_2, i_2', i_1'' >$. *In this case also, $\mathcal{Y}_2(\mathcal{S}_1(j_1))$ is $\frac{1}{2}$-integral.*

*Proof.* Note that $\mathcal{S}_1(j_1) = \mathcal{S}(j_1)$. Suppose there exists a *level 2* scenario, say $\mathcal{S}_2(j_2)$, such that $j_2 \in \mathcal{CC}_{G_1}(j_1)$ and contains 2 $type(1)$ facilities. Clearly $type(j_2) = 2$. Let $i$ and $i'$ be the two facilities. Then, by Lemma 6.11, $\bar{x}_{ij_1} = \bar{y}_i = 1$, $\bar{x}_{i'j_1} = 0$ and $0 < \bar{x}_{ij_2} < \bar{y}_i$. Then it can be easily seen that $< i' >$ is the required chain and we are done.

Otherwise, for every *level 2* scenario $\mathcal{S}_2(j_2)$ where $j_2 \in \mathcal{CC}_{G_1}(j_1)$, there is only at most one facility of each type. We can arrange the open $type(1)$ facilities in $\mathcal{S}_1(j_1)$ in increasing order of distance from $j_1$, say $i_1, i_2, \ldots, i_t, \ldots, i_w$, such that $\bar{x}_{ij_1} = \bar{y}_i$ for $i = i_1, \ldots, i_{t-1}$, $\bar{x}_{ij_1} \leq \bar{y}_i$ for $i = i_t$ and $\bar{x}_{ij_1} = 0$ for $i = i_{t+1}, \ldots, i_w$. Let $\mathcal{S}_2(j_{2,s})$ be the scenario containing $i_s$ for all $s = 1, \ldots, w$. Note that if $\mathcal{S}_2(j_1)$ contains a $type(1)$ facility, then this facility must be closest to $j_1$, i.e., it must be that $j_{2,1} = j_1$.

Consider the following cases:

- $\bar{x}_{i_t j_1} < \bar{y}_{i_t}$ *:*
  Let $i_s$ be any facility in $\mathcal{S}_1(j_1) \cap \mathcal{F}'_1$. Then note that $\Delta(j_1, \{i_s\} + \delta) = -\Delta(j_1, \{i_s\} - \delta) = \delta \cdot d_{j_1} \cdot (c_{i_s j_1} - c_{i_t j_1})$ if $s < t$ and 0 otherwise.

  Now, there must exist some scenario, say $\mathcal{S}_2(j_{2,s})$, such that $j_{2,s} \in \mathcal{CC}_{G_1}(j_1)$ and $\mathcal{Y}_1(\mathcal{S}_2(j_{2,s}))$ is not $\frac{1}{2}$-integral. By Lemma 6.12, there exists a chain $\mathcal{C}$ of facilities such that $\Delta(\mathcal{D}(\mathcal{S}_2(j_{2,s})), \mathcal{C} + \delta) = -\Delta(\mathcal{D}(\mathcal{S}_2(j_{2,s})), \mathcal{C} - \delta)$.

  Consider the following subcases:

  - *If $\mathcal{C}$ is of the form $< i_s >$, where $i_s \in \mathcal{F}'_1$ :*
    It can be easily verified that $< i_s >$ is the required chain and we are done.
  - *If $\mathcal{C}$ is of the form $< i_s, i'_s >$, where $i_s \in \mathcal{F}'_1$, $i'_s \in \mathcal{F}'_2$ and $\mathcal{Y}_2(\mathcal{S}(j_1))$ is not $\frac{1}{2}$-integral:*
    It can be verified that $< i_s, i'_s >$ is the required chain and we are done.
  - *If $\mathcal{C}$ is of the form $< i_s, i'_s >$, where $i_s \in \mathcal{F}'_1$, $i'_s \in \mathcal{F}'_2$ and $\mathcal{Y}_2(\mathcal{S}(j_1))$ is $\frac{1}{2}$-integral:*
    Then, there must be at least one other *level 2* scenario, say $\mathcal{S}_2(j_{2,r})$, such that $j_{2,r} \in \mathcal{CC}_{G_1}(j_1)$ and $\mathcal{Y}_2(\mathcal{S}_2(j_{2,r}))$ is also not $\frac{1}{2}$-integral. By Lemma 6.13, there exists a chain $\mathcal{C}'$ of facilities in $\mathcal{S}_2(j_{2,r})$ such that $\Delta(\mathcal{D}(\mathcal{S}_2(j_{2,r})), \mathcal{C}' + \delta) = -\Delta(\mathcal{D}(\mathcal{S}_2(j_{2,r})), \mathcal{C}' - \delta)$.
    - *If $\mathcal{C}' =< i'_r >$, where $i'_r \in \mathcal{F}'_2$ :*
      It can be easily verified that $< i_s, i'_s, i'_r >$ is the required chain and we are done.
    - *If $\mathcal{C}' =< i'_r, i_r >$, where $i_r \in \mathcal{F}'_1$ and $i'_r \in \mathcal{F}'_2$ :*
      Consider the chain $\mathcal{C}'' =< i_s, i'_s, i'_r, i_r >$.
      If $s, r < t$, then $\Delta(j_1, \mathcal{C}'' + \delta) = -\Delta(j_1, \mathcal{C}'' - \delta) = \delta \cdot d_{j_1} \cdot (c_{i_s j_1} - c_{i_r j_1})$.

If $s, r \geq t$, then $\Delta(j_1, \mathcal{C}'' + \delta) = -\Delta(j_1, \mathcal{C}'' - \delta) = 0$.
If $s < t$, $r \geq t$, then $\Delta(j_1, \mathcal{C}'' + \delta) = -\Delta(j_1, \mathcal{C}'' - \delta) = \delta \cdot d_{j_1} \cdot (c_{i_s j_1} - c_{i_t j_1})$.
If $s \geq t$, $r < t$, then $\Delta(j_1, \mathcal{C}'' + \delta) = -\Delta(j_1, \mathcal{C}'' - \delta) = \delta \cdot d_{j_1} \cdot (c_{i_t j_1} - c_{i_r j_1})$.
Therefore in all cases $\Delta(\mathcal{D}(\mathcal{S}(j_1)), \mathcal{C}'' + \delta) = -\Delta(\mathcal{D}(\mathcal{S}(j_1)), \mathcal{C}'' - \delta)$. This implies that the LP solution is a non-optimal or non-vertex solution – contradiction. Thus, this case is not possible.

- $\bar{x}_{i_t j_1} = \bar{y}_{i_t}$ *:*
  Note that, in this case $\sum_{v \leq t} \bar{x}_{i_v j_1} = \sum_{v \leq t} \bar{y}_{i_v} = 1$. Therefore, there is some scenario, say $\mathcal{S}_2(j_{2,s})$, such that $s > t$, $j_{2,s} \in \mathcal{CC}_{G_1}(j_1)$ and $\mathcal{Y}_1(\mathcal{S}_2(j_{2,s}))$ is not $\frac{1}{2}$-integral. By Lemma 6.12, there exists a chain $\mathcal{C}$ of facilities s.t. $\Delta(\mathcal{D}(\mathcal{S}_2(j_{2,s})), \mathcal{C} + \delta) = -\Delta(\mathcal{D}(\mathcal{S}_2(j_{2,s})), \mathcal{C} - \delta)$.

  – *If $\mathcal{C}$ is of the form $< i_s >$, where $i_s \in \mathcal{F}_1'$ :*
    It can be easily seen that $< i_s >$ is the required chain and we are done.

  – *If $\mathcal{C}$ is of the form $< i_s, i_s' >$, where $i_s \in \mathcal{F}_1'$ and $i_s' \in \mathcal{F}_2'$ :*
    If $\mathcal{Y}_2(\mathcal{S}(j_1))$ is not $\frac{1}{2}$-integral, then it can be easily verified that $< i_s, i_s' >$ is the required chain and we are done.

    Otherwise, $\mathcal{Y}_2(\mathcal{S}(j_1))$ is $\frac{1}{2}$-integral. Therefore, there must be another scenario, $\mathcal{S}_2(j_{2,r})$, such that $j_{2,r} \in \mathcal{CC}_{G_1}(j_1)$ and $\mathcal{Y}_2(\mathcal{S}_2(j_{2,r}))$ is not $\frac{1}{2}$-integral. Moreover, by Lemma 6.11, we know that $type(j_{2,r}) \neq 1$.
    By Lemma 6.13, there exists a chain $\mathcal{C}_r$ of facilities such that $\Delta(\mathcal{D}(\mathcal{S}_2(j_{2,r})), \mathcal{C}_r + \delta) = -\Delta(\mathcal{D}(\mathcal{S}_2(j_{2,r})), \mathcal{C}_r - \delta)$.
    Now, consider the following subcases:

    * $r > t$ *:*
      · *If $\mathcal{C}_r$ is of the form $< i_r' >$, where $i_r' \in \mathcal{F}_2'$ :*
        It can be easily verified that $< i_s, i_s', i_r' >$ is the required chain and we are done.
      · *If $\mathcal{C}_r$ is of the form $< i_r', i_r >$, where $i_r \in \mathcal{F}_1'$ and $i_r' \in \mathcal{F}_2'$ :*
        Consider the chain $\mathcal{C}' =< i_s, i_s', i_r', i_r >$. Then, it can be verified that $\Delta(\mathcal{D}(\mathcal{S}(j_1)), \mathcal{C}' + \delta) = -\Delta(\mathcal{D}(\mathcal{S}(j_1)), \mathcal{C}' - \delta)$. This implies that the solu-

tion returned by the LP is a non-optimal or non-vertex solution – contradiction. Hence this case is not valid.

* $r \leq t$ :
  - *If $\mathcal{C}_r$ is of the form $< i'_r >$, where $i'_r \in \mathcal{F}'_2$ :*
    It can be easily seen that $< i_s, i'_s, i'_r >$ is the required chain and we are done.
  - *If $\mathcal{C}_r$ is of the form $< i'_r, i_r >$, where $i_r \in \mathcal{F}'_1$ and $i'_r \in \mathcal{F}'_2$ :*
    In this case, $\mathcal{Y}(\mathcal{S}_2(j_{2,r}))$ is $\frac{1}{2}$-integral and $\mathcal{Y}_1(\mathcal{S}_2(j_{2,r}))$ is not $\frac{1}{2}$-integral. But since $\sum_{v=1}^{t} \bar{y}_{i_v} = 1$ and $r \leq t$, there must exist another scenario, $\mathcal{S}_2(j_{2,q})$ such that $j_{2,q} \in \mathcal{CC}_{G_1}(j_1)$, $\mathcal{Y}_1(\mathcal{S}_2(j_{2,q}))$ is not $\frac{1}{2}$-integral and $q \leq t$.

    If $type(j_{2,q}) = 1$, then $q = 1$. In this case, let $i_q$ be the $type(1)$ facility in $\mathcal{S}_2(j_{2,q})$. Then, it can be verified that $< i_s, i'_s, i'_r, i_r, i_q >$ is the required chain and we are done.

    Otherwise $type(j_{2,q}) = 2$. By Lemma 6.12, there exists a chain $\mathcal{C}_q$ of facilities such that $\Delta(\mathcal{D}(\mathcal{S}_2(j_{2,q})), \mathcal{C}_q + \delta) = -\Delta(\mathcal{D}(\mathcal{S}_2(j_{2,q})), \mathcal{C}_q - \delta)$.
    1. *If $\mathcal{C}_q$ is of the form $< i_q >$, where $i_q \in \mathcal{F}'_1$ :*
       It can be easily verified that $< i_s, i'_s, i'_r, i_r, i_q >$ is the required chain and we are done.
    2. *If $\mathcal{C}_q$ is of the form $< i_q, i'_q >$, where $i_q \in \mathcal{F}'_1$ and $i'_q \in \mathcal{F}'_2$ :*
       Consider the chain $\mathcal{C}' =< i_r, i'_r, i'_q, i_q >$. Then, it can be verified that $\Delta(\mathcal{D}(\mathcal{S}(j_1)), \mathcal{C}' + \delta) = -\Delta(\mathcal{D}(\mathcal{S}(j_1)), \mathcal{C}' - \delta)$. This implies that the solution returned by the LP is a non-optimal or non-vertex solution – contradiction. Hence this case is not valid.

∎

**Lemma 6.17** *Let $j_1$ be a $type(1)$ demand such that $\frac{1}{2} < \mathcal{Y}_1(\mathcal{S}(j_1)) < 1$. Then there exists a chain of facilities $\mathcal{C}$ such that $\Delta(\mathcal{D}(\mathcal{S}_1(j_1)), \mathcal{C} + \delta) = -\Delta(\mathcal{D}(\mathcal{S}_1(j_1)), \mathcal{C} - \delta)$. Moreover $\mathcal{C}$ is of one of the following forms:*

1. $< i_1 >$.
2. $< i_1, i_2 >$. *Moreover, in this case $\mathcal{Y}_2(\mathcal{S}(j_1))$ is not $\frac{1}{2}$-integral.*

*3.* $< i_1, i_2, i_2' >$*. Moreover, in this case* $\mathcal{Y}_2(\mathcal{S}(j_1))$ *is* $\frac{1}{2}$*-integral.*

*Proof.* Some fraction of the demand $j_1$ (between $\frac{1}{2}$ and 1) is assigned to facilities in the scenario of its nearest assignable demand. Also the total sum of $type(1)$ facilities in the nearest assignable demand is at least $\frac{1}{2}$. Moreover, the distance from $j_1$ to all these facilities is same, say $\bar{c}_{j_1}$.

There exists some *level 2* scenario, say $\mathcal{S}_2(j_2)$, such that $j_2 \in \mathcal{CC}_{G_1}(j_1)$ and $\mathcal{Y}_1(\mathcal{S}_2(j_2))$ is not $\frac{1}{2}$-integral.

If $type(j_2) = 1$, then there is only one $type(1)$ facility in $\mathcal{S}_2(j_2)$. Let $i_1$ be this facility. Consider the chain $\mathcal{C} =< i_1 >$. Then, $\Delta(j_1, \mathcal{C} + \delta) = -\Delta(j_1, \mathcal{C} - \delta) = \delta \cdot d_{j_1} \cdot (c_{i_1 j_1} - \bar{c}_{j_1})$ and therefore $\Delta(\mathcal{D}(\mathcal{S}(j_1)), \mathcal{C} + \delta) = -\Delta(\mathcal{D}(\mathcal{S}(j_1)), \mathcal{C} - \delta)$. Hence $\mathcal{C}$ is the required chain and we are done.

Otherwise $type(j_2) = 2$. By Lemma 6.12, there exists a chain $\mathcal{C}$ of facilities such that $\Delta(\mathcal{D}(\mathcal{S}_2(j_2)), \mathcal{C} + \delta) = -\Delta(\mathcal{D}(\mathcal{S}_2(j_2)), \mathcal{C} - \delta)$.

Consider the following cases:

- *C is of the form* $< i_1 >$*, where* $i_1 \in \mathcal{F}_1'$ *:*
  Then it can be easily seen that $< i_1 >$ is the required chain and we are done.

- *C is of the form* $< i_1, i_2 >$*, where* $i_1 \in \mathcal{F}_1', i_2 \in \mathcal{F}_2'$ *:*
  In this case, $\mathcal{Y}_2(\mathcal{S}_2(j_2))$ is not $\frac{1}{2}$-integral.

  If $\mathcal{Y}_2(\mathcal{S}(j_1))$ is also not $\frac{1}{2}$-integral, then it can be verified that $< i_1, i_2 >$ is the required chain and we are done.

  Otherwise $\mathcal{Y}_2(\mathcal{S}(j_1))$ is $\frac{1}{2}$-integral. Then there must exist another scenario, say $\mathcal{S}_2(j_2')$, such that $j_2' \in \mathcal{CC}_{G_1}(j_1)$ and $\mathcal{Y}_2(\mathcal{S}_2(j_2'))$ is not $\frac{1}{2}$-integral. Note that $type(j_2') = 2$. Again, by Lemma 6.13, there exists a chain $\mathcal{C}'$ of facilities such that $\Delta(\mathcal{D}(\mathcal{S}_2(j_2')), \mathcal{C}' + \delta) = -\Delta(\mathcal{D}(\mathcal{S}_2(j_2')), \mathcal{C}' - \delta)$. Consider the following subcases:

  - $\mathcal{C}'$ *is of the form* $< i_2' >$*, such that* $i_2' \in \mathcal{F}_2'$ *:*
    It can be verified that $< i_1, i_2, i_2' >$ is the required chain and we are done.

  - $\mathcal{C}'$ *is of the form* $< i_2', i_1' >$*, such that* $i_1' \in \mathcal{F}_1', i_2' \in \mathcal{F}_2'$ *:*
    Consider the chain $\mathcal{C}'' = < i_1, i_2, i_2', i_1' >$. Then, it can be verified that $\Delta(\mathcal{D}(\mathcal{S}(j_1)), \mathcal{C}'' + \delta) = -\Delta(\mathcal{D}(\mathcal{S}(j_1)), \mathcal{C}'' - \delta)$. This implies that the solution returned by the LP is a non-optimal or non-vertex solution – contradiction. Hence this case is not valid.

This completes the proof of the Lemma.

∎

**Adjusting a *level 1* scenario with non-$\frac{1}{2}$-integral $type(2)$ facilities**

The main result of this section is as follows:

**Lemma 6.18** *Let $\mathcal{S}(j)$ be a* level 1 *scenario such that $\mathcal{Y}_2(\mathcal{S}_1(j))$ is not $\frac{1}{2}$-integral. Then there exists a chain of facilities $\mathcal{C}$ such that $\Delta(\mathcal{D}(\mathcal{S}_1(j)), \mathcal{C}+\delta) = -\Delta(\mathcal{D}(\mathcal{S}_1(j)), \mathcal{C}-\delta)$. Moreover $\mathcal{C}$ is of one of the following forms:*

1. $< i_2 >$.
2. $< i_2, i_1 >$. *Moreover, in this case $\mathcal{Y}_1(\mathcal{S}_1(j))$ is not $\frac{1}{2}$-integral.*
3. $< i_2, i_1, i_1' >$. *Moreover, in this case $\mathcal{Y}_1(\mathcal{S}_1(j))$ is $\frac{1}{2}$-integral.*
4. $< i_2, i_2', i_1, i_1', i_2'' >$. *In this case also, $\mathcal{Y}_1(\mathcal{S}_1(j))$ is $\frac{1}{2}$-integral.*

*Proof.* The proof follows from Lemmas 6.19, 6.20 and 6.21. ∎

**Lemma 6.19** *Let $j_2$ be a $type(2)$ demand such that there is no $type(1)$ demand in $\mathcal{CC}_{G_1}(j_2)$ and $\mathcal{Y}_2(\mathcal{S}(j_2))$ is not $\frac{1}{2}$-integral. Then there exists a chain of facilities $\mathcal{C}$ such that $\Delta(\mathcal{D}(\mathcal{S}(j_2)), \mathcal{C}+\delta) = -\Delta(\mathcal{D}(\mathcal{S}(j_2)), \mathcal{C}-\delta)$. Moreover $\mathcal{C}$ is of one of the following forms:*

1. $< i_2 >$.
2. $< i_2, i_1 >$. *Moreover, in this case $\mathcal{Y}_1(\mathcal{S}(j_2))$ is not $\frac{1}{2}$-integral.*

*Proof.* Note that the *level 1* and *level 2* scenarios to which $j_2$ belongs are the same, i.e., $\mathcal{S}_1(j_2) = \mathcal{S}_2(j_2) = \mathcal{S}(j_2)$. Therefore the proof follows from Lemma 6.13. ∎

**Lemma 6.20** *Let $j_1$ be a $type(1)$ demand such that $\mathcal{Y}_1(\mathcal{S}(j_1)) > 1$ and $\mathcal{Y}_2(\mathcal{S}(j_1))$ is not $\frac{1}{2}$-integral. Then there exists a chain of facilities $\mathcal{C}$ such that $\Delta(\mathcal{D}(\mathcal{S}(j_1)), \mathcal{C}+\delta) = -\Delta(\mathcal{D}(\mathcal{S}(j_1)), \mathcal{C}-\delta)$. Moreover $\mathcal{C}$ is of one of the following forms:*

1. $< i_2 >$.
2. $< i_2, i_1 >$. *Moreover, in this case $\mathcal{Y}_1(\mathcal{S}_1(j))$ is not $\frac{1}{2}$-integral.*

3. $< i_2, i_1, i'_1 >$. *Moreover, in this case* $\mathcal{Y}_1(\mathcal{S}_1(j))$ *is* $\frac{1}{2}$*-integral.*

4. $< i_2, i'_2, i_1, i'_1, i''_2 >$. *In this case also,* $\mathcal{Y}_1(\mathcal{S}_1(j))$ *is* $\frac{1}{2}$*-integral.*

*Proof.* We can arrange the open $type(1)$ facilities in $\mathcal{S}(j_1)$ in increasing order of distance from $j_1$, say $i_1, i_2, \ldots, i_t, \ldots, i_w$, such that $\bar{x}_{ij_1} = \bar{y}_i$ for $i = i_1, \ldots, i_{t-1}$, $\bar{x}_{ij_1} \leq \bar{y}_i$ for $i = i_t$ and $\bar{x}_{ij_1} = 0$ for $i = i_{t+1}, \ldots, i_w$. Let $\mathcal{S}_2(j_{2,s})$ be the scenario containing $i_s$ for all $s = 1, \ldots, w$.

Note that if there is a scenario with two $type(1)$ facilities, then this scenario is $\mathcal{S}_2(j_{2,1})$ and the $type(1)$ facilities of this scenario are $i_1$ and $i_2$. Moreover, $\bar{y}_{i_1} = 1$ and $\bar{y}_{i_v} = 0$ for all $v > 1$ and therefore $t = 1$. Also, in this case, $j_{2,1} = j_{2,2}$. We will always refer to this scenario as $j_{2,2}$ (not $j_{2,1}$). Therefore $i_1, i_2 \in \mathcal{S}_2(j_{2,2})$.

Consider the following cases:

- $\bar{x}_{i_t j_1} < \bar{y}_{i_t}$ *:*
  By the above argument, in this case there is no *level 2* scenario with two $type(1)$ facilities i.e., for every *level 2* scenario $\mathcal{S}_2(j_2)$ such that $j_2 \in \mathcal{CC}_{G_1}(j_1)$, there is only at most one facility of each type. Let $i_s$ be any facility in $\mathcal{S}(j_1) \cap \mathcal{F}'_1$. Then $\Delta(j_1, \{i_s\} + \delta) = -\Delta(j_1, \{i_s\} - \delta) = \delta \cdot d_{j_1} \cdot (c_{i_s j_1} - c_{i_t j_1})$ if $s < t$ and 0 otherwise.

  Now, there must exist some scenario, say $\mathcal{S}_2(j_{2,s})$, such that $j_2 \in \mathcal{CC}_{G_1}(j_1)$ and $\mathcal{Y}_2(\mathcal{S}_2(j_{2,s}))$ is not $\frac{1}{2}$-integral. Clearly $type(j_{2,s}) = 2$. By Lemma 6.13, there exists a chain $\mathcal{C}_s$ of facilities such that $\Delta(\mathcal{D}(\mathcal{S}_2(j_{2,s})), \mathcal{C}_s + \delta) = -\Delta(\mathcal{D}(\mathcal{S}_2(j_{2,s})), \mathcal{C}_s - \delta)$.

  - *If* $\mathcal{C}_s$ *is of the form* $< i'_s >$*, where* $i'_s \in \mathcal{F}'_2$ *:*
    Then, it can be easily seen that $< i_s >$ is the required chain and we are done.

  - *If* $\mathcal{C}_s$ *is of the form* $< i'_s, i_s >$*, where* $i_s \in \mathcal{F}'_1$*,* $i'_s \in \mathcal{F}'_2$ *and* $\mathcal{Y}_1(\mathcal{S}(j_1))$ *is not* $\frac{1}{2}$*-integral:*
    Then, it can be easily verified that $< i'_s, i_s >$ is the required chain and we are done.

  - *If* $\mathcal{C}_s$ *is of the form* $< i'_s, i_s >$*, where* $i_s \in \mathcal{F}'_1$*,* $i'_s \in \mathcal{F}'_2$ *and* $\mathcal{Y}_1(\mathcal{S}(j_1))$ *is* $\frac{1}{2}$*-integral:*
    Note that in this case $\mathcal{Y}_1(\mathcal{S}_2(j_{2,s}))$ is not $\frac{1}{2}$-integral. Therefore, there must be at least one other *level 2* scenario, say $\mathcal{S}_2(j_{2,r})$ such that $j_{2,r} \in \mathcal{CC}_{G_1}(j_1)$ and $\mathcal{Y}_1(\mathcal{S}_2(j_{2,r}))$ is

also not $\frac{1}{2}$-integral.

If $type(j_{2,r}) = 1$, then $r = 1$ and $i_1$ is the $type(1)$ facility in $\mathcal{S}_2(j_{2,r})$. Consider the chain $\mathcal{C} =< i'_s, i_s, i_1 >$. If $s < t$, then $\Delta(j_1, \mathcal{C} + \delta) = -\Delta(j_1, \mathcal{C} - \delta) = \delta \cdot d_{j_1} \cdot (c_{i_1 j_1} - c_{i_s j_1})$. If $s \geq t$, then $\Delta(j_1, \mathcal{C} + \delta) = -\Delta(j_1, \mathcal{C} - \delta) = \delta \cdot d_{j_1} \cdot (c_{i_1 j_1} - c_{i_t j_1})$. Hence $\mathcal{C}$ is the required chain and we are done.

Otherwise $type(j_{2,r}) = 2$. By Lemma 6.12, there exists a chain $\mathcal{C}_r$ of facilities in $\mathcal{S}_2(j_{2,r})$ such that $\Delta(\mathcal{D}(\mathcal{S}_2(j_{2,r})), \mathcal{C}_r + \delta) = -\Delta(\mathcal{D}(\mathcal{S}_2(j_{2,r})), \mathcal{C}_r - \delta)$.

* *If $\mathcal{C}_r =< i_r >$, where $i_r \in \mathcal{F}'_1$ :*
  It can be verified that $< i'_s, i_s, i_r >$ is the required chain by examining the cases when *(a)* $s, r < t$, *(b)* $s, r \geq t$, *(c)* $s < t$, $r \geq t$ and *(d)* $s \geq t, r < t$ separately.
* *If $\mathcal{C}_r =< i_r, i'_r >$, where $i_r \in \mathcal{F}'_1$ and $i'_r \in \mathcal{F}'_2$ :*
  Consider the chain $\mathcal{C} =< i'_s, i_s, i_r, i'_r >$.
  If $s, r < t$, then $\Delta(j_1, \mathcal{C} + \delta) = -\Delta(j_1, \mathcal{C} - \delta) = \delta \cdot d_{j_1} \cdot (c_{i_r j_1} - c_{i_s j_1})$.
  If $s, r \geq t$, then $\Delta(j_1, \mathcal{C} + \delta) = -\Delta(j_1, \mathcal{C} - \delta) = 0$.
  If $s < t$, $r \geq t$, then $\Delta(j_1, \mathcal{C} + \delta) = -\Delta(j_1, \mathcal{C} - \delta) = \delta \cdot d_{j_1} \cdot (c_{i_t j_1} - c_{i_s j_1})$.
  If $s \geq t$, $r < t$, then $\Delta(j_1, \mathcal{C} + \delta) = -\Delta(j_1, \mathcal{C} - \delta) = \delta \cdot d_{j_1} \cdot (c_{i_r j_1} - c_{i_t j_1})$.
  Therefore in all cases $\Delta(\mathcal{D}(\mathcal{S}(j_1)), \mathcal{C} + \delta) = -\Delta(\mathcal{D}(\mathcal{S}(j_1)), \mathcal{C} - \delta)$. This implies that the LP solution is a non-optimal or non-vertex solution – contradiction. Thus, this case is not possible.

- $\bar{x}_{i_t j_1} = \bar{y}_{i_t}$ *:*
  There must be some scenario, say $\mathcal{S}_2(j_{2,s})$ such that $j_{2,s} \in \mathcal{CC}_{G_1}(j_1)$, and $\mathcal{Y}_2(\mathcal{S}_2(j_{2,s}))$ is not $\frac{1}{2}$-integral. By Lemma 6.13, there exists a chain $\mathcal{C}_s$ of facilities such that $\Delta(\mathcal{D}(\mathcal{S}_2(j_{2,s})), \mathcal{C}_s + \delta) = -\Delta(\mathcal{D}(\mathcal{S}_2(j_{2,s})), \mathcal{C}_s - \delta)$.

  – *If $\mathcal{C}_s$ is of the form $< i'_s >$, where $i'_s \in \mathcal{F}'_2$ :*
    It can be easily seen that $< i'_s >$ is the required chain and we are done.
  – *If $\mathcal{C}_s$ is of the form $< i'_s, i_s >$, where $i_s \in \mathcal{F}'_1$ and $i'_s \in \mathcal{F}'_2$ :*

    Consider the following cases:

* $s > t$ *and* $\mathcal{Y}_1(\mathcal{S}(j_1))$ *is not* $\frac{1}{2}$*-integral:*
  It can be easily seen that $< i'_s, i_s >$ is the required chain and we are done.

* $s > t$ *and* $\mathcal{Y}_1(\mathcal{S}(j_1))$ *is* $\frac{1}{2}$*-integral :*
  Note that if $\mathcal{S}_2(j_{2,s})$ contains two $type(1)$ facilities, then $i_s = i_2$ ($i_s \neq i_1$) and therefore $\bar{y}_{i_s j_1} = 0$.
  Note that, in this case $\sum_{v \leq t} \bar{x}_{i_v j_1} = \sum_{v \leq t} \bar{y}_{i_v} = 1$.
  Therefore, there must exist another scenario, say $\mathcal{S}_2(j_{2,r})$ such that $j_2 \in \mathcal{CC}_{G_1}(j_1)$, $r > t$ and $\mathcal{Y}_1(\mathcal{S}_2(j_{2,r}))$ is not $\frac{1}{2}$-integral.

  If $type(j_{2,r}) = 1$, it can be verified that $< i'_s, i_s, i_r >$ is the required chain and we are done.

  Otherwise $type(j_{2,r}) = 2$. Then, by Lemma 6.12, there exists a chain $\mathcal{C}_r$ of facilities such that $\Delta(\mathcal{D}(\mathcal{S}_2(j_{2,r})), \mathcal{C}_r + \delta) = -\Delta(\mathcal{D}(\mathcal{S}_2(j_{2,r})), \mathcal{C}_r - \delta)$.

  - *If* $\mathcal{C}_r$ *is of the form* $< i_r >$*, where* $i_r \in \mathcal{F}'_1$ *:*
    It can be easily seen that $< i'_s, i_s, i_r >$ is the required chain and we are done.
  - *If* $\mathcal{C}_r$ *is of the form* $< i_r, i'_r >$*, where* $i_r \in \mathcal{F}'_1$ *and* $i'_r \in \mathcal{F}'_2$ *:*
    Consider the chain $\mathcal{C} = < i'_s, i_s, i_r, i'_r >$. Then, it can be verified that $\Delta(\mathcal{D}(\mathcal{S}(j_1)), \mathcal{C} + \delta) = -\Delta(\mathcal{D}(\mathcal{S}(j_1)), \mathcal{C} - \delta)$. This implies that the solution returned by the LP is a non-optimal or non-vertex solution – contradiction. Hence this case is not valid.

* $s \leq t$ *:*
  Then, since $\sum_{v \leq t} \bar{x}_{i_v j_1} = \sum_{v \leq t} \bar{y}_{i_v} = 1$ and $\mathcal{Y}_1(\mathcal{S}_2(j_{2,s}))$ is not $\frac{1}{2}$-integral, there must exist another scenario, say $\mathcal{S}_2(j_{2,r})$, such that $j_{2,r} \in \mathcal{CC}_{G_1}(j_1)$, $r \leq t$ and $\mathcal{Y}_1(\mathcal{S}_2(j_{2,r}))$ is not $\frac{1}{2}$-integral.
  Note that both $\mathcal{S}_2(j_{2,s})$ and $\mathcal{S}_2(j_{2,r})$ have only one $type(1)$ facility because for a scenario $\mathcal{S}_2(j_{2,v})$ having two $type(1)$ facilities $v = 2 > 1 = t$.
  By Lemma 6.12, there exists a chain $\mathcal{C}_r$ of facilities such that $\Delta(\mathcal{D}(\mathcal{S}_2(j_{2,r})), \mathcal{C}_r + \delta) = -\Delta(\mathcal{D}(\mathcal{S}_2(j_{2,r})), \mathcal{C}_r - \delta)$.

  - *If* $\mathcal{C}_r$ *is of the form* $< i_r >$*, where* $i_r \in \mathcal{F}'_1$ *:*
    It can be easily seen that $< i'_s, i_s, i_r >$ is the required chain and we are done.

· *If $\mathcal{C}_r$ is of the form $< i_r, i'_r >$, where $i_r \in \mathcal{F}'_1$ and $i'_r \in \mathcal{F}'_2$ :*
Consider the chain $\mathcal{C} = < i'_s, i_s, i_r, i'_r >$. Then, it can be verified that $\Delta(\mathcal{D}(\mathcal{S}(j_1)), \mathcal{C} + \delta) = -\Delta(\mathcal{D}(\mathcal{S}(j_1)), \mathcal{C} - \delta)$. This implies that the solution returned by the LP is a non-optimal or non-vertex solution – contradiction. Hence this case is not valid.

∎

**Lemma 6.21** *Let $j_1$ be a type(1) demand such that $\frac{1}{2} < \mathcal{Y}_2(\mathcal{S}_1(j_1)) < 1$. Then there exists a chain of facilities $\mathcal{C}$ such that $\Delta(\mathcal{D}(\mathcal{S}(j_1)), \mathcal{C} + \delta) = -\Delta(\mathcal{D}(\mathcal{S}(j_1)), \mathcal{C} - \delta)$. over $\mathcal{C}$ is of one of the following forms:*

1. $< i_2 >$.
2. $< i_2, i_1 >$. *Moreover, in this case $\mathcal{Y}_1(\mathcal{S}(j_1))$ is not $\frac{1}{2}$-integral.*
3. $< i_2, i_1, i'_1 >$. *Moreover, in this case $\mathcal{Y}_1(\mathcal{S}(j_1))$ is $\frac{1}{2}$-integral.*

*Proof.* Some fraction of the demand $j_1$ (between $\frac{1}{2}$ and 1) is assigned to facilities in the scenario of its nearest assignable demand. Also the total sum of $type(1)$ facilities in the nearest assignable demand is at least $\frac{1}{2}$. Moreover, the distance from $j_1$ to all these facilities is same, say $\bar{c}_{j_1}$.

There exists some *level 2* scenario, say $\mathcal{S}_2(j_2)$ where $j_2 \in \mathcal{CC}_{G_1}(j_1)$, such that $\mathcal{Y}_2(\mathcal{S}_2(j_2))$ is not $\frac{1}{2}$-integral.

By Lemma 6.13, there exists a chain $\mathcal{C}$ of facilities such that $\Delta(\mathcal{D}(\mathcal{S}_2(j_2)), \mathcal{C} + \delta) = -\Delta(\mathcal{D}(\mathcal{S}_2(j_2)), \mathcal{C} - \delta)$.

Consider the following cases:

- *C is of the form $< i_2 >$, where $i_2 \in \mathcal{F}'_2$ :*
  It can be easily seen that $< i_2 >$ is the required chain and we are done.
- *C is of the form $< i_2, i_1 >$, where $i_1 \in \mathcal{F}'_1, i_2 \in \mathcal{F}'_2$ and $\mathcal{Y}_1(\mathcal{S}(j_1))$ is not $\frac{1}{2}$-integral :*
  It can be verified that $< i_2, i_1 >$ is the required chain and we are done.

- *C is of the form* $< i_2, i_1 >$*, where* $i_1 \in \mathcal{F}'_1, i_2 \in \mathcal{F}'_2$ *and* $\mathcal{Y}_1(\mathcal{S}(j_1))$ *is* $\frac{1}{2}$*-integral :*
  Since $\mathcal{Y}_1(\mathcal{S}_2(j_2))$ is not $\frac{1}{2}$-integral, there must exist another scenario, say $\mathcal{S}_2(j'_2)$ such that $j'_2 \in \mathcal{CC}_{G_1}(j_1)$, and $\mathcal{Y}_1(\mathcal{S}_2(j'_2))$ is also not $\frac{1}{2}$-integral. Again, by Lemma 6.12, there exists a chain $\mathcal{C}'$ of facilities such that $\Delta(\mathcal{D}(\mathcal{S}_2(j'_2)), \mathcal{C}' + \delta) = -\Delta(\mathcal{D}(\mathcal{S}_2(j'_2)), \mathcal{C}' - \delta)$.

  Consider the following subcases:

  - $\mathcal{C}'$ *is of the form* $< i'_1 >$*, such that* $i'_1 \in \mathcal{F}'_1$ *:*
    It can be verified that $< i_2, i_1, i'_1 >$ is the required chain and we are done.
  - $\mathcal{C}'$ *is of the form* $< i'_1, i'_2 >$*, such that* $i'_1 \in \mathcal{F}'_1, i'_2 \in \mathcal{F}'_2$ *:*
    Consider the chain $\mathcal{C}'' = < i_2, i_1, i'_1, i'_2 >$. Then, it can be verified that $\Delta(\mathcal{D}(\mathcal{S}(j_1)), \mathcal{C}'' + \delta) = -\Delta(\mathcal{D}(\mathcal{S}(j_1)), \mathcal{C}'' - \delta)$. This implies that the solution returned by the LP is a non-optimal or non-vertex solution – contradiction. Hence this case is not valid.

∎

**Adjustment of Chains for $\frac{1}{2}$-integrality**

Note that all the chains that we consider have a special structure. The chains are in the form of pairs of facilities of the same type with possibly the first or last or both (first and last) facilities being of different types. More formally,

**Definition** *Special Chain:* Let $\mathcal{C} =< i_1, i_2, ..., i_t >$ be any chain such that:

- $t$ is odd and
  - either $type(i_s) = type(i_{s+1})$ for odd $s$, $s \neq t$ and $type(i_t) = 1$
  - or $type(i_s) = type(i_{s+1})$ for even $s$ and $type(i_1) = 1$

  We call such a chain a type 1 open chain since performing the operation $\mathcal{C}+\delta$ or $\mathcal{C}-\delta$ changes $\sum_{i \in \mathcal{C} \cap \mathcal{F}'_1} \bar{y}_i$ but not $\sum_{i \in \mathcal{C} \cap \mathcal{F}'_2} \bar{y}_i$. In the first case above, we say it is type 1 open from the right and in the second case we say that it is type 1 open from the left.

- $t$ is odd and
  - either $type(i_s) = type(i_{s+1})$ for odd $s$, $s \neq t$ and $type(i_t) = 2$

  - or $type(i_s) = type(i_{s+1})$ for even $s$ and $type(i_1) = 2$

  We call such a chain a type 2 open chain since performing the operation $\mathcal{C}+\delta$ or $\mathcal{C}-\delta$ changes $\sum_{i\in\mathcal{C}\cap\mathcal{F}'_2} \bar{y}_i$ but not $\sum_{i\in\mathcal{C}\cap\mathcal{F}'_1} \bar{y}_i$. In the first case above, we say it is type 2 open from the right and in the second case we say that it is type 2 open from the left.

- $t$ is even and

  - $type(i_s) = type(i_{s+1})$ for odd $s$
  - or $type(i_s) = type(i_{s+1})$ for even $s$ and $type(i_1) = type(i_t)$

  We call such a chain a closed chain since performing the operation $\mathcal{C} + \delta$ or $\mathcal{C} - \delta$ does not change $\sum_{i\in\mathcal{C}\cap\mathcal{F}'_1} \bar{y}_i$ or $\sum_{i\in\mathcal{C}\cap\mathcal{F}'_2} \bar{y}_i$.

- $t$ is even, $type(i_s) = type(i_{s+1})$ for even $s$, $s \neq t$ and moreover $type(i_1) \neq type(i_t)$. We call such a chain a type 1-2 open chain since performing the operation $\mathcal{C}+\delta$ or $\mathcal{C}-\delta$ changes both $\sum_{i\in\mathcal{C}\cap\mathcal{F}'_1} \bar{y}_i$ and $\sum_{i\in\mathcal{C}\cap\mathcal{F}'_2} \bar{y}_i$ by equal amounts but of opposite magnitude. If $type(i_1) = 1$, we say that it is type 1 open from the left and type 2 open from the right and if $type(i_1) = 2$, we say that it is type 2 open from the left and type 1 open from the right.

**Definition** *Reversal Operation on Chains:*
We define the reversal operation on a chain as follows: Let $\mathcal{C} =< i_1, ..i_t >$ be a chain of facilities. Then the reverse chain $\mathcal{R}(\mathcal{C})$ is the chain $< i_t, .., i_1 >$ where the facility order is reversed.

**Definition** *Concatenation Operation on Chains:*
We define the concatenation operation on a chain as follows: Let $\mathcal{C} =< i_1, ..i_t >$ and $\mathcal{C}' = < i'_1, ..i'_s >$ be two chains of facilities. Then the concatenated chain (denoted by $\mathcal{C}+\mathcal{C}'$) is the chain $< i_1, .., i_t, i'_1, .., i'_s >$.

**Definition** *Merge operation on Chains:*
Let $\mathcal{C} =< i_1, i_2, ..., i_t >$ and $\mathcal{C}' =< i'_1, i'_2, ..., i'_s >$ be two special chains. We define merge operation (denoted $\mathcal{C} \cdot \mathcal{C}'$) on these as follows:

- If both are type $w$ open from the left, then $\mathcal{C} \cdot \mathcal{C}' = \mathcal{R}(\mathcal{C}) + \mathcal{C}'$
- If both are type $w$ open from the right, then $\mathcal{C} \cdot \mathcal{C}' = \mathcal{C} + \mathcal{R}(\mathcal{C}')$
- If $\mathcal{C}$ is type $w$ open from the left and $\mathcal{C}'$ is type $w$ open from the right then $\mathcal{C} \cdot \mathcal{C}' = \mathcal{C}' + \mathcal{C}$

- If $\mathcal{C}$ is type $w$ open from the right and $\mathcal{C}'$ is type $w$ open from the left then $\mathcal{C} \cdot \mathcal{C}' = \mathcal{C} + \mathcal{C}'$

When multiple conditions are satisfied, for instance, when we merge two type 1-2 open chains, then we apply the first applicable condition in the above order.

Note that the chain obtained by merging special chains is also a special chain.

**Theorem 6.22** *The solution to the LP specified in Section 6.2.4 is half-integral.*

*Proof.* For simplicity assume that $\sum_{i \in \mathcal{F}'_1} \bar{y}_i$ and $\sum_{i \in \mathcal{F}'_2} \bar{y}_i$ are both integral. We handle the case when this is not so later.

We first show that $\mathcal{Y}_1(\mathcal{S})$ and $\mathcal{Y}_2(\mathcal{S})$ are $\frac{1}{2}$-integral for all *level 1* scenarios $\mathcal{S}$. We then prove that within any *level 1* scenario for which this is true $\mathcal{Y}_1(\mathcal{S}')$ and $\mathcal{Y}_2(\mathcal{S}')$ are $\frac{1}{2}$-integral for all *level 2* scenarios $\mathcal{S}'$ within that scenario.

First consider all the *level 1* scenarios for which $\mathcal{Y}_1(\mathcal{S})$ is not $\frac{1}{2}$-integral. Let $\mathcal{S}_1, \mathcal{S}_2, .., \mathcal{S}_v$ be these scenarios. We will prove that $v = 0$. Suppose otherwise. Then $v \geq 2$ since the total sum of all $type(1)$ facilities is integral. Let $\mathcal{C}_1, \mathcal{C}_2, .., \mathcal{C}_v$ be the special chains returned for these scenarios when Lemma 6.14 is invoked on these scenarios.

Note that all these special chains must be either type 1 open or type 1-2 open.

We will show that we can always combine these chains to form a closed chain.

If there are two chains, say $\mathcal{C}_x$ and $\mathcal{C}_y$ that are type 1 open then $\mathcal{C}_x \cdot \mathcal{C}_y$ is a closed chain.

Similarly, if there are two chains, say $\mathcal{C}_x$ and $\mathcal{C}_y$ that are type 1-2 open then $\mathcal{C}_x \cdot \mathcal{C}_y$ is a closed chain.

Therefore the only case left is that there are only 2 chains such that one of them, say $\mathcal{C}_x$, is type 1 open and the other, say $\mathcal{C}_y$ is type 1-2 open. Then $\mathcal{Y}_2(\mathcal{S}_y)$ is not $\frac{1}{2}$-integral. Therefore, there must be another *level 1* scenario, say $\mathcal{S}'$ such that $\mathcal{Y}_2(\mathcal{S}')$ is also not $\frac{1}{2}$-integral. Let $\mathcal{C}'$ be the special chain obtained by invoking Lemma 6.18 on this scenario. Then either $\mathcal{C}'$ is type 2 open or it is type 1-2 open. If it is type 2 open. Then $(\mathcal{C}_x \cdot \mathcal{C}_y) \cdot \mathcal{C}'$ is a closed chain. Otherwise $\mathcal{C}_y \cdot \mathcal{C}'$ is a closed chain.

If we still do not have a closed chain then $v = 0$.

Now consider all the *level 1* scenarios for which $\mathcal{Y}_2(\mathcal{S})$ is not $\frac{1}{2}$-integral. Let $\mathcal{S}'_1, \mathcal{S}'_2, .., \mathcal{S}'_w$ be these scenarios. We will prove that $w = 0$. Suppose otherwise. Then $w \geq 2$ since the total sum of all $type(1)$ facilities is integral. Let $\mathcal{C}'_1, \mathcal{C}'_2, .., \mathcal{C}'_w$ be the special chains returned for these scenarios when Lemma 6.18 is invoked on these scenarios.

Note that all these special chains must be either type 2 open or type 1-2 open. But they cannot be type 1-2 open because $\mathcal{Y}_1(\mathcal{S}')$ is $\frac{1}{2}$-integral for all the above *level 1* scenarios. Therefore they must all be type 2 open.

We will show that we can always combine these chains to form a closed chain.

If there are two chains, say $\mathcal{C}'_x$ and $\mathcal{C}'_y$ that are type 2 open then $\mathcal{C}'_x \cdot \mathcal{C}'_y$ is a closed chain.

Moreover, $w \neq 1$, because the total sum of all the $type(2)$ facilities is $\frac{1}{2}$-integral. Therefore there must always exist a closed chain or $w = 0$.

Therefore in all the cases, we obtain a closed chain. We call this closed chain $\mathcal{C}$. Now, note that

$$\Delta(\mathcal{D}', \mathcal{C} + \delta) = -\Delta(\mathcal{D}', \mathcal{C} - \delta).$$

This implies that the solution to the LP is either non-optimal or non-vertex – contradiction. Hence there is no *level 1* scenario $\mathcal{S}$ such that $\mathcal{Y}_1(\mathcal{S})$ is not $\frac{1}{2}$-integral or $\mathcal{Y}_2(\mathcal{S})$ is not $\frac{1}{2}$-integral.

We can show that within any *level 1* scenario for which $\mathcal{Y}_1(\mathcal{S})$ and $\mathcal{Y}_2(\mathcal{S})$ are $\frac{1}{2}$-integral $\mathcal{Y}_1(\mathcal{S}')$ and $\mathcal{Y}_2(\mathcal{S}')$ are $\frac{1}{2}$-integral for all *level 2* scenarios $\mathcal{S}'$ within that scenario. This follows by exactly the same arguments applied on the contained *level 2* scenarios and the chains returned by applying Lemmas 6.12 and 6.13 instead of Lemmas 6.14 and 6.18 wherever applicable.

Now, since within each *level 2* scenario there is only one $type(2)$ facility, it must be $\frac{1}{2}$-integral. Also there are at most two $type(1)$ facilities within any *level 2* scenario. If there is only one $type(1)$ facility, then it is $\frac{1}{2}$-integral. If on the other hand, there are two $type(1)$ facilities, then we know that for one of these facilities, say $i_1$, $\bar{y}_{i_1} = 1$ so that the other facility must be $\frac{1}{2}$-integral also.

When $\sum_{i \in \mathcal{F}'_t} \bar{y}_i$ is not integral for some type $t$, we simply add another *level 1* scenario very far away from other scenarios that contains a facility of that type, say $i_t$. We open $i_t$ to the least extent possible such that $\sum_{i \in \mathcal{F}'_t} \bar{y}_i$ becomes integral (note that this does not violate constraint (6.21) of the LP). Note that this facility is not used by any demand and therefore can be freely adjusted, i.e., this scenario can always return a chain $< i_t >$ without affecting the cost of the solution. We can then use this scenario in our adjustment of facilities across scenarios as described above.

Hence the LP solution is half-integral. ∎

### 6.2.6 Rounding to an integral solution

Let $\bar{\mathcal{F}}_j$ denote the facilities used by demand $j$. Let $\bar{C}^*_j = \sum_{i \in \bar{\mathcal{F}}_j} c_{ij} \bar{x}_{ij}$.

Note that since the solution is $\frac{1}{2}$-integral, $c_{ij} \leq 2\bar{C}^*_j$ for all $i \in \bar{\mathcal{F}}_j$. Therefore, it does not matter which of the facilities in $\bar{\mathcal{F}}_j$ a demand $j$ is assigned to in the integral solution as long as we can fully open that facility, as the cost only increases by at most a constant factor.

We now consolidate the demands again. We consider the demands in order of increasing $\bar{C}^*_j$ values. For each demand $j$, if there exists another demand, say $j'$ considered already (i.e., $\bar{C}^*_{j'} \leq \bar{C}^*_j$)

which is of the same or smaller type and $\bar{C}^*_j \geq c_{jj'}/8$, then we merge $j$ with $j'$. Note that this only increases the cost of the solution by a constant factor. It is easy to see that an integral solution to this modified problem instance can be used to obtain an integral solution to the problem instance before consolidation of demands by loosing at most a constant factor in our approximation.

The following observations follow from the consolidation of demands described above.

**Observation 6.7** *$\bar{\mathcal{F}}_j \cap \bar{\mathcal{F}}_{j'} = \phi$ for any two demands $j$, $j'$ of the same type.*

**Observation 6.8** *A $type(2)$ demand only shares facilities with at most one $type(1)$ demand.*

**Lemma 6.23** *By loosing at most a constant factor, the $\frac{1}{2}$-integral solution to the LP obtained in the previous section can be modified to an integral solution.*

*Proof.* We start by reassigning demands to use fully open facilities wherever possible.

Consider a $type(2)$ demand. If it uses two facilities and one of them is fully open, then we can simply reassign it completely to this facility. If it uses two $type(2)$ facilities, we can combine them to the one that is closer to the demand, making it integral. Note that these facilities are not used by any other demand. Similarly, if it uses two $type(1)$ facilities, we can combine them to the one that is closer to the $type(1)$ demand. Thus we are only left to deal with $type(2)$ demands that are $\frac{1}{2}$-assigned to a $type(1)$ facility and a $type(2)$ facility.

Now consider a $type(1)$ demand. If it is assigned to two $type(1)$ facilities and one of them is fully open, then we can simply assign it completely to this facility. If it uses two $type(1)$ facilities and at least one of them is not used by any other ($type(2)$) demand, then we can combine them to one facility. If exactly one of them is infact used by another $type(2)$ demand, then we fully open the corresponding $type(1)$ facility and we also reassign that $type(2)$ demand to fully utilize this facility. Now we are only left to deal with $type(1)$ demands that are $\frac{1}{2}$-assigned to two $type(1)$ facilities, where both these facilities are shared with $type(2)$ demands. Moreover each of these $type(2)$ demands is $\frac{1}{2}$-assigned to another $type(2)$ facility. We simply modify the solution to fully open the $type(1)$ facility that one of these $type(2)$ demands is assigned to and the $type(2)$ facility that the other $type(2)$ demand is assigned to.

Note that while performing the above adjustments, it is possible that we are left with some $\frac{1}{2}$-open facilities that are unused by any demand.

Now we are only left to deal with $type(2)$ demands that are $\frac{1}{2}$-assigned to a $type(1)$ facility and a $type(2)$ facility where the $type(1)$ facility is not shared with a $type(1)$ demand. For every pair

of such scenarios, we simply modify the solution to fully open the $type(1)$ facility for one of these scenarios and the $type(2)$ facility for the other scenario. We repeatedly apply this reassignment till there is at most one such scenario. If there is one such scenario, then since the sum of all the $type(1)$ (and $type(2)$) facilities is integral, it must be that there are unutilized facilities of each type. We borrow from one of these unused facility types and open the corresponding facility type fully in the scenario and reassign the demand to fully utilize this facility.

Note that any demand $j$, is reassigned at most once – to a fully open facility within a distance of at most $2\bar{C}_j^*$. Therefore, the total cost increases by at most a factor of 2. ∎

Putting together our discussions, we get the following algorithm. We solve the LP relaxation of the natural integer programming formulation specified in Section 6.1. We then consolidate demands in this fractional solution as specified in Section 6.2.1 and then modify the assignments in this fractional solution as specified in Section 6.2.3 to obtain a fractional solution that satisfies the Structure property. We then formulate a modified LP for this more structured instance as specified in Section 6.2.4. The solution to this modified LP is half-integral as shown in Theorem 6.22. We finally round it off to an integral solution as specified in Lemma 6.23. This integral solution can now be used to obtain an integral solution to the original LP by loosing at most a constant factor of approximation when separating out the demands that were consolidated together, leading to the following result.

**Theorem 6.24** *There exists an algorithm that solves the priority $k$-median problem with two priorities within a constant factor of approximation in polynomial time.*

The approximation ratio obtained using our algorithm is fairly large (in the region of a few hundreds). We have not attempted to minimize it. With more careful analysis, we believe that it can be lowered significantly.

## 6.3 Open Problems and Concluding Remarks

It would be interesting to know if there is a constant factor approximation algorithm or a large integrality gap for the given LP for the prioritized $k$-median problem when there are exactly 3 priorities.

The natural integer program formulation for the *facility location* problem is a variation of the integer program for the *$k$-median* problem that incorporates facility costs instead of bounding the number of facilities to open. Our integer program formulation for *$k$-median with priorities* can also

be easily modified to a formulation for the *facility location with priorities* problem. We can then similarly consolidate demands and facilities to create scenarios that form a nice nested structure and then formulate another linear problem for this seemingly simpler problem and show that it is unimodular, resulting in an integral solution. However, the approximation ratio obtained using this method is unlikely to be better than the approximation ratio obtained using previously known algorithms for this problem [59].

# Bibliography


[1] Sanjeev Arora. Polynomial time approximation schemes for euclidean tsp and other geometric problems. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pages 2–11, 1996.

[2] Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. Approximation schemes for euclidean -medians and related problems. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 106–113, 1998.

[3] Sunil Arya, Theocharis Malamatos, and David M. Mount. Space-efficient approximate voronoi diagrams. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 721–730, 2002.

[4] Sunil Arya and David M. Mount. Approximate nearest neighbor queries in fixed dimensions. In *Proceedings of the 4th annual ACM-SIAM symposium on Discrete algorithms*, pages 271–280, 1993.

[5] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM*, 45(6):891–923, 1998.

[6] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristic for k-median and facility location problems. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pages 21–29, 2001.

[7] D. Attali and J.D. Boissonnat. Complexity of the delaunay triangulation of points on a smooth surface. *http://www- sop.inria.fr/prisme/personnel/boissonnat/papers.html*, 2001.

[8] Franz Aurenhammer. Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345–405, 1991.

[9] Mihai Badoiu, Sariel Har-Peled, and Piotr Indyk. Approximate clustering via core-sets. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 250–257, 2002.

[10] Marshall Wayne Bern and David Eppstein. Approximation algorithms for geometric problems. In Dorit Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*, chapter 8, pages 296–345. PWS Publishing, 1996.

[11] Daniel Boley. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2(4):325–344, 1998.

[12] Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. Syntactic clustering of the web. *Computer Networks*, 29(8-13):1157–1166, 1997.

[13] Paul B. Callahan and S. Rao Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of the ACM*, 42(1):67–90, 1995.

[14] Timothy M. Chan. Approximate nearest neighbor queries revisited. *Discrete & Computational Geometry*, 20(3):359–373, 1998.

[15] Moses Charikar and Sudipto Guha. Improved combinatorial algorithms for the facility location and k-median problems. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pages 378–388, 1999.

[16] Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. A constant-factor approximation algorithm for the k-median problem. *Journal of Computer and System Sciences*, 65(1):129–149, 2002.

[17] Moses Charikar, Joseph Naor, and Baruch Schieber. Resource optimization in qos multicast routing of real-time multimedia. *IEEE/ACM Transactions on Networking*, 12(2):340–348, 2004.

[18] Bernard Chazelle, Ding Liu, and Avner Magen. Sublinear geometric algorithms. In Artur Czumaj, S. Muthu Muthukrishnan, Ronitt Rubinfeld, and Christian Sohler, editors, *Sublinear Algorithms*. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2006.

[19] Ke Chen. On -median clustering in high dimensions. In *Proceedings of the 17th annual ACM-SIAM symposium on Discrete algorithms*, pages 1177–1185. ACM Press, 2006.

[20] Kenneth L. Clarkson. An algorithm for approximate closest-point queries. In *Symposium on Computational Geometry*, pages 160–164, 1994.

[21] T.M. Cover and P.E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.

[22] Douglas R. Cutting, Jan O. Pedersen, David R. Karger, and John W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In Nicholas J. Belkin, Peter Ingwersen, and Annelise Mark Pejtersen, editors, *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 318–329. ACM, 1992.

[23] Wenceslas Fernandez de la Vega, Marek Karpinski, Claire Kenyon, and Yuval Rabani. Approximation schemes for clustering problems. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 50–58. ACM, 2003.

[24] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[25] Inderjit S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274, 2001.

[26] Petros Drineas, Alan M. Frieze, Ravi Kannan, Santosh Vempala, and V. Vinay. Clustering in large graphs and matrices. In *Proceedings of the 10th annual ACM-SIAM symposium on Discrete algorithms*, pages 291–299, 1999.

[27] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience Publication, 2000.

[28] C. A. Duncan. *Balanced aspect ratio trees.* PhD thesis, Johns Hopkins University, 1999.

[29] Jeff Erickson. Nice point sets can have nasty delaunay triangulations. *Discrete & Computational Geometry*, 30(1):109–132, 2003.

[30] Christos Faloutsos, Ron Barber, Myron Flickner, Jim Hafner, Wayne Niblack, Dragutin Petkovic, and William Equitz. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3(3/4):231–262, 1994.

[31] E. Fix and Joseph L. Hodges. Discriminatory analysis — nonparametric descrimination: consistency properties. Report 4, USAF School of Aviation Medicine, Randolph Field, Texas, 1951. Project No. 21-29-004.

[32] Allen Gersho and Robert M. Gray. *Vector quantization and signal compression*. Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[33] Sariel Har-Peled. A replacement for voronoi diagrams of near linear size. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science*, pages 94–103, 2001.

[34] Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 291–300. ACM, 2004.

[35] Sariel Har-Peled and Manor Mendel. Fast construction of nets in low dimensional metrics, and their applications. *SIAM Journal on Computing*, 35:1148, 2006.

[36] Sariel Har-Peled and Bardia Sadri. How fast is the k-means method? *Algorithmica*, 41(3):185–202, 2005.

[37] Mary Inaba, Naoki Katoh, and Hiroshi Imai. Applications of weighted voronoi diagrams and randomization to variance-based -clustering (extended abstract). In *Symposium on Computational Geometry*, pages 332–339, 1994.

[38] Piotr Indyk. *High Dimensional Computational Geometry*. PhD thesis, Stanford University, 1999.

[39] Piotr Indyk and Alexandr Andoni. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *to appear in the Proceedings of the 47th Symposium on Foundations of Computer Science*, 2006.

[40] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 604–613, 1998.

[41] Kamal Jain and Vijay V. Vazirani. Primal-dual approximation algorithms for metric facility location and k-median problems. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pages 2–13, 1999.

[42] Jon M. Kleinberg. Two algorithms for nearest-neighbor search in high dimensions. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 599–608, 1997.

[43] Stavros G. Kolliopoulos and Satish Rao. A nearly linear-time approximation scheme for the euclidean kappa-median problem. In Jaroslav Nesetril, editor, *7th Annual European Symposium on Algorithms*, volume 1643 of *Lecture Notes in Computer Science*, pages 378–389. Springer, 1999.

[44] Madhukar R. Korupolu, C. Greg Plaxton, and Rajmohan Rajaraman. Analysis of a local search heuristic for facility location problems. In *Proceedings of the 9th annual ACM-SIAM symposium on Discrete algorithms*, pages 1–10, 1998.

[45] Amit Kumar, Yogish Sabharwal, and Sandeep Sen. A simple linear time (1+ ) -approximation algorithm for k-means clustering in any dimensions. In *Proceedings of the 45th Annual Symposium on Foundations of Computer Science*, pages 454–462. IEEE Computer Society, 2004.

[46] Amit Kumar, Yogish Sabharwal, and Sandeep Sen. Linear time algorithms for clustering problems in any dimensions. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *32nd International Colloquium on Automata, Languages and Programming*, volume 3580 of *Lecture Notes in Computer Science*, pages 1374–1385. Springer, 2005.

[47] Eyal Kushilevitz, Rafail Ostrovsky, and Yuval Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 614–623, 1998.

[48] Bjornar Larsen and Chinatsu Aone. Fast and effective text mining using linear-time document clustering. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 16–22, 1999.

[49] Jyh-Han Lin and Jeffrey Scott Vitter. epsilon-approximations with minimum packing constraint violation (extended abstract). In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 771–782. ACM, 1992.

[50] Jirí Matousek. On approximate geometric k-clustering. *Discrete & Computational Geometry*, 24(1):61–84, 2000.

[51] R. Francis P. Mirchandani. *Discrete Location Theory, New York, NY*. Wiley, 1990.

[52] G. O. Wesolowsky R. F. Love, J. G. Morris. *Facilities Location: Models and Methods, New York, NY*. North-Holland, 1998.

[53] Yogish Sabharwal and Sandeep Sen. A linear time algorithm for approximate 2-means clustering. *Computational Geometry: Theory and Applications*, 32(2):159–172, 2005.

[54] Yogish Sabharwal, Nishant Sharma, and Sandeep Sen. Nearest neighbors search using point location in balls with applications to approximate voronoi decompositions. In Manindra Agrawal and Anil Seth, editors, *Foundations of Software Technology and Theoretical Computer Science*, volume 2556 of *Lecture Notes in Computer Science*, pages 311–323. Springer, 2002.

[55] Yogish Sabharwal, Nishant Sharma, and Sandeep Sen. Nearest neighbor searching using point location in balls with applications to approximate voronoi decompositions. In *Journal of Computer and System Sciences*, pages 955–977, 2006.

[56] Gerard Salton. *Automatic Text Processing – The Transformation, Analysis, and Retrieval of Information by Computer*. Addison–Wesley, 1989.

[57] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[58] Gerard Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[59] David B. Shmoys, Chaitanya Swamy, and Retsef Levi. Facility location with service installation costs. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1088–1097, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

[60] Michael J. Swain and Dana H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[61] C. J. van Rijsbergen. *Information Retrieval*. Butterworth, 1979.

[62] Andrew Chi-Chih Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal on Computing*, 11(4):721–736, 1982.

[63] Oren Zamir, Oren Etzioni, Omid Madani, and Richard M. Karp. Fast and intuitive clustering of web documents. In *Proceedings of the third ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 287–290, 1997.

[64] Ying Zhao and George Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Proceedings of the 2002 ACM CIKM International Conference on Information and Knowledge Management*, pages 515–524. ACM, 2002.

## Bio-data

Yogish Sabharwal obtained his Bachelors degree in Mathematics from Hansraj College, Delhi University in 1994. He obtained his Master of Computer Applications degree from Delhi University in 1997. He joined the doctoral program at IIT Delhi in 2001 on a part-time basis. He is working as a Research Staff Member at IBM India Research Laboratory since June 2003. Between June 1997 and June 2003, he has worked with Nortel Networks and Hughes Software Systems. His research interests include computational geometry, approximation algorithms, high performance computing and distributed computing.