Approximation Algorithms for Clustering and Facility Location Problems

by

Sara Ahmadian

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2017

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner	Anupam Gupta Professor
Supervisor	Chaitanya Swamy Professor
Internal Member	Joseph Cheriyan Professor
Internal Member	Laura Sanità Assistant Professor
Internal-external Member	Lap Chi Lau Associate Professor

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

In this thesis, three results based on three different publications are presented. I have made major contributions to all three publications.

The first result is on *mobile facility location* problem (Chapter 2) which is a joint work with Zachary Friggstad and Chaitanya Swamy [5]. All parties contributed equally; Chaitanya Swamy's role was more of an advisory nature.

The second result is *minimum-load k facility location* problem (Chapter 3) which is a joint work with Babak Behsaz, Zachary Friggstad, Amin Jorati, Mohammad Salavatipour, and Chaitanya Swamy [4]. This project was initially investigated independently by two groups, one group at the University of Waterloo (Ahmadian, Friggstad, Swamy), and the other at the University of Alberta (Behsaz, Friggstad, Jorati, Salavatipour; Friggstad, who was a postdoctoral scholar at Waterloo joined the University of Alberta as a faculty member, while the project was ongoing). After the realization of the other groups' work, two groups collaborated and all parties had contributed equally.

The last result is on lower-bounded clustering (Chapter 4) and is a joint work with Chaitanya Swamy [7] who contributed chiefly in an advisory capacity.

Abstract

Facility location problems arise in a wide range of applications such as plant or warehouse location problems, cache placement problems, and network design problems, and have been widely studied in Computer Science and Operations Research literature. These problems typically involve an underlying set $\mathcal{F}$ of facilities that provide service, and an underlying set $\mathcal{D}$ of clients that require service, which need to be assigned to facilities in a cost-effective fashion. This abstraction is quite versatile and also captures clustering problems, where one typically seeks to partition a set of data points into k clusters, for some given k, in a suitable way, which themselves find applications in data mining, machine learning, and bioinformatics.

Basic variants of facility location problems are now relatively well-understood, but we have much-less understanding of more-sophisticated models that better model the real-world concerns. In this thesis, we focus on three models inspired by some real-world optimization scenarios.

In Chapter 2, we consider *mobile facility location* (MFL) problem, wherein we seek to relocate a given set of facilities to destinations closer to the clients as to minimize the sum of facility-movement and client-assignment costs. This abstracts facility-location settings where one has the flexibility of moving facilities from their current locations to other destinations so as to serve clients more efficiently by reducing their assignment costs. We give the first *local-search based* approximation algorithm for this problem and achieve the best-known approximation guarantee. Our main result is $(3+\epsilon)$-approximation for this problem for any constant $\epsilon > 0$ using local search which improves the previous best guarantee of 8-approximation algorithm due to [34] based on LP-rounding. Our results extend to the weighted generalization wherein each facility i has a non-negative weight w_i and the movement cost for i is w_i times the distance traveled by i.

In Chapter 3, we consider a facility-location problem that we call the *minimum-load k-facility location* (MLkFL), which abstracts settings where the cost of serving the clients assigned to a facility is incurred by the facility. This problem was studied under the name of min-max star cover in [32, 10], who (among other results) gave bicriteria approximation algorithms for MLkFL when $\mathcal{F} = \mathcal{D}$. ML$k$FL is rather poorly understood, and only an $O(k)$-approximation is currently known for MLkFL, *even for line metrics*. Our main result is the *first polytime approximation scheme* (PTAS) for MLkFL on line metrics (note that no non-trivial true approximation of any kind was known for this metric). Complementing this, we prove that MLkFL is strongly *NP*-hard on line metrics.

In Chapter 4, we consider clustering problems with *non-uniform lower bounds and outliers*, and obtain the *first approximation guarantees* for these problems. We consider objective functions involving the radii of open facilities, where the radius of a facility i is the maximum distance

between i and a client assigned to it. We consider two problems: minimizing the sum of the radii of the open facilities, which yields the *lower-bounded min-sum-of-radii with outliers* (LBkSRO) problem, and minimizing the maximum radius, which yields the *lower-bounded k-supplier with outliers* (LBkSupO) problem. We obtain an approximation factor of 12.365 for LBkSRO, which improves to 3.83 for the non-outlier version. These also constitute the *first* approximation bounds for the min-sum-of-radii objective when we consider lower bounds and outliers *separately*. We obtain approximation factors of 5 and 3 respectively for LBkSupO and its non-outlier version. These are the *first* approximation results for k-supplier with *non-uniform* lower bounds.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Chaitanya Swamy. I am grateful for his enthusiastic supervision of all aspects of my professional development from academic advice to correcting punctuation in my work. This thesis would not have been possible without his tremendous support, patience, and unsurpassed knowledge. I am forever indebted to him.

My special thanks go to my research collaborators for stimulating discussions in our research meetings: Umang Bhaskar, Zachary Friggstad, Hamideh Hosseinzadeh, Ashkan Norouzi-fard, Laura Sanità, Ola Svensson, and Justin Ward. Especially, I would like to thank Zac for patiently teaching me various topics during his Postdoc at the University of Waterloo (UW) and also inviting me for a research visit to the University of Alberta. I am indebted to Laura for making cooperation fun and easy. Special thanks to Ola for giving me the opportunity for an internship at the EPFL and providing an energetic and fun work environment.

I am grateful to my examining committee, Joseph Cheriyan, Anupam Gupta, Lap Chi Lau, and Laura Sanità for accepting to serve on my committee and their helpful suggestions.

My life at the department has been a fantastic experience, not just due to many great department gatherings and free cookies but most importantly, the amazing people I met. Many thanks to my friends and colleagues for discussions we have had regarding different projects and for being such amazing and helpful colleagues: Abbas, Alan, André, Charupriya, Luis, Mehdi, Miaolan, Sahar, Sharat, Shubham, and Venus. Special thanks to Ahmad, Costis, Kostya for organizing reading group seminars from which I learned a lot. I am much obliged to the department's administration team, Alfred, Carol, Jochen, Megan, and Melissa who were always ready to help with smiling faces. I am grateful to UW as well as to the Natural Sciences and Engineering Research Council of Canada (NSERC) for funding my studies.

I thank my wonderful friends in Waterloo who all enriched my life in one way or another. Special thanks to Yasaman, Aynaz, Bita, Kaveh, Negar, Mohammad, Mahsa, Bijan, Elaheh, Amin, Ana, Vajih, Avin, Pouria, Abtin, Zahra, Mojtaba, Nazgol, and Iman. I would like to thank Negar and Mohammad for their hospitality during my visits to Waterloo in my last semesters.

I am truly privileged to have an amazingly supportive and loving family. My greatest, deepest thanks to my dear parents Mohammad and Ashraf. I am grateful to my in-laws, Aziz, Fatemeh, and Sara for their endless patience, encouragement and unlimited love. I am grateful to my sisters and their families in Iran for their unconditional love and support: Fariba, Behzad, Maryam, Ali, Aida, Atrin, and Ilia. I owe a debt of gratitude to my family in Canada, Mahnaz, Aria, and Amir, for their never-ending support and providing the second home for me here. Last but not least, I would like to thank Siavash for his love, care, and understanding. He encouraged and lightened me up all the time and never failed to put a smile on my face.

Dedication

To my beloved parents,
Ashraf and Mohammad.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

An important exercise that rises in operation research is that of optimizing the cost of repetitive tasks. Organizations seek to optimize their operations to minimize costs and improve efficiency. Typically, each operation is associated with some cost effectiveness and it provides some service to a set of demands or clients. Examples of such operations include setting up manufacturing plants, storage/distribution centers, hospitals, and fire stations. More-modern examples include the placement of proxy servers on the web. *Facility location* (FL) problems were proposed in the Operation Research literature as a means of providing mathematical formulations of the common optimization problems underlying these examples. We start by elaborating two examples of optimization scenarios motivated by real-world settings that are captured by facility location problems.

Consider a media company that aims to locate newspaper stands in a city. The company has determined the cost of setting up a stand in each potential location/neighborhood and it knows the demand in each neighborhood. The optimization question here is where the company should locate its stands in order to minimize the total setup cost together with the cost incurred by its customers traveling to these stands.

Consider a government that wants to locate polling stations for an election. There might be a large set of potential locations such as schools and sports halls but since the government has limited resources such as voting booths or staff members, it wants to select a fixed number of these locations. The government knows the number of potential voters in each neighborhood. There are two main attractive objectives to pursue. The first objective, which comes from a ”fairness” perspective, is to maximize the maximum travel time of potential voters, and the second objective, which comes from a ”social welfare” perspective, is to minimize the average travel time of all potential voters.

The preceding questions are typical examples of FL problems that have been widely studied since the early 1960's. These problems are described by four common elements:

- A set of locations where *centers* or *facilities* may be built or opened. There is usually an opening cost associated with each location.
- A set of *demand points* or *clients*, which need to be served. Each client interacts with a facility incurring a certain cost, e.g., traveling time in the last example, that is often termed the *connection* or *assignment cost*.
- A list of requirements that need to be satisfied by the open facilities or the assignments of clients to the open facilities.
- A cost function that associates each solution, specified by a set of open facilities and the assignment of clients to these open facilities, with a cost.

The goal in FL problems is to determine a set of facilities to open and an assignment of clients to these set that satisfies the requirement conditions while minimizing the cost function.

Numerous facility-location problems arise by varying the above elements, in particular, the list of requirements and the cost function. The simplest FL problem is *uncapacitated facility location* (UFL) problem, also known as *plant location*, where there are there are no requirements (other than that every client must be assigned to an open facility), and the cost is simply the sum of the opening costs of the facilities and the connection costs of the clients (newspaper example). A more-realistic generalization, called *capacitated facility location* (CFL) problem, emerges from UFL by adding the requirement that each open facility can serve only a certain bounded number of clients.

FL problems also capture clustering problems, which involve aggregating data points into different groups (with the underlying metric space specifying the similarity between data points); such problems can often be viewed as FL problems where the facilities are free. Clustering problems find application in a variety of settings (see, e.g., Jain [51]): examples include image segmentation, information retrieval, and the grouping of customers into different types for efficient marketing. Two popular examples of clustering problems are the k-center problem, wherein we seek to open k centers (or facilities) so as to minimize the maximum connection cost of a client (a fairness perspective), and the k-median problem, where we again seek to open k facilities to minimize the sum of client connection costs (a social-welfare perspective).

Most FL problems are *NP*-hard, so we do not expect that there is a polytime algorithm that finds an optimal solution for all instances of the problem. We, therefore, focus on the development of *approximation algorithms*. An α-approximation algorithm is one that, for every instance,

returns a solution of cost at most α times the optimum; the quantity α is usually called the approximation guarantee or ratio of the algorithm. A polynomial-time approximation scheme (PTAS) is an (family of) approximation algorithm(s) that, for any fixed $\epsilon > 0$, achieves approximation ratio $(1+\epsilon)$ in time polynomial in the input size; a fully polynomial-time approximation scheme (FPTAS) is a PTAS whose running time is also polynomial in $1/\epsilon$.

FL problems have been extensively studied in Operation Research and Theoretical Computer Science communities and different techniques have been developed for devising approximation algorithms for these problems. These techniques can be categorized into three major groups: LP-rounding, Primal-Dual(PD), and local search. In the LP rounding technique, the algorithm rounds an optimal solution to an underlying LP-formulation of the problem to an integral solution. The maximum ratio between the solution quality of the integer program and its relaxation is called *integrality gap* of an LP formulation. Clearly, LP-rounding techniques only succeed if the integrality gap of the underlying LP-formulation is bounded. A merit of using LP-rounding techniques is that they are often versatile and extend to other related problems with similar relaxation. The PD approach implicitly relies on an LP-formulation of the problem, and it typically constructs a feasible dual solution and a feasible (integral) primal solution simultaneously, and then bounds the cost of the constructed primal solution in terms of the cost of the constructed dual solution. In local search, the algorithm repeatedly moves from one feasible solution to a neighboring solution with lower cost, and terminates with a locally-optimal solution. Such algorithms have been successfully applied to a large variety of FL problems, and in some cases, they are the only successful approach and yield the current-best approximation guarantees. The maximum ratio between the solution quality of a local optimum and a global optimum is called *locality-gap*. Ease of understanding and implementation has made local search the method of choice for implementation by practitioners.

Whereas basic variants of FL location problems are now-relatively well-understood, we have much less understanding of more-sophisticated variants that better model real-world settings. In this thesis, our goal is to leverage the insights gained from the study of the basic variants to better understand such sophisticated models. Our focus is on developing good approximation algorithms for these problems. From practitioner's viewpoint, one should remember that our analysis gives an upper-bound on the running time and the ratio of the cost of our solution to the cost of an optimum but in practice, it is possible that the algorithm terminates in a faster time and it might return a solution which has a better ratio. We next discuss the various FL problems considered in this thesis.

Problem		Our results	Previous work
Mobile FL (Ch 2)		$3+\epsilon$	8 [34, 79]
Minimum-load k-FL (Ch 3)		PTAS	$(3+\epsilon, 3)$-bicriteria ([10]) $O(k)$ (via $O(1)$ for k-median)
Lower-bounded (LB) clustering (Ch 4)	LB k-supplier	3	2 (uniform LB, $\mathcal{F}=\mathcal{D}$ [3, 2])
	LB k-supplier with outliers	5	4 (uniform LB, $\mathcal{F}=\mathcal{D}$ [3, 2])
	LB min-sum-of-radii	3.83	3.53 (no lower-bounds [22]) QPTAS (no lower-bounds)
	LB min-sum-of-radii with outliers	12.365	-

Table 1.1: Summary of our main results. We use $\mathcal{F}$ to denote the facility set and $\mathcal{D}$ to denote the client set.

1.1 Models studied in this thesis

In this section, we introduce three problems inspired by realistic optimization scenarios. These problems model more sophisticated real-world problems. In each subsection, we focus on one problem and first start with an example motivating the problem, then we highlight our contribution to the given problem (more details of used techniques and more comprehensive description of related work are given in the corresponding chapter). Table 1.1 summarizes our main results for the three facility locations problems that we considered in this thesis.

1.1.1 Mobile facility location

Consider a setting where a company seeks a cost-efficient method for delivering products from its plants to its customers. The company also owns some distribution centers, and each plant has a truck. So a reasonable approach is to move products from each plant to a distribution center, from where the customers pick up their demanded products. This setting is abstracted by *mobile facility location*(MFL) problem wherein we are given an initial location of k facilities and clients, all located in a common metric and the goal is to find a final location for each facility and assign clients to these facilities such that the total facility movement cost and client connection costs is

minimized. Mobile facility location can also be motivated from the perspective of reoptimization. Suppose we have a current UFL solution, but clients get relocated after getting service, or equivalently, a new set of clients appear after servicing one set of clients. For example, in newspaper stand example, the media company may have changed its content resulting in a new set of demands in each neighborhood. The goal in MFL is to minimize the sum of relocation costs of facilities and the sum of the (new) clients' connection costs. MFL was introduced by Demaine et al. [28] in the context of movement problems.

We devise the first local-search based algorithm for MFL (see Chapter 2), which also yields the current best approximation for this problem. Our algorithm achieves an approximation ratio of $(3+\epsilon)$, for any $\epsilon > 0$; The previous best guarantee for MFL was an 8-approximation algorithm due to Friggstad and Salavatipour [34] based on LP-rounding. There is an approximation preserving reduction from the k-median problem to MFL and our approximation guarantee matches the current-best local-search based approximation ratio of $(3+\epsilon)$ for k-median [11] (the approximation ratio for k-median was however improved recently using LP-based approaches). Our analysis is tight (up to $o(1)$ factors) as the example showing locality gap of 3 for k-median can be translated to yield the same locality gap for out local-search algorithm for MFL. Our results extend to the weighted generalization where each facility i has a non-negative weight w_i, and the movement cost for i is w_i times the distance traveled by i.

1.1.2 Minimum load k-facility location

Consider the polling station example with the following twist instead of clients coming to the stations, we have volunteers that start at their own station and collect votes from potential voters in their home. Each volunteer returns to its station after collecting votes from each location to store the results at his/her station for privacy and security reasons. In order to insure fair distribution of work, we want to minimize the maximum distance traveled by each volunteer. This setting inspires an FL model where we want to open k facilities and assign clients to the open facilities, and the goal is to minimize the maximum cost borne by each facility. This problem is called *minimum load k-facility* (MLkFL) problem, wherein the objective is to minimize the maximum load of each facility, where the load of each facility defined as the sum of connection cost of clients assigned to it.

The MLkFL problem was studied under the name min-max star cover in Even et al. [32] and Arkin et al. [10], who (among other results) gave bicriteria approximation algorithms for the setting where the client set and facility set are the same. MLkFL is rather poorly understood, and only an $O(k)$-approximation is currently known for MLkFL, even for line metrics. Our main result is the first polynomial time approximation scheme (PTAS) for MLkFL (see Chapter 3).

Complementing this, we prove that MLkFL is strongly *NP*-hard on line metrics.

1.1.3 Clustering problems with lower-bounds and outliers

An important concern that arises in publishing data (e.h., census data) is *data privacy*. For example, one would like to publish the data obtained from a study on patients of a hospital while maintaining the privacy of individual patients. A naive approach for achieving this is by removing the fields of data corresponding to personal identification, such as social security number or name. However, this does not quite work, since by combining multiple databases, one can use the combination of non-key attributes, called quasi-identifiers, to identify the identity of an individual. In order to achieve anonymity, Samarati [73] proposed perturbing some attributes of data points and then clustering these perturbed data points such that there are at least L identical perturbed data points in each cluster, for some fixed L, thus making it difficult to identify an individual from this perturbed data. Aggrawal et al. [3, 2] observed that this problem can be abstracted as a lower-bounded clustering problem where the objective function captures the cost of perturbing data points.

The preceding example provides one motivation for the study of lower-bounded clustering problems. Again various objectives can be considered in this setting. Suppose that a clustering is obtained by assigning data points to a center, and define the radius of the cluster to be defined as the distance between the center and the furthest data point in the cluster. Two natural objectives are minimizing the maximum radius, which is the k-center objective, and minimizing the sum of the cluster radii. We use the term *lower-bounded k-supplier* (LBkSup) problem to refer to the first objective function (k-supplier is a generalization of k-center, where the assumption that facility set and client set are the same, is dropped); The problem corresponding to the second objective is called the*lower-bounded min-sum-of-radii* (LBkSR) problem.

A common woe in clustering problems is the existence of data points that are quite dissimilar from other data points. Any clustering of the entire data points in such circumstances is likely to have poor quality. Allowing the flexibility of not clustering all data points, e.g., requiring only 90% of data points to be clustered, allows one to direct attention to the data points of interest, and can thereby improve the quality of the clustering (and in some cases drastically). The unclustered data points are called *outliers* and the outlier version of the two versions of clustering problems we consider are called *lower-bounded k-supplier with outliers*(LBkSupO) and *lower-bounded min-sum-of-radii with outliers* (LBkSRO).

We obtain approximation factors of 5 and 3 respectively for LBkSupO and its non-outlier version. These are the *first* approximation results for k-supplier with *non-uniform* lower bounds.

We obtain an approximation factor of 12.365 for LBkSRO, which improves to 3.83 for the non-outlier version These also constitute the *first* approximation bounds for the min-sum-of-radii objective when we consider lower bounds and outliers *separately*. For the min-sum-of-radii problems, we apply the primal-dual method to the relaxation where we Lagrangify the constraint on the number of opened centers. The chief technical contribution and novelty of our algorithm is that, departing from the standard paradigm used for such constrained problems, we obtain an $O(1)$-approximation *despite the fact that we do not obtain a Lagrangian-multiplier-preserving algorithm for the Lagrangian relaxation*. We believe that our ideas have broader applicability to other clustering problems with outliers as well (see Chapter 4 for more details).

1.2 Related work

There is a wealth of work on clustering and facility location problems (see e.g., [69] and the survey [74]). Here, we briefly survey the work on more classical facility location problems: UFL, CFL, k-center, and k-median; we discuss the related work on the specific problems considered in this thesis in the chapters pertaining to these problems. UFL is the most investigated facility location problem. Although the problem was intensively studied since the 1960s (see, e.g., Stollsteimer [77], Balinski and Wolfe [12], Kuehn and Hamburger [61], Manne [68]), it was in 1997 that the first constant-factor approximation was devised for UFL with metric connection costs by Shmoys et al. [76]. This result initiated a long line of research and resulted in the development of various algorithmic techniques, including LP-rounding [24, 15], primal-dual methods [54, 53], and local-search methods [11, 18]. The current best is due to Li [63] who combined an algorithm by Byrka [15] and an algorithm by Jain et. al. [53] to achieve an approximation ratio of 1.488. This result almost settles the approximation of metric UFL as Guha and Khuller [40] proved that it is *NP*-hard to devise an algorithm with approximation ratio better than $s_0 \sim 1.463$ where s_0 is the solution to the equation $s_0 = 1 + 2e^{s_0}$ (e is the base of the natural algorithm).

The CFL problem is another highly investigated problem. All initial known constant-factor approximation algorithm for CFL were based on local-search paradigm. The first constant-factor approximation was obtained for a special case of uniform capacities by Korupolu et al. [59] who analyzed a simple local-search based algorithm of Kuehn and Hamburger [61]. Subsequently, Chudak and Williamson [25] improved the analysis, and finally, Aggrawal et al. [1] added the last ingredients to obtain the current-best $(3+\epsilon)$-approximation ratio for this special case. For the general case of CFL (non-uniform capacities), the first constant approximation ratio was obtained by Pál et al. [71]. Since then more sophisticated local-search based algorithms were suggested and the current best $(5 + \epsilon)$-approximation is due to Bansal et al. [13]. In 2014, An et al. [9] made an important advance by devising the first LP-relaxation for CFL with constant integrality

gap.

Clustering methods have been studied since the 1950's in various fields including statistics, biology, computer science, social science, and engineering. k-center is a fairly well-understood clustering problem. Gonzalez [38] and Hochbaum and Shmoys [47] developed 2-approximation algorithm for k-center. This result is the best possible as the existence of α-approximation algorithm with $\alpha < 2$, implies *P =NP* ([46], [50]). An extension of k-center is obtained when the set of centers and clients are not necessarily the same and this problem is called *k-supplier*. Hochbaum and Shmoys [49] provided a 3-approximation algorithm for k-supplier and this result is again tight as Karloff proved, if there exists a $(3-\epsilon)$-approximation algorithm for k-supplier for any fixed $\epsilon > 0$, then *P =NP* (see [49]).

Metric k-median is another fundamental location problem in Combinatorial Optimization which falls under clustering category. The first constant approximation ratio of $6\frac{2}{3}$ was obtained via LP-rounding by Charikar et al. [19]. Jain and Vazirani [54] developed an elegant primal-dual approach for UFL and combined this with the idea of Lagrangian relaxation to obtain an improved 6-approximation for k-median. In the Lagrangian relaxation of k-median, the hard constraint on the number of facilities/centers is lifted and instead a fixed cost is incurred every time a facility is opened, thus yielding a UFL instance. Using their primal-dual technique, Jain and Vazirani obtained two solutions for UFL, corresponding to very nearby values of z, one opening fewer than k facilities but having (possibly) large cost, and the other opening more than k facilities (but having low cost). These two solutions are then combined to obtain a k-median solution with the cost at most 6 times the cost of the optimal k-median solution. Later Jain et al. [52] improved this ratio to 4. Arya et al. [11] analyzed a simple local-search based algorithm and showed that this achieves $(3+\epsilon)$ approximation ratio for k-median; their analysis was subsequently simplified by Gupta and Tangwongsa [42]. In 2012, Li and Svensson [64] improved this decade-old result by devising a $(1+\sqrt{3}+\epsilon)$ approximation ratio. Subsequently, Byrka et al. [16] achieved approximation ratio of 2.675, the current state-of-the-art, by optimizing the ingredients of the Li-Svensson algorithm using dependent rounding. The best hardness result for k-median is due to Jain et al. [52] showing that k-median cannot be approximated within a factor $(1+\frac{2}{e}-\epsilon)$ for $\epsilon > 0$ unless $NP \subseteq DTIME[n^{O(\log\log n)}]$.

1.3 Notation used in the thesis

Concluding this chapter, we introduce some common notation used throughout this thesis here. Recall that every FL problem consists of a set of facilities or centers and a set of clients (the two sets could be the same). We use $\mathcal{F}$ to denote the set of facilities or centers and we usually use i for indexing facilities. For FL problems where an opening cost is given for each facility, we

use f_i to denote the opening cost of facility i. We use $\mathcal{D}$ to denote the set of clients or demands and j to index the clients. In the settings where clients may have non-unit demands, we use d_j to denote client j's demand. For all problems (and the vast majority of FL problems in the literature), we assume that clients and facilities are located in a common metric space, and $c(i, j)$ denotes the distance between locations $i, j \in \mathcal{F} \cup \mathcal{D}$; that is, the $c(i, j)$ distances satisfy the triangle inequality:

$$c(i, j) \leq c(i, l) + c(l, j) \qquad \forall\ i, j, l \in \mathcal{F} \cup \mathcal{D}$$

Given an underlying FL solution (which will be clear from the context), we use $\sigma(j)$ for each client j, to denote the facility to which j is assigned.

For clustering problems, i.e., problems with no facility opening costs but with a bound k on the number of opened facility in each solution, each cluster is identified by a center and a set of clients assigned to this center. As we use $\sigma(j)$ to denote the assigned cluster center for each client j, so set $\sigma^{-1}(i)$ denotes the set of clients in the cluster with center i. For a cluster with center i, we define the *radius* of the cluster, denoted by r_i, to be the distance between i and the furthest client assigned to i, i.e., $r_i = \max_{j \in \sigma^{-1}(i)} c(i, j)$ (if $\sigma^{-1}(i)$ is empty, we define $r_i = 0$).

Chapter 2

Mobile Facility Location

2.1 Introduction

In this chapter, we consider settings where facilities are mobile and may be relocated to destinations near the clients in order to serve them more efficiently by reducing the client-assignment costs. Formally, we consider the *mobile facility location* (MFL) problem introduced by [28, 34], which generalizes the classical k-median problem (see below). We are given a metric space $(V, \{c(i,j)\}_{i,j \in V})$ on a set V of locations, a set $\mathcal{F} \subseteq V$ of k initial facility, and a set $\mathcal{D} \subseteq V$ of clients. A solution S to MFL moves each facility $i \in \mathcal{F}$ to a final location $s_i \in V$ (which could be the same as i), incurring a *movement cost* $c(i, s_i)$, and assigns each client j to the nearest location in S (break ties arbitrarily) denoted by $\sigma(j)$, incurring the assignment cost of $c(j, \sigma(j))$. The total cost of S is the sum of all the movement costs and assignment costs.

MFL can be motivated as a re-optimization version of UFL. Suppose we have initially opened some facilities, but then the client locations change (i.e., the demand-pattern changes). Instead of sticking with the original facility locations, which may incur a large assignment cost for the clients, one may find it advantageous to relocate the facilities to other locations closer to the clients incurring a certain movement cost; MFL captures the resulting optimization problem of minimizing the sum of the movement and (new) client assignment costs.

Mobile facility location falls into the genre of *movement problems* introduced by Demaine et al. [28]. In these problems, we are given an initial configuration in a weighted graph specified by placing "pebbles" on the nodes and/or edges; the goal is to move the pebbles so as to obtain a desired final configuration while minimizing the maximum, or total, pebble movement. MFL was introduced by Demaine et al. as the movement problem where facility- and client-

pebbles are placed respectively at the initial locations of the facilities and clients, and in the final configuration every client-pebble should be co-located with some facility-pebble.

2.1.1 Summary of results

We give the first *local-search based* approximation algorithm for this problem and achieve the best-known approximation guarantee. Our main result is a $(3+\epsilon)$-approximation for this problem for any constant $\epsilon > 0$ using a simple local-search algorithm. This improves upon the previous best 8-approximation guarantee for MFL due to Friggstad and Salavatipour [34], which is based on LP-rounding and is not combinatorial.

The local-search algorithm we consider is quite natural and simple. Observe that given the final locations of the facilities, we can find the minimum-cost way of moving facilities from their initial locations to the final locations by solving a minimum-cost perfect-matching problem (and the client assignments are determined by the function σ defined above). Thus, we concentrate on determining a good set of final locations. In our local-search algorithm, at each step, we are allowed to swap in and swap out a fixed number (say p) of (final) locations. Clearly, for any fixed p, we can find the best local move efficiently (since the cost of a set of final locations can be computed in polytime). Note that we do not impose any constraints on how the matching between the initial and final locations may change due to a local move, and a local move might entail moving all facilities. It is important to allow this flexibility, as it is known [34] that the local-search procedure that moves, at each step, a constant number of facilities to chosen destinations has an unbounded approximation ratio.[1]

Our analysis is tight (Section 2.5). Notice that there is an approximation-preserving reduction from the k-median problem to MFL [34]: choose arbitrary initial facility locations and give each client a huge demand D (see Section 2.7). The tightness of our analysis (up to $o(1)$ factors) follows because by suitably setting D in this reduction, we can ensure that our local-search algorithm for MFL essentially coincides with the local-search algorithm for k-median in [11], which has a tight approximation ratio of 3.

We also consider a weighted generalization of the problem (Section 2.6), wherein each facility i has a weight w_i indicating the cost incurred per-unit distance moved and the cost for moving i to s_i is $w_i c(i, s_i)$. (This can be used to model, for example, the setting where different facilities move at different speeds.) Our analysis is versatile and extends to this weighted generalization to yield the same performance guarantee. For the further generalization of the problem, where the

[1] Whereas we consider p-swaps here for $p > 1$, it is natural to consider what happens if we only swap in and out a single final location. In the arxiv version of this work [5], we show that this also yields a constant-factor approximation, but the analysis is significantly different and involved and we obtain a larger constant.

facility-movement costs may be arbitrary and unrelated to the client-assignment costs (for which an 8-approximation can be obtained via LP-rounding; see "Related work"), we show that local search based on multiple swaps has a bad approximation ratio (Section 2.8).

The results presented in this chapter are part of a joint work with Chaitanya Swamy and Zachary Friggstad and it was published in 2013 [5].

2.1.2 Related work

As mentioned earlier, MFL was introduced by Demaine et al. [28] in the context of movement problems. Demaine et al [28] considered various movement objectives and for the case that the objective is to minimize the maximum facility movement cost or client connection cost, they devise a 2-approximation algorithm. They left the problem with the objective function defined as the sum of facility movement costs and clients' connection costs, an open question. Friggstad and Salavatipour [34] designed the first approximation algorithm for MFL. They gave an 8-approximation algorithm based on LP rounding by building upon the LP-rounding algorithm of Charikar et al. [19] for the k-median problem; this algorithm works only however for the unweighted case. They also observed that there is an approximation-preserving reduction from k-median to MFL. Halper [44] in his thesis, proposed the same local-search algorithm that we analyze. His work focuses on experimental results and leaves open the question of obtaining theoretical guarantees about the performance of local-search.

Swamy [79] observed that MFL, even with arbitrary movement costs is a special case of the *matroid median* problem [60] wherein we are given a set of facilities with facility opening costs and a matroid on facility-set, and a set of clients with demand and connection costs, and the goal is to open an independent set of facilities and assign each client to an open facility so that the sum of facility opening costs and the client connection costs is minimized. Following the work of [60], Charikar et al. [21] and Swamy [79] improved the approximation ratio for matroid median. Algorithm in [79] yields an 8-approximation algorithm for MFL with arbitrary movement costs.

We now focus on results obtained using local-search algorithms for classical facility location problems; UFL, CFL, and k-median. Starting with the work of [59], local-search techniques have been utilized to devise $O(1)$-approximation algorithms for various facility-location problems. Korupolu, Plaxton, and Rajaraman [59] devised $O(1)$-approximation for UFL, and CFL with uniform capacities, and k-median (with a blow-up in k). Charikar and Guha [18], and Arya et al. [11] both obtained a $(1+\sqrt{2})$-approximation for UFL. The first constant-factor approximation for CFL was obtained by Pál, Tardos, and Wexler [71], and after some improvements, the current-best approximation ratio now stands at $5+\epsilon$ [13]. For the special case of uniform

capacities, the analysis in [59] was refined by [25], and Aggarwal et al. [1] obtain the current-best 3-approximation. Arya et al. [11] devised a $(3+\epsilon)$-approximation algorithm for k-median, which was also the first constant-factor approximation algorithm for this problem based on local search. Gupta and Tangwongsan [42] (among other results) simplified the analysis in [11]. We build upon some of their ideas in our analysis.

Local-search algorithms with constant approximation ratios have also been devised for various variants of the above three canonical problems. Mahdian and Pál [67], and Svitkina and Tardos [78] consider settings where the opening cost of a facility is a function of the set of clients served by it. In [67], this cost is a non-decreasing function of the number of clients, and in [78] this cost arises from a certain tree defined on the client set. Devanur et al. [29] and [42] consider k-facility location, which is similar to k-median except that facilities also have opening costs. Hajiaghayi et al. [43] consider a special case of the matroid median problem that they call the red-blue median problem. Gørtz and Nagarajan[39] considered a problem that they call the k-median forest problem, which generalizes k-median, and obtained a $(3+\epsilon)$-approximation algorithm.

2.2 Problem definition and preliminaries

Recall that in the mobile facility location (MFL) problem, we have a metric space $(V, \{c(i,j)\}_{i,j\in V})$, a set $\mathcal{F} \subseteq V$ of initial facility locations, and a set $\mathcal{D} \subseteq V$ of clients. We use term i to denote the facility whose initial location is $i \in \mathcal{F}$. A feasible solution $S \subseteq \mathcal{F}$ identifies k final locations for each of initial locations $i \in \mathcal{F}$, i.e., $S = \{s_1, s_2, \cdots, s_k\}$ and s_i is the final location of facility $i \in \mathcal{F}$, which could be the same as i. For solution S, each client j is assigned to the closest facility (breaking ties arbitrary), denoted by $\sigma(j)$, and this client pays assignment/connection cost of $c(j, \sigma(j))$. The cost of solution S consists of the *movement costs* of facilities which is $c(i, s_i)$ for each initial location $i \in \mathcal{F}$, and the assignment costs of clients which is $c(j, \sigma(j))$ for each client $j \in \mathcal{D}$. More precisely, the cost of S is

$$\mathsf{MFL}(S) = \sum_{i\in\mathcal{F}} c(i, s_i) + \sum_{j\in\mathcal{D}} c(j, \sigma(j)).$$

We assume throughout that the edges form a metric and we use terms nodes and locations interchangeably. We use S to denote the output of our algorithm which is a local optimum and we use $O = \{o_1, \ldots, o_k\}$ to denote the (globally) optimal solution, where again facility i is moved to o_i. Throughout, we use s to index locations in S, and o to index locations in O. For a node $v \in V$, let $\sigma(v)$ be the location in S nearest to v (breaking ties arbitrary). Similarly,

we define $\sigma^*(v)$ to be the location in O nearest to v. For notational similarity with facility location problems, we denote $c(i, s_i)$ by f_i, and $c(i, o_i)$ by f_i^*. (Thus, f_i and f_i^* are the movement costs of i in S and O respectively.) Also, we abbreviate $c\big(j, \sigma(j)\big)$ to c_j, and $c\big(j, \sigma^*(j)\big)$ to c_j^*. Thus, c_j and c_j^* are the assignment costs of j in the local and global optimum respectively. So $\mathsf{MFL}(S) = \sum_{i \in \mathcal{F}} f_i + \sum_{j \in \mathcal{D}} c_j$ and $\mathsf{MFL}(O) = \sum_{i \in \mathcal{F}} f_i^* + \sum_{j \in \mathcal{D}} c_j^*$.

2.3 The local-search algorithm

As mentioned earlier, to compute a solution to MFL, we only need to determine the set of final locations of the facilities, since we can then efficiently compute the best movement of facilities from their initial to final locations, and the client assignments. This motivates the following local-search operation. Given a current set S of $k = |\mathcal{F}|$ locations, we can move to any other set S' of k locations such that $|S \setminus S'| = |S' \setminus S| \leq p$, where p is some fixed value. We denote this move by $\mathrm{swap}(S \setminus S', S' \setminus S)$. The local-search algorithm starts with an arbitrary set of k final locations. At each iteration, we choose the local-search move that yields the largest reduction in total cost and update our final-location set accordingly; if no cost-improving move exists, then we terminate. (To obtain polynomial running time, as is standard, we modify the above procedure so that we choose a local-search move only if the cost-reduction is at least ϵ fraction of the current cost; see Subsection 2.4.2.)

Before proceeding to the analysis of algorithm, we introduce some notation and a basic lemma used repeatedly in the analysis. Let $D(s) = \{j \in \mathcal{D} : \sigma(j) = s\}$ be the set of clients assigned to the location $s \in S$, and $D^*(o) = \{j \in \mathcal{D} : \sigma^*(j) = o\}$. For a set $A \subseteq S$, we define $D(A) = \bigcup_{s \in A} D(s)$; we define $D^*(A)$ for $A \subseteq O$ similarly. Define $\mathsf{cap}(s) = \{o \in O : \sigma(o) = s\}$. We say that s *captures* all the locations in $\mathsf{cap}(s)$.

Lemma 2.3.1. *For any client j, we have $c\big(j, \sigma(\sigma^*(j))\big) - c\big(j, \sigma(j)\big) \leq 2c_j^*$.*

Proof. Let $s = \sigma(j),\ o = \sigma^*(j),\ s' = \sigma(o)$. The lemma clearly holds if $s' = s$. Otherwise,

$$c(j, s') \leq c(j, o) + c(o, s') \leq c_j^* + c(o, s) \leq c_j^* + c_j^* + c_j,$$

where the second inequality follows since s' is the closest location to o in S. So $c(j, s') - c(j, s) \leq 2c_j^* + c_j - c_j = 2c_j^*$ (see Figure 2.1).

□

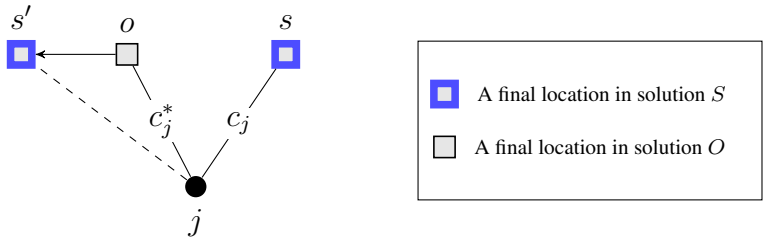


Figure 2.1: Reassignment of a client j from $s = \sigma(j)$ to $s' = \sigma(\sigma^*(j))$.

2.4 Analysis leading to a 5-approximation

We now analyze the above local-search algorithm and show that it is a $\big(5+o(1)\big)$-approximation algorithm. For notational simplicity, we assume that the local-search algorithm terminates at a local optimum; the modification to ensure polynomial running time degrades the approximation by at most a $(1+\epsilon)$-factor (see also Subsection 2.4.2).

Theorem 2.4.1. *Let F^* and C^* denote respectively the movement and assignment cost of an optimal solution. The total cost of any local optimum using at most p swaps is at most $\Big(3+O\big(\frac{1}{p^{1/3}}\big)\Big)F^* + \Big(5+O\big(\frac{1}{p^{1/3}}\big)\Big)C^*$.*

Although this is not the tightest guarantee that we obtain, we present this analysis first since it introduces many of the ideas that we build upon in Section 2.5 to prove a tight approximation guarantee of $\big(3+o(1)\big)$ for the local-search algorithm. All our analyses carry over trivially to the case of non-unit (integer) demand for each client since we can think of a client j having d_j demand as d_j co-located unit-demand clients.

To prove the approximation ratio, we will specify a set of local-search moves for the local optimum, and use the fact that none of these moves improve the cost to obtain some inequalities, which will together yield a bound on the cost of the local optimum. We describe these moves by using the following digraph. Consider the digraph $\widehat{G} = \big(\mathcal{F} \cup S \cup O, \{(s_i, i), (i, o_i), (o_i, \sigma(o_i))\}_{i \in \mathcal{F}}\big)$. We decompose $\widehat{G}$ into a collection of node-disjoint (simple) paths $\mathcal{P}$ and cycles $\mathcal{C}$ as follows; see Figure 2.2. Repeatedly, while there is a cycle C in our current digraph, we add C to $\mathcal{C}$, remove all the nodes of C and recurse on the remaining digraph. After this step, a node v in the remaining digraph, which is acyclic, has: exactly one outgoing arc if $v \in S$; exactly one incoming and one outgoing arc if $v \in \mathcal{F}$; and exactly one incoming, and at most one outgoing arc if $v \in O$. Now we repeatedly choose a node $v \in S$ starting at v in $\mathcal{P}$, remove all nodes of P and recurse on the remaining digraph. Thus, each triple (s_i, i, o_i) is on a unique path or cycle in $\mathcal{P} \cup \mathcal{C}$. Define

$\mathsf{center}(s)$ to be $o \in O$ such that (o, s) is an arc in $\mathcal{P} \cup \mathcal{C}$; if s has no incoming arc in $\mathcal{P} \cup \mathcal{C}$, then let $\mathsf{center}(s) = \mathsf{nil}$.

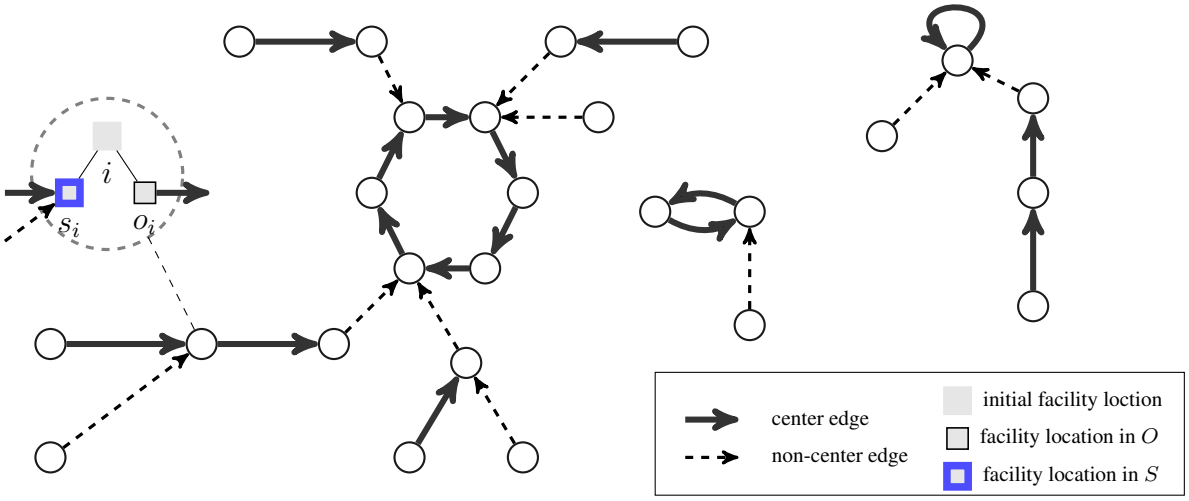


Figure 2.2: Example of decomposing digraph $\widehat{G}$ to set $\mathcal{P}$ of paths and set $\mathcal{C}$ of cycles.

We will use $\mathcal{P}$ and $\mathcal{C}$ to define our swaps. For a path $P = (s_{i_1}, i_1, o_{i_1}, \ldots, s_{i_r}, i_r, o_{i_r}) \in \mathcal{P}$, define $\mathsf{start}(P)$ to be s_{i_1} and $\mathsf{end}(P)$ to be o_{i_r}. Notice that $\sigma(o_{i_r}) \notin P$ by our decomposition process. For each $s \in S$, let $\mathcal{P}_c(s) = \{P : \mathsf{end}(P) \in \mathsf{cap}(s)\}$, $T(s) = \{\mathsf{start}(P) : P \in \mathcal{P}_c(s)\}$, and $H(s) = \{\mathsf{end}(P) : P \in \mathcal{P}_c(s)\} = \mathsf{cap}(s) \setminus \{\mathsf{center}(s)\}$. Note that $|\mathcal{P}_c(s)| = |T(s)| = |H(s)| = |\mathsf{cap}(s)| - 1$ for any $s \in S$ with $|\mathsf{cap}(s)| \geq 1$. For a set $A \subseteq S$, define $T(A) = \bigcup_{s \in A} T(s)$, $H(A) = \bigcup_{s \in A} H(s)$, $\mathcal{P}_c(A) = \bigcup_{s \in A} \mathcal{P}_c(s)$.

A basic building block in our analysis, involves a *shift* along an $s \rightsquigarrow o$ sub-path $Z = (s_{i_1} = s, i_1, o_{i_1}, \ldots, s_{i_r}, i_r, o_{i_r} = o)$ of some path or cycle in $\mathcal{P} \cup \mathcal{C}$. This means that we swap out s and swap in o. We bound the cost of the matching between $\mathcal{F}$ and $S \cup \{o\} \setminus \{s\}$ by moving each initial location $i \in Z$, $i \neq i_r$ to $\sigma(o_i) \in Z$ and moving i_r to o_{i_r} (see Figure 2.3). Thus, we obtain the following simple bound on the increase in movement cost due to this operation:

$$\mathsf{shift}(s, o) = \sum_{i \in Z} (f_i^* - f_i) + \sum_{i \in Z : o_i \neq o} c\big(o_i, \sigma(o_i)\big) \leq 2 \sum_{i \in Z} f_i^* - c(o, \sigma(o)). \tag{2.1}$$

The last inequality uses the fact that $c\big(o_i, \sigma(o_i)\big) \leq c(o_i, s_i) \leq f_i^* + f_i$ for all i. For a path $P \in \mathcal{P}$, we use $\mathsf{shift}(P)$ as a shorthand for $\mathsf{shift}\big(\mathsf{start}(P), \mathsf{end}(P)\big)$.

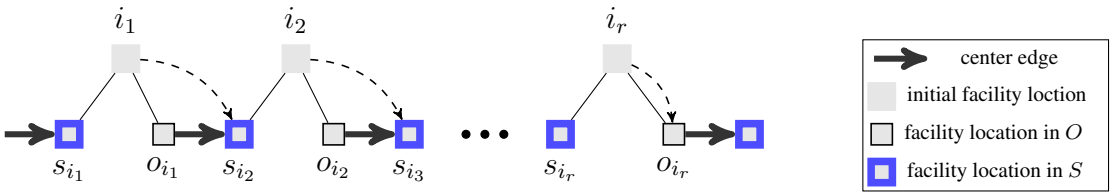


Figure 2.3: Depiction of shift move for subpath Z of a cycle or a path. For each facility i_l for $1 \leq l \leq r$, the dashed arrow leaving i_l shows the new location where the facility i_l is moved to.

2.4.1 The swaps used and their analysis

We now describe the local moves used in the analysis. We define a set of swaps such that each $o \in O$ is swapped in to an extent of at least one, and at most two. We classify each location in S as one of three types. Define $t = \lfloor p^{1/3} \rfloor$. We assume that $t \geq 2$.

- S_0: locations $s \in S$ with $|\mathsf{cap}(s)| = 0$.
- S_1: locations $s \in S \setminus S_0$ with $|D^*(\mathsf{center}(s))| \leq t$ or $|\mathsf{cap}(s)| > t$.
- S_2: locations $s \in S$ with $|D^*(\mathsf{center}(s))| > t$ and $0 < |\mathsf{cap}(s)| \leq t$.

Also define $S_3 := S_0 \cup \{s \in S_1 : |\mathsf{cap}(s)| \leq t\}$ (so $s \in S_3$ iff $|\mathsf{cap}(s)| \leq t$ and $|D^*(\mathsf{center}(s))| \leq t\}$).

To gain some intuition, notice that it is easy to generate a suitable inequality for a location $s \in S_0$: we can "delete" s (i.e., if $s = s_i$, then do $\mathrm{swap}(s, i)$) and reassign each $j \in D(s)$ to $\sigma(\sigma^*(j))$ (i.e., the location in S closest to the location serving j in O). The cost increase due to this reassignment is at most $\sum_{j \in D(s)} 2c^*_j$, and so this yields the inequality $f_i \leq \sum_{j \in D(s)} 2c^*_j$. (We do not actually do this since we take care of the S_0-locations along with the S_1-locations.) We can also generate a suitable inequality for a location $s \in S_2$ (see Lemma 2.4.3) since we can swap in $\mathsf{cap}(s)$ and swap out $\{s\} \cup T(s)$. The cost increase by this move can be bounded by $\sum_{P \in \mathcal{P}_c(s)} \mathsf{shift}(P)$ and $c\big(s, \mathsf{center}(s)\big)$, and the latter quantity can be charged to $\frac{1}{t} \sum_{j \in D^*(\mathsf{center}(s))} (c_j + c^*_j)$; our definition of S_2 is tailored precisely so as to enable this latter charging argument. Generating inequalities for the S_1-locations is more involved and requires another building block that we call an interval swap (this will also take care of the S_0-locations), which we define after proving Lemma 2.4.3. We start out by proving a simple bound that one can obtain using a cycle in $\mathcal{C}$.

Lemma 2.4.2. *For any cycle $Z \in \mathcal{C}$, we have $0 \leq \sum_{i \in Z} \big(-f_i + f^*_i + c(o_i, \sigma(o_i))\big)$.*

Proof. Consider the following matching M from $\mathcal{F} \cap Z$ to $S \cap Z$: we match i to $\sigma(o_i)$. Now consider the matching from $\mathcal{F}$ to S which matches every facility i not in Z to s_i and for facilities in Z, it uses M. The cost of the resulting new matching is $\sum_{i \notin Z} f_i + \sum_{i \in Z} c(i, \sigma(o_i))$ which should at least $\sum_{i \in \mathcal{F}} f_i$ since the latter is the min-cost way of matching $\mathcal{F}$ to S. So we obtain $0 \leq \sum_{i \in Z} \big(-f_i + c(i, \sigma(o_i))\big) \leq \sum_{i \in Z} \big(-f_i + f_i^* + c(o_i, \sigma(o_i))\big)$. □

Lemma 2.4.3. *Let $s \in S_2$ and $o = \mathsf{center}(s)$, and consider $\mathrm{swap}(X := \{s\} \cup T(s), Y := \mathsf{cap}(s))$. We have*

$$0 \leq \mathsf{MFL}\big((S \setminus X) \cup Y\big) - \mathsf{MFL}(S) \leq \sum_{\substack{P \in \mathcal{P}_c(s) \\ i \in P}} 2f_i^* + \sum_{j \in D^*(o)} \left(\tfrac{t+1}{t} \cdot c_j^* - \tfrac{t-1}{t} \cdot c_j\right) + \sum_{\substack{j \in D(\{s\} \cup T(s)) \\ j \notin D^*(o)}} 2c_j^*. \tag{2.2}$$

Proof. We can view this multi-location swap as doing $\mathrm{swap}(\mathsf{start}(P), \mathsf{end}(P))$ for each $P \in \mathcal{P}_c(s)$ and $\mathrm{swap}(s, o)$ simultaneously (see Figure 2.4). (Notice that no path $P \in \mathcal{P}_c(s)$ contains s, since $s = \sigma\big(\mathsf{end}(P)\big) \notin P$.) For each $\mathrm{swap}(\mathsf{start}(P), \mathsf{end}(P))$ the movement-cost increase is bounded by $\mathsf{shift}(P) \leq \sum_{i \in P} 2f_i^*$. For $\mathrm{swap}(s, o)$ we move the facility i, where $s = s_i$, to o, so the increase in movement cost is at most $c(s, o) = c(\sigma(o), o) \leq c(\sigma(j), o) \leq c_j + c_j^*$ for every $j \in D^*(o)$. So since $|D^*(o)| > t$, we have $c(s, o) \leq \sum_{j \in D^*(o)} \frac{c_j + c_j^*}{t}$. Thus, the increase in total movement cost is at most $\sum_{j \in D^*(o)} \frac{c_j + c_j^*}{t} + \sum_{P \in \mathcal{P}_c(s), i \in P} 2 \cdot f_i^*$.

We upper bound the change in assignment cost by reassigning the clients in $D^*(o) \cup D(X)$ as follows. We reassign each $j \in D^*(o)$ to o. Each $j \in D(X) \setminus D^*(o)$ is assigned to $\sigma^*(j)$, if $\sigma^*(j) \in Y$, and otherwise to $s' = \sigma(\sigma^*(j))$. Note that $s' \notin X$: $s' \neq s$ since $\sigma^*(j) \notin \mathsf{cap}(s)$, and $s' \notin T(s)$ since $\bigcup_{s'' \in T(s)} \mathsf{cap}(s'') = \emptyset$. The change in assignment cost for each such client j is at most $2c_j^*$ by Lemma 2.3.1. Thus the change in total assignment cost is at most $\sum_{j \in D^*(o)} (c_j^* - c_j) + \sum_{j \in D(X) \setminus D^*(o)} 2c_j^*$. Combining this with the bound on the movement-cost change proves the lemma. □

We now define a key ingredient of our analysis, called an *interval-swap* operation, that is used to bound the movement cost of the S_1- and S_0-locations and the assignment cost of the clients they serve. (We build upon this in Section 2.5 to give a tighter analysis proving a 3-approximation.) Let $S' = \{s'_1, \ldots, s'_r\} \subseteq S_0 \cup S_1$, $r \leq t^2$ be a subset of at most t^2 locations on a path or cycle Z in $\mathcal{P} \cup \mathcal{C}$, where s'_{q+1} is the next location in $(S_0 \cup S_1) \cap Z$ after s'_q. Let $O' = \{o'_1, \ldots, o'_r\} \subseteq O$ where $o'_{q-1} = \mathsf{center}(s'_q)$ for $q = 2, \ldots, r$ and o'_r is an arbitrary location that appears after s'_r (and before s'_1 if $Z \in \mathcal{C}$) on the corresponding path or cycle. Consider each s'_q. If $|\mathsf{cap}(s'_q)| > t$, choose a *random* path $P \in \mathcal{P}_c(s'_q)$ with probability $\frac{1}{|\mathcal{P}_c(s'_q)|}$, and set

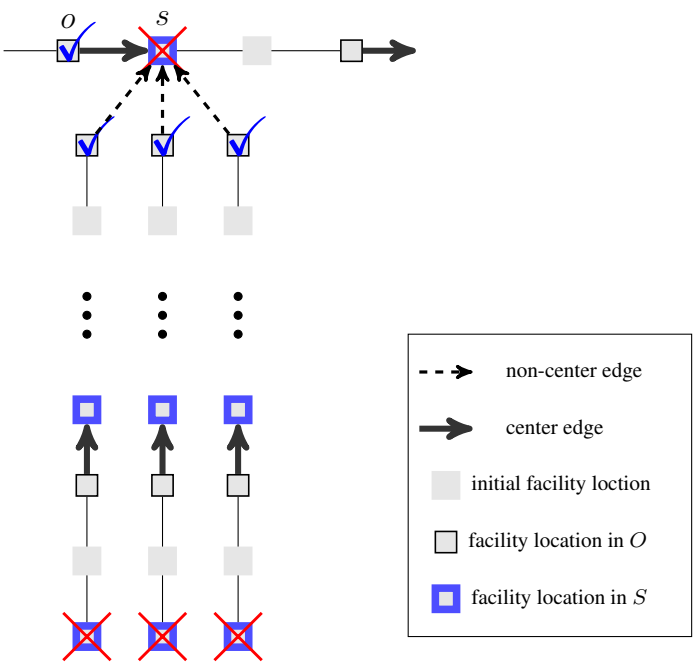


Figure 2.4: Example of $\text{swap}(X := \{s\} \cup T(s), Y := \mathsf{cap}(s))$ for node $s \in S_2$ (i.e., $0 < |cap(s)| < t$ and $|D^*(o)| > t$). Nodes in X are identified by a cross ($\times$) and nodes in Y are identified by a checkmark ($\checkmark$).

$X_q = \{\mathsf{start}(P)\}$ and $Y_q = \{o'_q\}$. If $|\mathsf{cap}(s'_q)| \leq t$, set $X_q = \{s'_q\} \cup T(s'_q)$, and $Y_q = \{o'_q\} \cup H(s'_q)$. Set $X = \bigcup_{q=1}^{r} X_q$ and $Y = \bigcup_{q=1}^{r} Y_q$. Note that $|X| = |Y| \leq t^3$ since $|X_q| = |Y_q| \leq t$ for every $q = 1, \ldots, r$. Notice that X is a random set, but $Y = O' \cup H(S' \cap S_3)$ is deterministic. To avoid cumbersome notation, we use $\text{swap}(X, Y)$ to refer to the distribution of swap-moves that results by the random choices above, and call this the *interval swap corresponding to S' and O'*. We bound the expected change in cost due to this move below. Let $\mathbf{1}(s)$ be the indicator function that is 1 if $s \in S_3$ and 0 otherwise.

Lemma 2.4.4. *Let $S' = \{s'_1, \cdots, s'_r\} \subseteq S_0 \cup S_1$, $r \leq t^2$ and O' be as given above. Let $o'_0 := \mathsf{center}(s'_1) = o_i$, where $o'_0 = \mathsf{nil}$ and $D^*(o'_0) = \emptyset$ if $s'_1 \in S_0$. Consider the interval swap $\text{swap}\big(X = \bigcup_{q=1}^{r} X_q, Y = \bigcup_{q=1}^{r} Y_q\big)$ corresponding to S' and O', as defined above (see Figure 2.5) We have*

$$0 \leq \mathrm{E}\left[\mathsf{MFL}\big((S \setminus X) \cup Y\big) - \mathsf{MFL}(S)\right] \leq \sum_{q=1}^{r} \mathsf{shift}(s'_q, o'_q) + \sum_{P \in \mathcal{P}_c(S'), i \in P} 2f^*_i + \sum_{j \in D^*(O')} (c^*_j - c_j)$$
$$+ \sum_{j \in D(T(S' \cap S_3) \cup (S' \cap S_3))} 2c^*_j + \sum_{j \in D(T(S' \setminus S_3))} \tfrac{2c^*_j}{t} + \mathbf{1}(s'_1) \sum_{j \in D^*(o'_0)} (f^*_{\hat{\imath}} + f_{\hat{\imath}} + c^*_j). \tag{2.3}$$

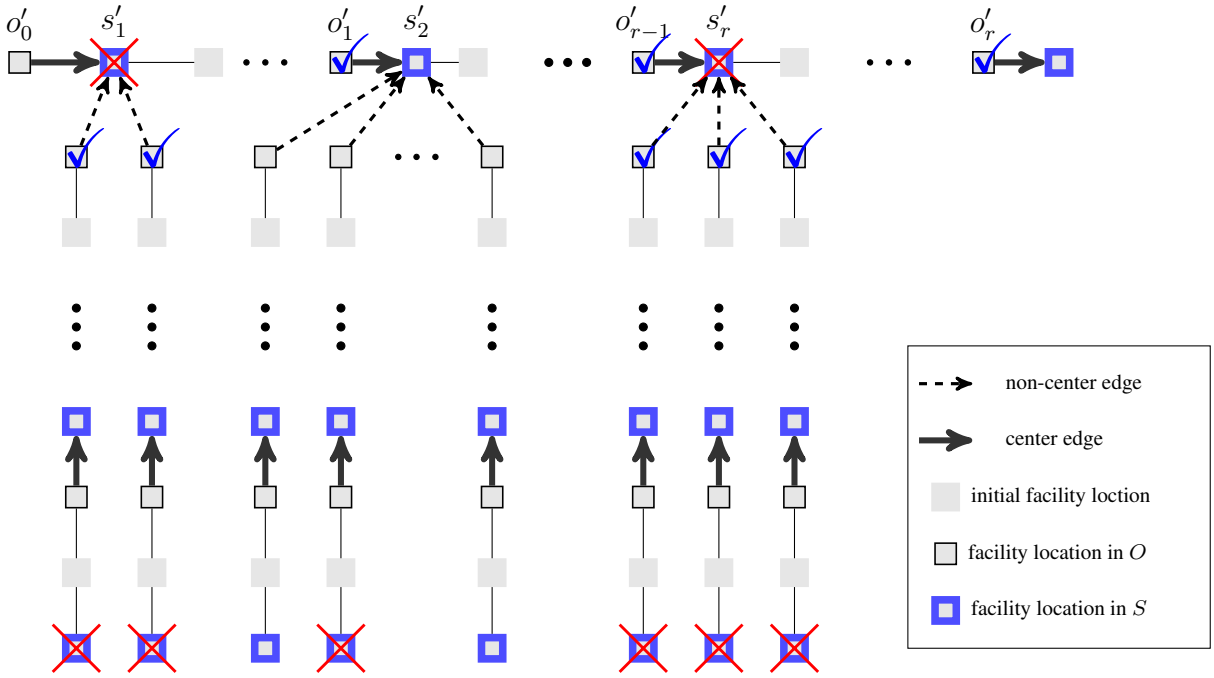


Figure 2.5: Interval-swap for consecutive $S_0 \cup S_1$ nodes. In this example, s'_1, s'_r capture less than t centers and s'_2 captures more than t centers. Nodes that are swapped out, are identified by a cross ($\times$), and nodes that are swapped in are identified by a checkmark ($\checkmark$).

Proof. Let Z be the path in $\mathcal{P}$ or cycle in $\mathcal{C}$ such that $S' \cup O' \subseteq Z$.

We first bound the increase in movement cost. The interval swap can be viewed as a collection of simultaneous $\mathrm{swap}(X_q, Y_q)$, $q = 1, \ldots, r$ moves. If $X_q = \{\mathsf{start}(P)\}$ for a random path $P \in \mathcal{P}_c(s'_q)$, the movement-cost increase can be broken into two parts. We do a shift along P, but move the last initial location on P to s'_q, and then do shift on Z from s'_q to o'_q. So the expected

movement-cost change is at most

$$\frac{1}{|\mathcal{P}_c(s'_q)|}\cdot\sum_{P\in\mathcal{P}_c(s'_q)}\big(\mathsf{shift}(P)+c(\mathsf{end}(P),s'_q)\big)+\mathsf{shift}(s'_q,o'_q)\le\frac{1}{|\mathcal{P}_c(s'_q)|}\cdot\sum_{P\in\mathcal{P}_c(s'_q),i\in P}2f^*_i+\mathsf{shift}(s'_q,o'_q)$$

which is at most $\sum_{P\in\mathcal{P}_c(s'_q),i\in P}2f^*_i+\mathsf{shift}(s'_q,o'_q)$. Similarly, if $|\mathsf{cap}(s'_q)|\le t$, we can break the movement-cost increase into $\mathsf{shift}(P)\le\sum_{i\in P}2f^*_i$ for all $P\in\mathcal{P}_c(s'_q)$ and $\mathsf{shift}(s'_q,o'_q)$. Thus, the total increase in movement cost is at most

$$\sum_{q=1}^{r}\mathsf{shift}(s'_q,o'_q)+\sum_{P\in\mathcal{P}_c(S'),i\in P}2f^*_i. \tag{2.4}$$

Next, we bound the change in assignment cost by reassigning clients in $\widehat{D}=D^*(O')\cup D(X)$ as follows. We assign each client $j\in D^*(O')$ to $\sigma^*(j)$. If $|\mathsf{cap}(s'_1)|>t$, then $s'_1\notin X$. For every client $j\in\widehat{D}\setminus(D^*(O'))$, observe that either $\sigma^*(j)\in Y$ or $\sigma(\sigma^*(j))\notin X$. To see this, let $o=\sigma^*(j)$ and $s=\sigma(o)$. If $o\notin Y$ then $s\notin S'\cap S_3$; also $s\notin T(S')$, and so $s\notin X$. So we assign j to $\sigma^*(j)$ if $\sigma^*(j)\in Y$ and to $\sigma(\sigma^*(j))$ otherwise; the change in assignment cost of j is at most $2c^*_j$ (Lemma 2.3.1).

Now suppose $|\mathsf{cap}(s'_1)|\le t$, so $s'_1\in X$. For each $j\in\widehat{D}\setminus(D^*(O')\cup D^*(o'_o))$, we again have $\sigma^*(j)\in Y$ or $\sigma(\sigma^*(j))\notin X$, and we assign j to $\sigma^*(j)$ if $\sigma^*(j)\in Y$ and to $\sigma(\sigma^*(j))$ otherwise. We assign every $j\in\widehat{D}\cap D^*(o'_0)$ to $s_{\hat{\imath}}$ (recall that $o'_0=o_{\hat{\imath}}$), and overestimate the resulting change in assignment cost by $\sum_{j\in D^*(o'_o)}(c^*_j+f^*_{\hat{\imath}}+f_{\hat{\imath}})$. Finally, note that we reassign a client $j\in D(T(S'\setminus S_3))\setminus D^*(O')$ with probability at most $\frac{1}{t}$ (since $\sigma(j)\in X$ with probability at most $\frac{1}{t}$). So taking into account all cases, we can bound the change in total assignment cost by

$$\sum_{j\in D^*(O')}(c^*_j-c_j)+\sum_{j\in D(T(S'\cap S_3)\cup(S'\cap S_3))}2c^*_j+\sum_{j\in D(T(S'\setminus S_3))}\tfrac{2c^*_j}{t}+\mathbf{1}(s'_1)\sum_{j\in D^*(o'_0)}(f^*_{\hat{\imath}}+f_{\hat{\imath}}+c^*_j). \tag{2.5}$$

In (2.5), we are double-counting clients in $D\big(T(S')\cup(S'\cup S_3)\big)\cap D^*(O')$. We are also overestimating the change in assignment cost of a client $j\in D(X)\cap D^*(o'_0)$ since we include both the $\mathbf{1}(s'_1)(c^*_j+f^*_{\hat{\imath}}+f_{\hat{\imath}})$ term, and the $2c^*_j$ or $\frac{2c^*_j}{t}$ terms. Adding (2.4) and (2.5) yields the lemma. □

Notice that Lemma 2.4.3 immediately translates to a bound on the assignment cost of the clients in $D^*(\mathsf{center}(s))$ for $s\in S_2$. In contrast, it is quite unclear how Lemma 2.4.4 may be useful, since the expression $\sum_{j\in D^*(o'_0)}(f^*_{\hat{\imath}}+f_{\hat{\imath}})$ in the RHS of (2.3) may be as large as $t(f^*_{\hat{\imath}}+f_{\hat{\imath}})$ (but no more since $|D^*(o'_0)|\le t$ if $\mathbf{1}(s'_1)=1$) and it is unclear how to cancel the contribution of $f_{\hat{\imath}}$

on the RHS. One of the novelties of our analysis is that we show how to *amortize* such expensive terms and make their contribution negligible by considering multiple interval swaps. We cover each path or cycle Z in t^2 different ways using intervals comprising consecutive locations from $S_0 \cup S_1$. We then argue that averaging, over these t^2 covering ways, the inequalities obtained from the corresponding interval swaps yields (among other things) a good bound on the movement-cost of the $(S_0 \cup S_1)$-locations on Z and the assignment cost of the clients they serve.

Lemma 2.4.5. *Let $Z \in \mathcal{P} \cup \mathcal{C}$, $S' = \{s'_1, \ldots, s'_r\} = S_1 \cap Z$, where s'_{q+1} is the next S_1-location on Z after s'_q, and $O' = \{\mathsf{center}(s'_1), \ldots, \mathsf{center}(s'_r)\}$. Let $o'_r = \mathsf{end}(Z)$ if $Z \in \mathcal{P}$ and $\mathsf{center}(s'_1)$ otherwise. For $r \geq t^2$,*

$$\begin{aligned} 0 \;\leq\; & \sum_{i \in Z} \Big(\tfrac{t+1}{t} \cdot f^*_i - \tfrac{t-1}{t} f_i \Big) + \sum_{P \in \mathcal{P}_c(S'), i \in P} 2f^*_i + \sum_{j \in D^*(Z \cap O)} \Big(\tfrac{1}{t} \cdot c_j + \tfrac{t+1}{t^2} \cdot c^*_j \Big) \\ & + \sum_{j \in D^*(O' \cup \{o'_r\})} (c^*_j - c_j) + \sum_{j \in D(T(Z \cap S_3) \cup (Z \cap S_3))} 2c^*_j + \sum_{j \in D(T(S' \setminus S_3))} \tfrac{2c^*_j}{t}. \end{aligned} \tag{2.6}$$

Proof. We first define formally an interval of (at most) t^2 consecutive $(S_0 \cup S_1)$ locations along Z. As before, let $o'_{q-1} = \mathsf{center}(s'_q)$ for $q = 1, \ldots, r$. For a path Z, define $s'_q = \mathsf{start}(Z)$ for $q \leq 0$ and $s'_q = \mathsf{nil}$ for $q > r$. Also define $o'_q = o'_0$ for $q \leq 0$ and $o'_q = \mathsf{end}(Z)$ for $q \geq r$. If Z is a cycle, we let our indices wrap around and be $\bmod r$, i.e., $s'_q = s'_{q \bmod r}$, $o'_q = o'_{q \bmod r}$ for all q (so $o'_r = o'_0 = \mathsf{center}(s'_1)$).

For $1 - t^2 \leq h \leq r$, define $S'_h = \{s'_h, s'_{h+1}, \ldots, s'_{h+t^2-1}\}$ to be an interval of length at most t^2 on Z. Define $O'_h = \{o'_h, o'_{h+1}, \ldots, o'_{h+t^2-1}\}$. Note that we have $1 \leq |S'_h| = |O'_h| \leq t^2$ if Z is a path, and $|S'_h| = |O'_h| = t^2$ if Z is a cycle. Consider the collection of intervals, $\{S'_{-t^2+1}, S'_{-t^2+2}, \cdots, S'_r\}$. For each S'_h, O'_h, where $-t^2 + 1 \leq h \leq r$, we consider the interval swap (X_h, Y_h) corresponding to S'_h, O'_h. We add the inequalities $\frac{1}{t^2} \times$(2.3) for all such h. Since each $s' \in S' \cup \{s'_0\}$ participates in exactly t^2 such inequalities, and each $s'_h \in S'$ is the start of only the interval S'_h, we obtain the following.

$$\begin{aligned} 0 \;\leq\; & \sum_{q=0}^{r} \frac{1}{t^2} \cdot t^2 \cdot \mathsf{shift}(s'_q, o'_q) + \sum_{P \in \mathcal{P}_c(S'), i \in P} \frac{1}{t^2} \cdot t^2 \cdot 2f^*_i \\ & + \sum_{j \in D^*(O' \cup \{o'_r\}))} \frac{1}{t^2} \cdot t^2 \cdot (c^*_j - c_j) + \sum_{j \in D(T(Z \cap S_3) \cup (Z \cap S_3))} \frac{1}{t^2} \cdot t^2 \cdot 2c^*_j + \sum_{j \in D(T(S' \setminus S_3))} \frac{1}{t^2} \cdot t^2 \cdot \frac{2c^*_j}{t} \\ & + \sum_{i : \sigma(o_i) \in Z} \mathbf{1}(\sigma(o_i)) \cdot \frac{1}{t^2} \cdot \sum_{j \in D^*(o_i)} (f^*_i + f_i + c^*_j). \end{aligned} \tag{2.7}$$

Notice that the S-locations other than s'_q on the $s'_q \rightsquigarrow o'_q$ sub-paths of Z lie in S_2, and for each i such that $\sigma(o_i) \in Z \cap S_2$, we have $c(o_i, \sigma(o_i)) \leq \sum_{j \in D^*(o_i)} \frac{c_j + c_j^*}{t}$. Thus, using (2.1), we have

$$\sum_{q=0}^{r} \mathsf{shift}(s'_q, o'_q) = \sum_{i \in Z} (f_i^* - f_i) + \sum_{i:\sigma(o_i) \in Z \cap S_2} c(o_i, \sigma(o_i)) \leq \sum_{i \in Z} (f_i^* - f_i) + \sum_{j \in D^*(Z \cap O)} \frac{c_j + c_j^*}{t}. \tag{2.8}$$

Since $\mathbf{1}(\sigma(o)) = 1$ means that $\sigma(o) \in S_3$, and so $|D^*(o)| \leq t$, we have

$$\sum_{i:\sigma(o_i) \in Z \cap S_3} \sum_{j \in D^*(o_i)} \frac{f_i^* + f_i + c_j^*}{t^2} \leq \sum_{i \in Z} \Big(\frac{f_i^* + f_i}{t} + \frac{\sum_{j \in D^*(o_i)} c_j^*}{t^2} \Big) \leq \sum_{i \in Z} \frac{f_i^* + f_i}{t} + \sum_{j \in D^*(Z \cap O)} \frac{c_j^*}{t^2}. \tag{2.9}$$

Incorporating (2.8) and (2.9) in (2.7), and simplifying yields the desired inequality. □

For a path or cycle Z where $|S_1 \cap Z| < t^2$, we obtain an inequality similar to (2.6). Since we can now cover Z with a single interval, we never have a client j such that none of $\sigma(j)$, $\sigma^*(j)$, $\sigma(\sigma^*(j))$ are in our new set of final locations. So the resulting inequality does not have any $\frac{f_i^* + f_i}{t} + \frac{c_j^*}{t^2}$ terms.

Lemma 2.4.6. *Let $Z \in \mathcal{P} \cup \mathcal{C}$, $S' = \{s'_1, \ldots, s'_r\} = S_1 \cap Z$, where s'_{q+1} is the next S_1-location on Z after s'_q, and $O' = \{\mathsf{center}(s'_1), \ldots, \mathsf{center}(s'_r)\}$. Let $o'_r = \mathsf{end}(Z)$ if $Z \in \mathcal{P}$ and $\mathsf{center}(s'_1)$ otherwise. For $r < t^2$,*

$$\begin{aligned} 0 \leq & \sum_{i \in Z} (f_i^* - f_i) + \sum_{P \in \mathcal{P}_c(S'), i \in P} 2f_i^* + \sum_{j \in D^*(Z \cap O)} \frac{c_j + c_j^*}{t} \\ & + \sum_{j \in D^*(O' \cup \{o'_r\})} (c_j^* - c_j) + \sum_{j \in D(T(Z \cap S_3) \cup (Z \cap S_3))} 2c_j^* + \sum_{j \in D(T(S' \setminus S_3))} \frac{2c_j^*}{t}. \end{aligned} \tag{2.10}$$

Proof. The proof is similar to that of Lemma 2.4.5, except that since we can cover Z with a single interval, we only need to consider a single (multi-location) swap. We consider two cases for clarity.

1. ***Z* is a path, or $r > 0$.** As before, let $o'_{q-1} = \mathsf{center}(s'_q)$ for $q = 1, \ldots, r$. If Z is a path, define $s'_0 = \mathsf{start}(Z)$. If Z is a cycle, we again set $s'_q = s'_{q \bmod r}$, $o'_q = o'_{q \bmod r}$ for all q. Consider the interval swap (X, Y) corresponding to $S' \cup \{s'_0\}, O' \cup \{o'_r\}$. The inequality generated by this is similar to (2.3) except that we do not have any $(f_{\hat{i}}^* + f_{\hat{i}} + c_j^*)$ terms since

for client $j \in D(X) \cup D^*(Y)$, we always have that either $\sigma^*(j) \in Y$ or $\sigma(\sigma^*(j)) \notin X$. Thus, (2.3) translates to the following.

$$0 \;\le\; \sum_{q=0}^{r} \mathsf{shift}(s'_q, o'_q) + \sum_{P \in \mathcal{P}_c(S'), i \in P} 2f_i^* \\ + \sum_{j \in D^*(O' \cup \{o'_r\})} (c_j^* - c_j) + \sum_{j \in D(T(Z \cap S_3) \cup (Z \cap S_3))} 2c_j^* + \sum_{j \in D(T(S' \setminus S_3))} \tfrac{2c_j^*}{t}.$$

Substituting $\sum_{q=0}^{r} \mathsf{shift}(s'_q, o'_q) \le \sum_{i \in Z}(f_i^* - f_i) + \sum_{j \in D^*(Z \cap O)} \frac{c_j + c_j^*}{t}$ as in (2.8) yields the stated inequality.

2. ***Z* is a cycle with $r = 0$.** Here, Lemma 2.4.2 yields $0 \le \sum_{i \in Z}(f_i^* - f_i) + \sum_{j \in D^*(Z \cap O)} \frac{c_j + c_j^*}{t}$ (which is the special case of the earlier inequality with $s'_0 = \mathsf{nil} = o'_r,\ S' = O' = Z \cap S_3 = \emptyset$).

□

Proof of Theorem 2.4.1. We consider the following set of swaps.

A_1. For every $s \in S_2$, the move $\mathrm{swap}\big(\{s\} \cup T(s), \mathsf{cap}(s)\big)$.

A_2. For every path or cycle Z with $|Z \cap S_1| \ge t^2$, the $\frac{1}{t^2}$-weighted interval swaps as defined in Lemma 2.4.5.

A_3. For every path or cycle Z with $|Z \cap S_1| < t^2$, the interval swap defined in Lemma 2.4.6.

Notice that every location $o \in O$ is swapped in to an extent of at least 1 and at most 2. (By "extent" we mean the total weight of the inequalities involving o.) To see this, suppose first $o = \mathsf{end}(Z)$ for some path Z, then o is involved to an extent of 1 in the interval swaps for Z in A_2 or A_3. In this case, we say that the interval swap for Z is *responsible* for o. Additionally, if $s = \sigma(o) \in S_2$, then o belongs to the multi-swap for s in A_1, else if $s \in S_3$ then o is part of the interval swap for the path/cycle containing s in A_2 or A_3. Now suppose $o = \mathsf{center}(s)$. If $s \in S_2$, then o is included in the multi-swap for s in A_1. We say that this multi-swap is responsible for o. If $s \in S_1$, then o is included in the interval swap for the path/cycle containing s in A_2 or A_3, and we say that this interval swap is responsible for o.

Consider the compound inequality obtained by summing (2.2), (2.6), and (2.10) corresponding to the moves considered in A_1, A_2, and A_3 respectively. The LHS of this inequality is 0. We now need to do some bookkeeping to bound the coefficients of the f_i^*, f_i, c_j^*, c_j terms on the

RHS. We ignore $o(1)$ coefficients like $\frac{1}{t}$, $\frac{1}{t^2}$ in this bookkeeping, since for a given $\{f_i^*, f_i, c_j^*, c_j\}$ term, such coefficients appear in only a constant number of inequalities, so they have an $o(1)$ effect overall. Let F and C denote respectively the movement- and assignment- cost of the local optimum.

Contribution from the c_j^* and c_j terms. First, observe that for each $o \in O$, we have labeled exactly one move involving o as being responsible for it. Consider a client $j \in D(s) \cap D^*(o)$. Observe that c_j^* or c_j terms appear (with a $\Theta(1)$-coefficient) in an inequality generated by a move if (i) j is reassigned because the move is responsible for o; or (ii) s is swapped out (to an extent of 1) by the move (so this excludes the case where $s \in T(s')$, $s' \in S_1 \setminus S_3$ and the move is the interval swap for the path containing s'). If (i) applies, then the inequality generates the term $(c_j^* - c_j)$. If (ii) applies then the term $2c_j^*$ appears in the inequality. Finally, note that there are at most two inequalities for which (ii) applies:

- If $s = \mathsf{start}(Z) \in S_0$, then (ii) applies for the interval-swap move for Z. If $s' = \sigma(\mathsf{end}(Z)) \in S_2 \cup S_3$, then (ii) again applies, for the multi-swap move for s' if $s' \in S_2$, or for the interval swap for the path containing s' if $s' \in S_3$.
- If $s \in S_1 \cap S_3$, then (ii) applies for the interval swap for the path containing s.
- If $s \in S_2$, then (ii) applies for the multi-swap move for s.

So overall, we get a $5c_j^* - c_j$ contribution to the RHS, the bottleneck being the two inequalities for which (ii) applies when $s \in \mathsf{start}(Z)$ and $\sigma(\mathsf{end}(Z)) \in S_2 \cup S_3$.

Contribution from the f_i^* and f_i terms. For every $i \in \mathcal{F}$, the expression $(f_i^* - f_i)$ is counted once in the RHS of the inequality (2.6) or (2.10) for the unique path or cycle Z containing i. The total contribution of all these terms is therefore, $F^* - F$. The remaining contribution comes from expressions of the form $\sum_{P \in \mathcal{P}_c(s), i \in P} 2f_i^*$ on the RHS of (2.2), (2.6), and (2.10). The paths P involved in these expressions come from $\mathcal{P}_c(S_2) \cup \big(\bigcup_{Z \in \mathcal{P} \cup \mathcal{C}} \mathcal{P}_c(Z \cap S_3)\big) \subseteq \mathcal{P}$. Therefore, the total contribution of these terms is at most $2F^*$.

Thus, we obtain the compound inequality

$$0 \leq \big(5 + o(1)\big)C^* + \big(3 + o(1)\big)F^* - \big(1 - o(1)\big)(F + C) \tag{2.11}$$

where the $o(1)$ terms are $O\big(\frac{1}{t}\big) = O\big(p^{-1/3}\big)$. This shows that $F + C \leq \big(3 + o(1)\big)F^* + \big(5 + o(1)\big)C^*$. □

2.4.2 Polynomial time local search approach

A generic local search algorithm may not terminate in polynomial time. In order to make the algorithm run in polynomial time, the usual trick is to consider a local search move only if it improves the cost of the solution substantially. More precisely, let polynomial Q be a suitable integer which is polynomial in the size of the input. In each local step, a current solution S is updated to a solution S' in the neighborhood of S only if $\mathsf{MFL}(S) - \mathsf{MFL}(S') \geq \frac{\epsilon}{Q} \cdot \mathsf{MFL}(S)$ where ϵ is a constant. So this new local search can be abstracted as follows:

Algorithm 1 Local search approach

1: $S \leftarrow$ an arbitrary feasible solution.
2: **while** $\exists S'$ such that $|S \setminus S'| = |S' \setminus S| \leq p$ and $\mathsf{MFL}(S') < (1 - \epsilon/Q)\mathsf{MFL}(S)$ **do**
3: $\quad S \leftarrow S'$
4: **return** S.

Since the cost of solution improves by a factor $\frac{\epsilon}{Q}$ in each step of the algorithm, then the algorithm terminates in at most $\log(\mathsf{MFL}(S_0)/\mathsf{MFL}(O))/\log\frac{1}{1-\epsilon/Q}$ where O denotes the optimal solution and S_0 denotes the initial solution, which is polynomial in the input size. Now since Q is polynomial in the input size and each step considers at most $O(n^p)$ sets in the neighborhood of S, the algorithm will have polynomial running time.

Now we argue that this modification does not significantly affect the claimed approximation ratio. The above modification influences the inequalities that we used our analysis and now we have $-\frac{\epsilon}{Q} \cdot (F + C)$ instead of 0 in the left-hand side of each inequality for a non-improving local move. This means that on the left-hand side of (2.11), we get an extra term $-\frac{\epsilon}{Q} \cdot (F + C)N$, where N is the total weight placed on the inequalities generated from the various swap moves whose suitable linear combination yields (2.11). Note the weight for each inequality is at most 1 and there are at most $O(n^p)$ inequalities, so as long as $Q \geq N = O(n^p)$, we only lose a factor of $(1 + \epsilon)$ in the approximation factor.

2.5 Improved analysis leading to a 3-approximation

In this section, we improve the analysis from Section 2.4. Specifically, we prove the following theorem.

Theorem 2.5.1. *The cost of a locally-optimal solution using p swaps is at most $3+O\left(\sqrt{\frac{\log\log p}{\log p}}\right)$ times the optimum solution cost.*

To gain some intuition behind this tighter analysis, note that the *only* reason we lost a factor of 5 in the previous analysis was because there could be locations $s = \mathsf{start}(Z) \in S_0$ that are swapped out to an extent of 2; hence, there could be clients $j \in D(s)$ for which we "pay" $2c_j^*$ each time s is swapped out, and also pay an additional $c_j^* - c_j$ term when $\sigma^*(j)$ is swapped in. To improve the analysis, we will consider a set of test swaps that swap out each location in S to an extent of $1 + o(1)$.

The aforementioned bad case happens only when $s' = \sigma(\mathsf{end}(Z)) \in S_2 \cup S_3$, because when we close (i.e., swap out) s' as part of an interval swap or a multi-swap, we open (i.e., swap in) all the locations in $H(s')$, and we achieve this via path swaps (i.e., shift moves) along paths in $\mathcal{P}_c(s')$ that swap out locations in $T(s')$ (for a second time). The main idea behind our refined analysis is to not perform such path swaps, but instead to "recursively" start an interval swap on each path in $\mathcal{P}_c(s')$. Of course, we cannot carry out this recursion to an arbitrary depth (since we can only swap a bounded number of locations), so we terminate the recursion at a depth of t^2. So, whereas an interval swap included at most t^2 S_1-locations on the main path or cycle Z, we now consider a "subtree" swap obtained by aggregating interval swaps on the paths in $\bigcup \mathcal{P}_c(Z \cap S_3)$. A subtree swap can be viewed as a bounded-depth recursion tree where each leaf to root path encounters at most t^2 locations in S_1. Because we no longer initiate path swaps for S_3-locations, a leaf location $s'' \in S_3$ in this recursion tree will not have any locations in $\mathsf{cap}(s'')$ opened. But we will slightly redefine the S_1, S_2, S_3 sets to ensure that $|D^*(\mathsf{cap}(s''))| \leq t$, and use the same trick that we did with interval swaps in Section 2.4: we *average* over different sets of subtree swaps (like we did with interval swaps in Section 2.4) to ensure that s'' is a leaf location with probability at most $\frac{1}{t^2}$. This ensures that we incur, to an extent of at most $\frac{1}{t}$, the cost $f_i^* + f_i + c(o_i, s'')$, where $o_i = \mathsf{center}(s'')$, for moving j with $\sigma(\sigma^*(j)) = s''$ from s'' to s_i.

Notation. Let t be an integer such that $p \geq t^2 t^{t^2} + 1$. We prove that the local-search algorithm has approximation ratio $3 + O(t^{-1})$. We redefine S_0, S_1, S_2 and S_3 as follows.

- $S_0 = \{s \in S : |\mathsf{cap}(s)| = 0\}$.
- $S_1 = \{s \in S \setminus S_0 : |D^*(\mathsf{cap}(s))| \leq t \text{ or } |\mathsf{cap}(s)| > t\}$.
- $S_2 = \{s \in S : |D^*(\mathsf{cap}(s))| > t,\ |\mathsf{cap}(s)| \leq t\}$.
- $S_3 = S_0 \cup \{s \in S_1 : |\mathsf{cap}(s)| \leq t\}$.

Clearly, $S = S_0 \cup S_1 \cup S_2$. We also redefine $\mathsf{center}(s)$ to be the location in $\mathsf{cap}(s)$ *closest to* s.

Claim 2.5.2. *Let $s \in S_2$ and $o = \mathsf{center}(s)$. Then $c(s, o) \leq \frac{1}{t} \sum_{j \in D^*(\mathsf{cap}(s))} (c_j + c_j^*)$.*

Proof. We have $c(s,o) \le c(s,o')$ for any $o' \in \mathsf{cap}(s)$, and $c(s,o') \le c_j + c_j^*$ for any $j \in D^*(o')$. Therefore, $c(s,o) \le c_j + c_j^*$ for any $j \in D^*(\mathsf{cap}(s))$, and the claim follows since $|D^*(\mathsf{cap}(s))| > t$ as $s \in S_2$. □

It will be more convenient to work with the digraph $H = (\mathcal{F}, E)$ obtained from $\widehat{G}$ by contracting each triple $\{s_i, i, o_i\}$ of nodes associated with a facility i into a single node that we also denote by i. Thus, (i, i') is an arc in E if $\sigma(o_i) = s_{i'}$ (it may be that $i = i'$). Note that H may have self loops, and each node in H has outdegree exactly 1 (counting self-loops) so each component of H looks like a tree with all edges oriented toward the root, except the root is in fact a directed cycle (possibly a self-loop). Figure 2.6 illustrates this graph.

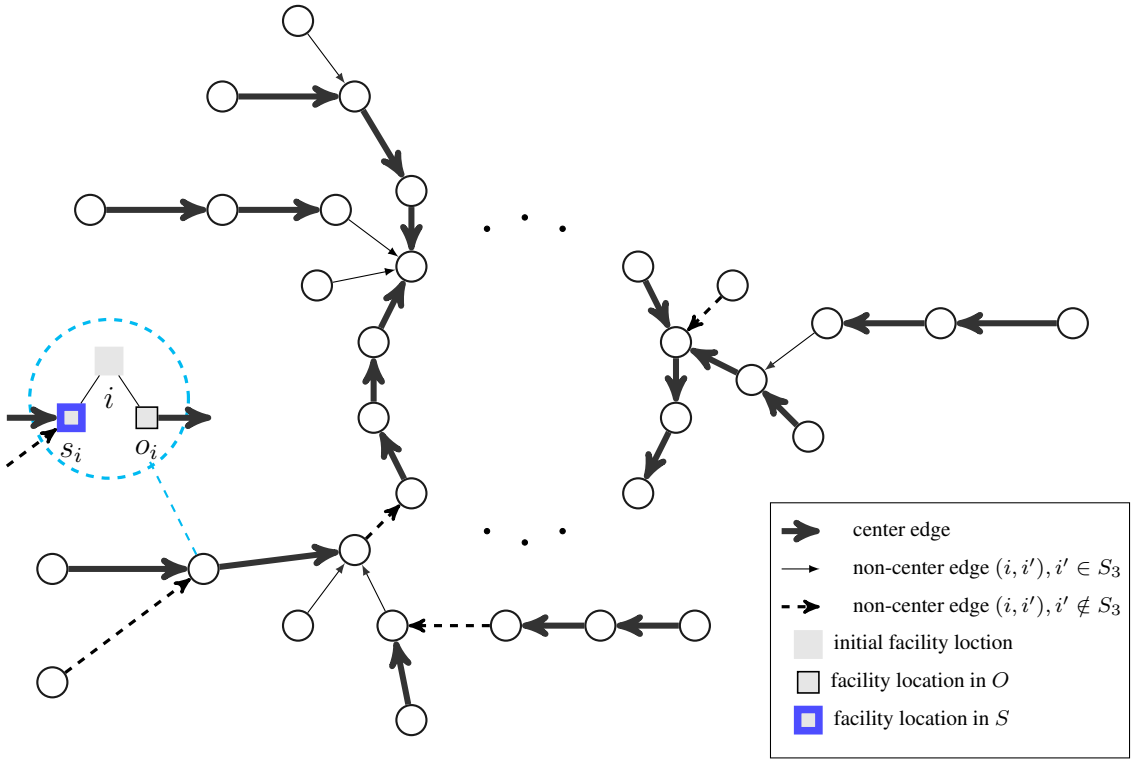


Figure 2.6: Example of graph H obtained from $\widehat{G}$ by contracting each triple (s_i, i, o_i) of nodes associated with a facility i. Edges of H can be partitioned into three categories: center edge (shown by a thick solid arrow), non-center edge ending in S_3 node (shown by a thin solid arrow), non-center edge not ending in S_3 node (shown by a dashed edge). In construction of H^* from H, the dashed edges are removed.

For brevity, we say that an edge (i, i') in H is a *center* edge if $o_i = \mathsf{center}(s_{i'})$. In the arc set $E' = \{(i, i') \in E : o_i = \mathsf{center}(s_{i'})\}$, each node has indegree and outdegree at most 1, so E' consists of a collection of node-disjoint paths $\mathcal{P}$ and cycles $\mathcal{C}$. For a facility $i \in \mathcal{P}$, let $\mathcal{P}(i)$ denote the unique path in $\mathcal{P}$ containing i. Let $\mathsf{start}(i)$ and $\mathsf{end}(i)$ denote the start and end facilities of $\mathcal{P}(i)$ respectively. For distinct facilities i, i' with $s_i, s_{i'} \in S_0$, the paths $\mathcal{P}(i)$ and $\mathcal{P}(i')$ are clearly vertex disjoint.

Now define $H^* = \big(\mathcal{F}, E' \cup \{(i, i') : s_{i'} \in S_3, \sigma(o_i) = s_{i'}\}\big)$; that is, H^* is the subgraph of H with node-set $\mathcal{F}$ and edges (i, i') of E where (i, i') is a center edge or $s_{i'} \in S_3$. Call a node i of H^* a *root* if i has no outgoing arc or i lies on a directed cycle in H^*.

We consider an integer $1 \le l \le t^2$ and describe a set of swaps for each index l. The inequalities for the swaps for different l will be averaged in the final analysis. We obtain H_l^* by deleting the edges (i, i') of H^* (see Figure 2.7) where: i is not on a cycle, $s_{i'} \in S_1$, and the number of facilities i'' with $s_{i''} \in S_1$ on the path between i' and the closest root of its component in H^* (both included) is congruent to $l \bmod t^2$ (note that when an H^* component is rooted at a cycle, we have multiple roots, and this is the reason behind using the term the "closest" root). We define a *subtree* of H_l^* to be an acyclic component of H_l^*, or a component that results by deleting the edges of the cycle contained in a component of H_l^*.

For a facility i, define $\mathsf{cand}(i) = \{i' : o_{i'} \in \mathsf{cap}(s_i) \setminus \{\mathsf{center}(s_i)\},\ \nexists i \rightsquigarrow i' \text{ path in } H^*\}$. Note that $|\mathsf{cand}(i)| \ge |\mathsf{cap}(s_i)| - 2$. The reason we define $\mathsf{cand}(i)$ is that we will sometimes perform a shift along some path $Z \in \mathcal{P}_c(s_i)$ to reassign the facilities on Z but we will not want this to interfere with the operations in the subtree of H_l^* containing i. For a facility i with $s_i \notin S_2$, let $\mathsf{next}(i)$ be the facility obtained as follows. Follow the unique walk from i in H using only center edges until the walk reaches a node i' with either no outgoing center edge, or the unique (i', i'') center edge satisfies $s_{i''} \in S_1$; we set $\mathsf{next}(i) = i'$.

Claim 2.5.3. *The number of facilities i with $s_i \in S_0 \cup S_1$ in any subtree of H_l^* is at most t^{t^2}.*

Proof. The facilities i in such a subtree that are in S_2 have indegree and outdegree at most 1. Shortcutting past these facilities yields a tree with depth at most t^2 and branching factor at most t. □

The test swaps. For a subtree T of H_l^*, we describe a set of nodes X_T to be swapped out and a set of nodes Y_T to be swapped in with $|X_T| = |Y_T| \le t^{t^2}$. We do not actually perform these swaps yet to generate the inequalities since we will have to combine some of these swaps for various components.

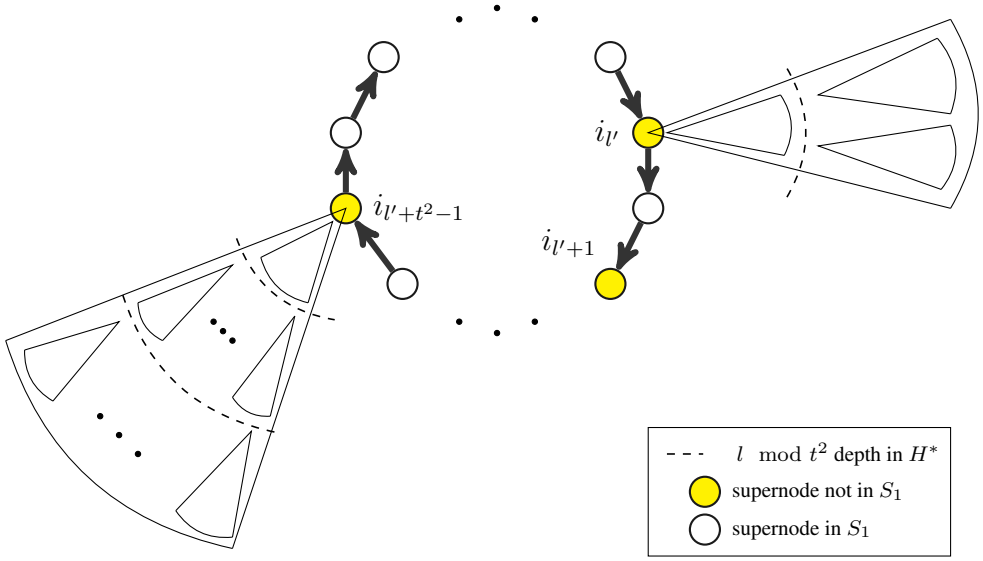


Figure 2.7: Example of construction of H_l^* from H^*: we remove edges (i, i') such that i is not in a cycle, $s_{i'} \in S_1$, and the number of i'' nodes with $s_{i''} \in S_1$ from i' to the closest root of its component is congruent to $l \mod t^2$. In this figure, all the edges on the dashed curved lines are removed.

For each $i \in T$ with $s_i \in S_0 \cup S_1$, we add the following location in S to X_T: if $s_i \in S_3$ we add s_i to X_T; otherwise (so $s_i \in S_1 \setminus S_3$), we choose any single $i' \in \mathsf{cand}(i)$ uniformly at random and add $s_{\mathsf{start}(i')}$ to X_T. We also add $o_{\mathsf{next}(i)}$ to Y_T.

When we say perform $\mathrm{swap}(X_T, Y_T)$ (see Figure 2.8 for an example), we specifically mean the following reassignment of facilities. For $s_i \in X_T$ with $s_i \in S_3$, we perform $\mathsf{shift}(s_i, o_{\mathsf{next}(i)})$. For $s_i \in X_T$ with $s_i \in S_1 \setminus S_3$, say i' is the facility in $\mathsf{cand}(i)$ for which $s_{\mathsf{start}(i')}$ was added to X_T. Then we perform $\mathsf{shift}(s_{\mathsf{start}(i')}, o_{i'})$, move facility i' from $o_{i'}$ to s_i, and finally perform $\mathsf{shift}(s_i, o_{\mathsf{next}(i)})$. As always, each client is then assigned to its nearest final location. Lemma 2.5.4 implies that these shift operations charge different portions of the local and global optimum.

Lemma 2.5.4. *For a subtree T, all of the shift operations described for $\mathrm{swap}(X_T, Y_T)$ have their associated paths being vertex disjoint.*

Proof. For any subtree T, the paths between s_i and $o_{\mathsf{next}(i)}$ for the facilities $i \in T$ with $s_i \in S_0 \cup S_1$ are vertex-disjoint by definition of $\mathsf{next}(i)$. Also, for any two distinct $i_1, i_2 \in T$, and any $i \in \mathsf{cand}(i_1),\ i' \in \mathsf{cand}(i_2)$, we have $\mathsf{start}(i) \neq \mathsf{start}(i')$, and so their associated paths $\mathcal{P}(i)$ and $\mathcal{P}(i')$ are also vertex-disjoint.

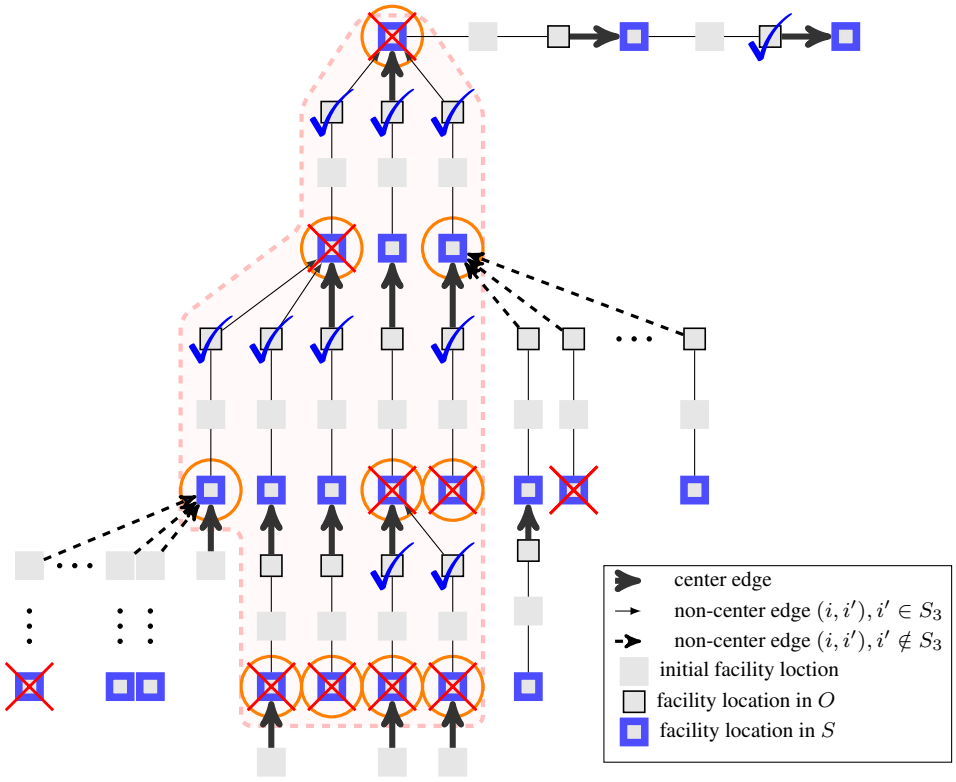


Figure 2.8: Example of $\mathrm{swap}(X_T, Y_T)$ in a subtree T of H_l^*. In this example, the subtree T includes the nodes in the dashed-bordered shaded area. The S_1-nodes in T are surrounded by a circle, nodes in X_T are identified by a cross($\times$) and nodes in Y_T are identified by a checkmark($\checkmark$).

Finally, consider any $i \in T$ with $s_i \in S_0 \cup S_1$, and $i'' \in T$ (i'' could be i) with $s_{i''} \in S_1 \setminus S_3$. Consider the paths involved in $\mathrm{swap}(s_i, o_{\mathsf{next}(i)})$ and $\mathrm{swap}(s_{\mathsf{start}(i')}, o_{i'})$, where $i' \in \mathsf{cand}(i'')$. Note that both of these paths consist of only center edges. Therefore, since each facility has at most one incoming and one outgoing center edge, and $i' = \mathsf{end}(i')$, if these paths are not vertex-disjoint, then it must be that the path involved in $\mathrm{swap}(s_i, o_{\mathsf{next}(i)})$ is a subpath of the path involved in $\mathrm{swap}(s_{\mathsf{start}(i')}, o_{i'})$. This means that i and i', and hence, i, i', i'', are all in the same component of H^*. Also, the edge (i', i'') is not in H^* so i' is the root of its component in H^*. But then there is a path from i'' to i', which contradicts that $i' \in \mathsf{cand}(i'')$. □

We need to coordinate the swaps for the various subtrees of H_l^*. Consider a component Z in H^*. Let $C = \emptyset$ if Z is rooted at a node, otherwise let C be its cycle of root nodes. Let

$i_1, \ldots, i_k$ be the facilities on C with $s_i \in S_1$, indexed by order of appearance on C starting from an arbitrary facility i_1 on C ($k = 0$ if $C = \emptyset$). We consider four kinds of swaps.

Type 1. If $1 \leq k \leq t^2$, simultaneously do $\mathrm{swap}(X_T, Y_T)$ for all subtrees T rooted at some $i \in C$ with $s_i \in S_1$.

Type 2. Otherwise, if $k > t^2$, define $I_{l'} = \{i_{l'}, i_{l'+1}, \ldots, i_{l'+t^2-1}\}$ for all $l' = 1, \ldots, k$ (where the indices are $\mathrm{mod} k$). Simultaneously perform $\mathrm{swap}(X_T, Y_T)$ for each subtree T rooted at a facility in $I_{l'}$. Reasoning similarly as in Lemma 2.5.4 and noting that the subtrees involved in a single type-1 or type-2 swap are all disjoint, we can see that all shift paths involved in a single type-1 or type-2 swap are vertex-disjoint.

Type 3. For each i with $s_i \in S_2$, simultaneously perform $\mathrm{swap}(X_T, Y_T)$ for all subtrees T rooted at some i' with $o_{i'} \in \mathsf{cap}(s_i) \setminus \{\mathsf{center}(s_i)\}$. At the same time, we also swap out s_i and swap in $o_{i''} = \mathsf{center}(s_i)$ for a total of at most $t^{t^2+1} + 1 \leq p$ swaps. It may be that some (at most one) shift path in this swap includes s_i, but then we just move i'' to $o_{i''}$ instead of s_i, and then move i according to the shift operation.

Type 4. Finally, for every other subtree T of H_l^* that was not swapped in the previous cases, perform $\mathrm{swap}(X_T, Y_T)$ on its own.

Analysis. We first bound the net client-assignment cost increase for any single one of these test swaps. So, fix one such swap, let $\{T_r\}_{r=1}^k, k \leq t^2$ be the set of subtrees involved in the swap, and let B denote the set of facilities i such that $o_i = \mathsf{center}(\sigma(o_i))$ and $\sigma(o_i)$ is closed during the swap while o_i is not opened. So B consists of facilities with a center edge to some leaf of some subtree T_r or, if the swap is of type 2, to the start of an interval $I_{l'}$. For this swap, let $C_1 = \{j \in \mathcal{D} : \sigma^*(j) \text{ is opened}\}$, $C_2 = D^*(\{o_i : i \in B\})$, and $C_3 = \{j : \sigma(j) = s_i \in S_0 \text{ and } \mathsf{end}(i) \in \bigcup_r \bigcup_{i' \in T_r} \mathsf{cand}(i')\}$.

Lemma 2.5.5. *The expected change in client-assignment cost for a test swap is at most* $\sum_{j \in C_1}(c_j^* - c_j) + \sum_{j \in C_2} 2c_j^* + \frac{1}{t-1}\sum_{j \in C_3} 2c_j^* + 2t\sum_{i \in B}(f_i^* + f_i)$. *Here, the expectation is over the random choices involved in selecting facilities from the appropriate* $\mathsf{cand}(.)$ *sets.*

Proof. After the swap, we move every $j \in C_1$ from $\sigma(j)$ to $\sigma^*(j)$ for a cost change of $c_j^* - c_j$. Every client in $j \in C_2 \cup C_3$ for which $\sigma(j)$ is closed is moved initially to $\sigma(\sigma^*(j))$ for a cost increase of at most $2c_j^*$.

Suppose i is such that $\sigma(o_i) = \sigma(\sigma^*(j))$ and $o_i = \mathsf{center}(\sigma(o_i))$. It may be that $\sigma(o_i)$ is still not open which means that $i \in B$. Note that either s_i or o_i is opened after the shift and we move every client that was moved to $\sigma(o_i)$ to s_i or o_i (whichever is open). This extra distance moved

is at most $f_i^* + f_i + c(o_i, \sigma(o_i)) \leq 2f_i^* + 2f_i$. Note that $i \in B$ implies that $\sigma(o_i) \in S_3$, otherwise $\sigma(o_i)$ would not have been closed down in the swap. So $|D^*(\mathsf{cap}(\sigma(o_i)))| \leq t$, by definition of S_3, and at most t clients will be moved to either s_i or o_i in this manner.

Finally, we note that while $j \in C_3$ may have $\sigma(j)$ being closed, this only happens with probability at most $\frac{1}{t-1}$. □

Now, we consider the following weightings of the swaps. First, for a particular index $1 \leq l \leq t^2$ we perform all type 1, 3, and 4 swaps. For a component of H_l^* containing a cycle C, we perform all type-2 swaps for the various intervals $I_{l'}$ for C and weight the client and facility cost change by $\frac{1}{t^2}$. Finally, these weighted bounds on the client and facility cost change are averaged over all $1 \leq l \leq t^2$.

Lemma 2.5.6. *The expected change in client-assignment cost under the weighting described above, is at most* $\sum_j 3c_j^* - c_j + O\left(\frac{1}{t}\right)\left(\sum_i (f_i^* + f_i) + \sum_j c_j^*\right)$.

Proof. For a fixed l, every client j is in C_1 as in Lemma 2.5.5 to an extent of 1; either once in a type 1, 3, or 4 swap or exactly t^2 times among the type-2 swaps, each of which is counted with weight $\frac{1}{t^2}$. Similarly, every client j is in C_2 to an extent of at most 1 and is in C_3 to an extent of at most 1 over all swaps for this fixed l. Finally, we note each facility i on a cycle in H^* lies in the set B for at most one offset $1 \leq l' \leq k$ for that cycle, so its contribution $2t(f_i^* + f_i)$ to the bound is only counted with weight $\frac{1}{t^2}$ for this fixed l.

Lastly, every facility i not on a cycle in H^* lies in B for at most one index $l, 1 \leq l \leq t^2$ and, then, in only one swap for that particular l. Since we average the bound over all indices l between 1 and t^2, the contribution $2t(f_i^* + f_i)$ of such i is counted with weight only $\frac{1}{t^2}$. □

Next, we bound the expected facility movement cost change. Let F' be the set of facilities i that do not lie on a cycle in H^* consisting solely of facilities i' with $s_{i'} \in S_2$.

Lemma 2.5.7. *The expected change in movement cost (under the weighting described above) is at most* $\sum_{i \in F'} (f_i^* - f_i) + O\left(\frac{1}{t}\right)\left(\sum_i f_i^* + \sum_j (c_j^* + c_j)\right)$.

Proof. We consider two cases for a facility i. First, suppose $s_i \in S_0 \cup S_1$. Then when s_i is swapped out in a subtree during the shift from s_i to $o_{\mathsf{next}(i)}$, i is moved to either o_i, if $i = \mathsf{next}(i)$, or to $\sigma(o_i)$, if $i \neq \mathsf{next}(i)$. The latter case implies that $\sigma(o_i) \in S_2$. The total movement change is at most $f_i^* - f_i$ if i is moved to o_i and is at most $f_i^* - f_i + c(o_i, \sigma(o_i))$ if i is moved to $\sigma(o_i)$. Since $\sigma(o_i) \in S_2$ and $o_i = \mathsf{center}(\sigma(o_i))$, by Claim 2.5.2 we have that $c(o_i, \sigma(o_i)) \leq \frac{1}{t} \sum_{j \in D^*(\mathsf{cap}(\sigma(o_i)))} (c_j^* + c_j)$.

The only other time i is moved is when $\mathsf{end}(i)$ is randomly chosen from $\mathsf{cand}(i')$ for some facility i'. But this happens with probability at most $\frac{1}{t-1}$. In this case, i is shifted from s_i to $\sigma(o_i)$. We do not necessarily have $\sigma(o_i) \in S_2$ in this case, but we can use the bound $c(o_i, \sigma(o_i)) \leq f_i^* + f_i$ to bound the expected movement-cost change for i in this case to be at most $\frac{2f_i^*}{t-1}$. Overall, the expected movement-cost change for i is at most

$$\left(1 + \frac{2}{t-1}\right) f_i^* - f_i + \frac{1}{t-1} \sum_{j \in D^*\left(\mathsf{cap}(\sigma(o_i))\right)} (c_j^* + c_j).$$

Next, we consider the case $s_i \in S_2$. Let $\mathsf{center}(s_i) = o_{i'}$. When the swap consisting of i and all subtrees rooted at $\mathsf{cap}(s_i) \setminus \{o_{i'}\}$ is performed, i is moved from s_i to $o_{i'}$ unless i lies on a shift path during that swap, in which case it is moved like in the shift. Since $s_i \in S_2$, we have $c(s_i, o_{i'}) \leq \frac{1}{t} \sum_{j \in D^*(\mathsf{cap}(s_i))} (c_j^* + c_j)$. Unless i lies on a cycle with no S_1-locations, that is, $i \notin F'$, i lies between i'' and $\mathsf{next}(i'')$ for exactly one i'', and $\mathsf{shift}(s_{i''}, o_{\mathsf{next}(i'')})$ is performed to an extent of 1; this holds even if s_i lies on a shift path during the corresponding type-3 swap involving i. All other times i when is moved, it is due to the same reasons as in the previous case, so the total change in movement cost for facility i is at most

$$\left(1 + \frac{2}{t-1}\right) f_i^* - f_i + \frac{1}{t} \sum_{j \in D^*(\mathsf{cap}(s_i))} (c_j^* + c_j) + \frac{1}{t} \sum_{j \in D^*\left(\mathsf{cap}(\sigma(o_i))\right)} (c_j^* + c_j).$$

Adding up the appropriate expression for each facility accounts for the expected change in total movement cost. □

Proof of Theorem 2.5.1. By local optimality, the change in total cost for every test swap (counting every random choice) is nonnegative. By averaging over the various swaps, the expected change in total cost is nonnegative, so the sum of the expressions in Lemmas 2.5.6 and 2.5.7 is nonnegative. To generate an inequality involving a $-f_i$ term for facilities $i \notin F'$, we sum the bound given by Lemma 2.4.2 here over all cycles of H^* involving only facilities i with $s_i \in S_2$. This yields $0 \leq \sum_{i \notin F'} \left(-f_i + f_i^* + c(o_i, \sigma(o_i))\right)$, and we can bound $c(o_i, \sigma(o_i))$ by $\frac{1}{t} \sum_{j \in D^*(\mathsf{cap}(\sigma(o_i)))} (c_j^* + c_j)$. Adding this to the inequality that the expected change in total cost is nonnegative gives $\left(1 - O\left(\frac{1}{t}\right)\right)(C + F) \leq \left(3 + O\left(\frac{1}{t}\right)\right) C^* + \left(1 + O\left(\frac{1}{t}\right)\right) F^*$. □

2.6 Extension to the weighted case

The analysis in Section 2.5 (as also the proof of the 5 approximation) extends easily to the weighted generalization, wherein each facility i has a weight $w_i \geq 0$ and the cost of moving i to

s is given by $w_i c(i,s)$, to yield the same $\big(3+o(1)\big)$-approximation guarantee. With the exception of one small difference in the analysis, this requires only minor changes in the arguments. We discuss these briefly in this section.

Unless otherwise stated, the same notation from Section 2.5 is used in this section. We emphasize that f_i^* and f_i now refer to the weighted movement cost of facility i in the global or local optimum, respectively. So, $f_i^* = w_i \cdot c(i, o_i)$ and $f_i = w_i \cdot c(i, s_i)$.

One difference in notation is that the definition of S_1 is slightly revised to this weighted setting: $s_i \in S_1$ if $|\mathsf{cap}(s_i)| > t$, or $0 < |\mathsf{cap}(s_i)| \leq t$ and $|D^*(\mathsf{cap}(s_i))| \leq \max\{w_i, w_{i'}\} \cdot t$, where i' is such that $o_{i'} = \mathsf{center}(s_i)$ (equivalently, (i', i) is a center edge in H). If all facility weights are 1, then this definition of S_1 agrees with the definition in Section 2.5. Similarly, we say $s_i \in S_2$ if $|\mathsf{cap}(s_i)| \leq t$ and $|D^*(\mathsf{cap}(s_i))| > \max\{w_i, w_{i'}\} \cdot t$. Under these definitions, similar to Claim 2.5.2, we now have that $w_i \cdot c(s_i, o_{i'}) \leq \frac{1}{t} \sum_{j \in D^*(\mathsf{cap}(s_i))} (c_j + c_j^*)$ (since $c(o_i, \sigma(o_i)) \leq c_j^* + c_j$ for any $j \in D^*(\mathsf{cap}(s_i))$ as before, and $|D^*(\mathsf{cap}(s_i))| > w_i t$).

We consider the same set of test swaps and the same averaging of the inequalities generated by these swaps. When a test swap is performed, we consider the same shift and reassignment of facilities to generate the inequalities. In most cases, we also move the clients in the same way as before with the exception that if a client j has all of $\sigma(j), \sigma^*(j)$ and $\sigma(\sigma^*(j))$ being closed, then we do not necessarily send it to s_i where i is such that $o_i = \sigma^*(j)$. This is discussed in Lemma 2.6.1.

As in the discussion before Lemma 2.5.5, we consider a swap involving subtrees $\{T_r\}_{r=1}^k$. Let B be as before, and let B' be the set of facilities i such that i is a leaf in some T_r or, if the swap is a type-2 swap, that i is the first facility in $I_{l'}$. Note that $i \in B$ if and only if the unique (i, i') arc in H^* is a center arc with $i' \in B'$. We let C_1, C_2, and C_3 also be defined as in Section 2.5.

Lemma 2.6.1. *The expected change in client assignment cost for a test swap is at most* $\sum_{j \in C_1} (c_j^* - c_j) + \sum_{j \in C_2} 2c_j^* + \frac{1}{t-1} \sum_{j \in C_3} 2c_j^* + 4t \sum_{i \in B \cup B'} (f_i^* + f_i)$.

Proof. Consider one particular swap. As in the proof of Lemma 2.5.5, we move $j \in C_1$ to $\sigma^*(j)$ and $j \in C_2 \cup C_3$ to $\sigma(\sigma^*(j))$ and bound their assignment cost change in the same way. As before, it may be that for some of these clients $j \in C_2 \cup C_3$ we have that $\sigma(\sigma^*(j))$ was closed in the swap. For such clients, we do the following slight variant of the reassignment that was done in the proof of Lemma 2.5.5.

Suppose (i, i') is the center edge such that $\sigma^*(j) = o_i$ for a client $j \in C_2 \cup C_3$, and $s_{i'}$ is not open. If $w_i \geq w_{i'}$, then we send j to either s_i or o_i. As in the proof of Lemma 2.5.5, one of these must be open and the total cost of moving j from $s_{i'}$ to either s_i or o_i is at most $2c(i, s_i) + 2c(i, o_i)$.

Otherwise, if $w_{i'} > w_i$ then we send j to either $o_{i'}$ or $\sigma(o_{i'})$ (one of them must be open). The distance from $s_{i'}$ to either $o_{i'}$ or $\sigma(o_{i'})$ is bounded by $2c(i', s_{i'}) + 2c(i', o_{i'})$.

We conclude by noting that each facility $\hat{i} \in B$ has at most $w_{\hat{i}} \cdot t$ clients sent to either $s_{\hat{i}}$ or $o_{\hat{i}}$ from $\sigma(o_{\hat{i}})$ in the manner just described, because $\sigma(o_{\hat{i}})$ must be in S_3. Similarly, each $\hat{i} \in B'$ has at most $w_{\hat{i}} \cdot t$ clients sent to either $o_{\hat{i}}$ or $\sigma(o_{\hat{i}})$ from $s_{\hat{i}}$ in the manner described above, since $s_{\hat{i}} \in S_3$. So, the total client movement charged to $i \in B \cup B'$ this way is at most $4tw_i(c(i, s_i) + c(i, o_i)) = 4tf_i^* + 4tf_i$. □

Using the same weighting of the swaps as in Section 2.5 we get the following bound on the contribution of the client movement cost changes over these swaps. The proof is nearly identical, except we notice that a facility i' lies in the B'-set for various swaps to an extent of at most $\frac{1}{t^2}$ (under this weighting), since the facility i such that (i, i') is a center edge lies in some B-set to an extent of at most $\frac{1}{t^2}$.

Lemma 2.6.2. *The expected total client assignment cost change, weighted in the described manner, is at most* $\sum_j 3c_j^* - c_j + O\left(\frac{1}{t}\right)\left(\sum_i (f_i^* + f_i) + \sum_j c_j^*\right)$.

The contribution of the facility movement costs is bounded in essentially the same way as in Lemma 2.5.7. We just provide the details on how to account for the weights of the facilities. As before, let F' be the set of facilities i that do not lie on a cycle in H^* consisting solely of facilities i' with $s_{i'} \in S_2$.

Lemma 2.6.3. *The expected change in movement cost is at most*

$$\sum_{i \in F'} (f_i^* - f_i) + O\left(\frac{1}{t}\right)\left(\sum_i f_i^* + \sum_j (c_j^* + c_j)\right).$$

Proof. When $\mathsf{shift}(s, o)$ is performed, we move facilities i from s_i to $\mathsf{center}(o_i)$. If this shift was performed during a path swap, then the movement-cost change for a facility i moved in this shift is at most $w_i c(i, o_i) + w_i c(o_i, \sigma(o_i)) - w_i c(i, s_i) \leq 2w_i c(i, o_i) = 2f_i^*$ so the same bound used before applies.

If such a shift was performed along a path in a subtree, then we did not want to bound $c(o_i, \sigma(o_i))$ by $c(i, s_i) + c(i, o_i)$ because we do not want to cancel the contribution of $-c(i, s_i)$ to the bound. However, this only happened when $\sigma(o_i) \in S_2$ so we can use the fact that $|D^*(\mathsf{cap}(\sigma(o_i)))|$ is large and that $c(o_i, \sigma(o_i)) \leq c_j^* + c_j$ for any $j \in D^*(\mathsf{cap}(\sigma(o_i)))$. In our setting, as noted earlier, the movement cost $w_i \cdot c(o_i, \sigma(o_i))$ can be bounded by $\frac{1}{t}\sum_{j \in D^*(\mathsf{cap}(\sigma(o_i)))}(c_j^* + c_j)$, which is the same upper bound we used in the unweighted case.

Finally, the only other time we moved a facility was from some $s_i \in S_2$ to $\mathsf{center}(s_i)$. The cost of this move is now $w_i \cdot c(s_i, \mathsf{center}(s_i))$ which can also be bounded by $\frac{1}{t}\sum_{j \in D^*(\mathsf{cap}(s_i))}(c^*_j + c_j)$ using the same argument as in the previous paragraph. So, all bounds for the unweighted facility movement cost increase averaged over the swaps also hold in the weighted case. □

Finally, we remark that the same bound for the facility movement cost for facilities on a cycle with only S_2 facilities holds for the weighted case, again using arguments like in the proof of Lemma 2.6.3 to bound $w_i \cdot c(o_i, \sigma(o_i))$. Thus, the proof of Theorem 2.5.1 is adapted to prove the following result for the weighted case.

Theorem 2.6.4. *The cost of a locally-optimal solution using p swaps is at most $3+O\left(\sqrt{\frac{\log\log p}{\log p}}\right)$ times the optimum solution cost in weighted instances of mobile facility location.*

2.7 Reduction from k-median to MFL

In this section, we present approximation preserving reduction from the k-median problem to the MFL problem (this was originally shown by Friggstad and Salavatipour [34]).

Theorem 2.7.1. *If there exists an α-approximation for* MFL *problem then there exists an $(\alpha + o(1))$-approximation for the k-median problem.*

Proof. Consider an instance $\mathcal{I}_{k_{med}}$ of k-median problem in which we are given a distance metric $(V, \{c(i,j)\})_{i,j \in V}$, and a set $\mathcal{D} \subseteq V$ of clients. Without loss of generality, we assume that the minimum non-zero distance in c is at least one (this can be achieved by scaling). Let Δ denote the maximum distance between two vertices in V and let $\eta = \alpha n k \Delta$ where $n := |\mathcal{D}|$. We construct an MFL instance $\mathcal{I}$ as follows: the distance metrics is the same as the distance metric in $\mathcal{I}_{k_{med}}$, the client set is η copies of $\mathcal{D}$ where for each client in $cI_{k_{med}}$, we include η copies of it, or equivalently, we can assume each client j has demand $d_j = \eta$, and set $\mathcal{F} \subseteq V$ of arbitrary k initial facility locations. Let F^* and C^* denote respectively the movement and assignment cost of an optimal solution of $\mathcal{I}$, and let O denote an optimal solution of $\mathcal{I}_{k_{med}}$ with cost $opt_{k_{med}}$.

We claim $C^* + F^* \leq \eta \cdot opt_{k_{med}} \cdot (1 + \frac{1}{\alpha n})$. To see this, consider a solution that moves facilities in $\mathcal{F}$ to the locations in O and assigns clients as in O. The connection cost of clients in this solution is $\eta opt_{k_{med}}$ and the facility movement cost is at most $k\Delta = \frac{\eta}{\alpha n}$. Thus, we have

$$C^* + F^* \leq \eta opt_{k_{med}} + \frac{\eta}{\alpha n} \leq \eta \cdot opt_{k_{med}} \cdot (1 + \frac{1}{\alpha n})$$

since $opt_{k_{med}} \geq 1$.

Now suppose there exists an α-approximation algorithm $\mathcal{A}$ for MFL problem. Using $\mathcal{A}$ on $\mathcal{I}$, we can find a solution with cost at most $\alpha(\eta \cdot opt_{k_med} \cdot (1 + \frac{1}{\alpha n}))$. This solution translates to a solution of $\mathcal{I}_{k_{med}}$ with cost at most $\alpha \cdot opt_{k_{med}} \cdot (1 + \frac{1}{\alpha n}) = (\alpha + \frac{1}{n}) \cdot opt_{k_{med}}$ which proves the statement in the theorem. □

2.8 Bad locality gap with arbitrary facility-movement costs

In this section, we present an example that shows that if the facility-movement costs and the client-assignment costs come from different (unrelated) metrics then the p-swap local-search algorithm has an unbounded locality gap; that is, the cost of a local optimum may be arbitrarily large compared to optimal cost.

We first show a simple example for a single swap case, which we will later generalize for p swaps. Suppose we have two clients j_0, j_1 and two facilities i_0, i_1. Some distances between these clients and facilities are shown in the Figure 3.9(a); all other distances are obtained by taking the metric completion. Note that in this example, in order to have a bounded movement cost for facilities, the only option is to have one of i_0, j_1 as a final location of facility i_0 and one of i_1, j_0 as a final location of facility i_1.

As can be seen from the figure, the solution $O = \{i_0, i_1\}$ has total cost 2 (the movement cost is 0 and the client-assignment cost is 2). Now consider the solution $S = \{j_0, j_1\}$ which has a total cost of $2D$ (the movement cost is $2D$ and the client-assignment cost is 0). This is a local optimum since if we swap out j_0, then we have to swap in i_1 to have a bounded movement cost, which leads j_0 having assignment cost of ∞. By symmetry, there is no improving move for solution S, and the locality gap is D.

Now consider the example shown in Figure 3.9(b) for local-search with p simultaneous swaps. Suppose we have facility set $\{i_0, i_1, \ldots, i_p\}$ and client set $\{j_0, j_1, \ldots, j_p\}$. The global optimum $O = \{i_0, i_1, \cdots, i_p\}$ has total cost $p+1$ (facility movement cost is 0 and client-assignment cost is $(p+1) \cdot 1$) while $S = \{j_0, j_1, \ldots, j_p\}$ is a local optimum whose total cost is $(p+1) \cdot D$ (facility movement cost is $(p+1) \cdot D$ and client-assignment cost is 0). Consider any move $swap(X, Y)$. Note that $j_k \in X$ if and only if $i_{k-1} \in Y$ (where indices are $\mod(p+1)$) to ensure bounded movement cost. Let k be such that $j_k \in X$ and $j_{k+1} \notin X$. Then, j_k has an assignment cost of ∞ in the solution $(S \setminus X) \cup Y$. Hence, S is a local optimum.

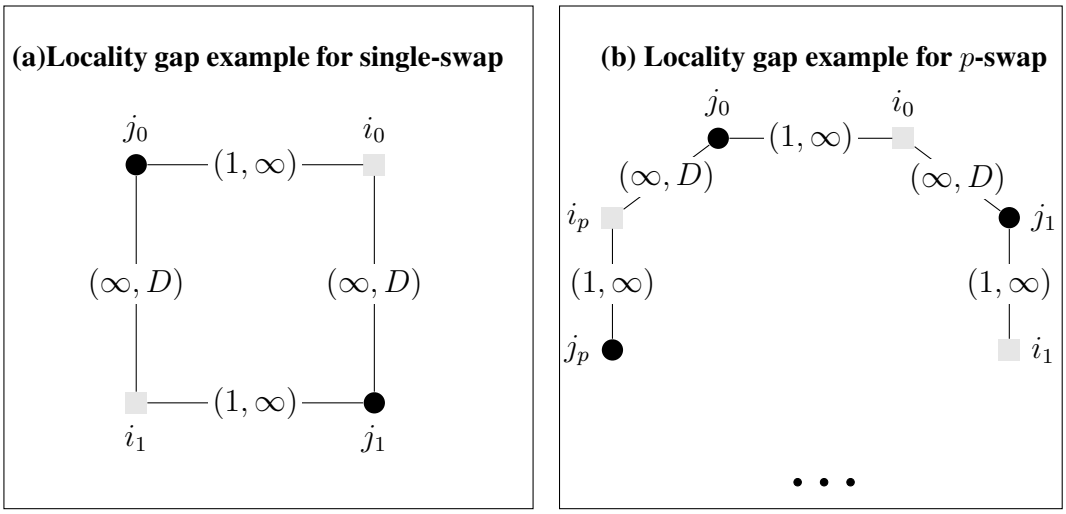


Figure 2.9: Examples showing large locality gap for the cases where local search allows (a) single swaps (b) at most p simultaneous swaps. The label (c, f) of an edge gives client-assignment cost c and the movement cost f of a facility along that edge.

Chapter 3

Minimum-Load k Facility Location

3.1 Introduction

In this chapter, we study facility-location problems from a less-considered point-of-view on the associated cost of a solution. We consider settings where the cost of serving a client by a facility is incurred by the facility. Each facility, such as a distribution center or a warehouse, may be responsible for supplying its clients and consequently bears a cost equal to the total cost of servicing its clients. In such settings, it is natural to consider the problem of minimizing the maximum cost borne by any facility.

Formally, we consider the following mathematical model: recall that we have a metric space $(\mathcal{F} \cup \mathcal{D}, \{c(i,j)\}_{i,j \in \mathcal{F} \cup \mathcal{D}})$, where $\mathcal{F}$ and $\mathcal{D}$ denote the facility and client sets, respectively. There are no facility-opening costs. The goal is to open k facilities from $\mathcal{F}$ and assign each client j to an open facility $\sigma(j)$ so as to minimize the maximum *load* of an open facility, where the load of an open facility i is defined to be $\sum_{j \in \mathcal{D}: \sigma(j)=i} c(i,j)$; that is, the load of i is the total connection cost incurred in serving the clients assigned to it. We call this the *minimum-load k-facility location* (MLkFL) problem. As is common in the study of facility-location problems, we assume that the clients and facilities lie in a common metric space, so the $c(i,j)$s form a metric.

To give an example that motivates the minimize the maximum load objective, consider a hospital with a nursing station where all the nurses start their morning round at the same time. Each nurse has to go back to the nursing station after visiting a patient to drop off patient's report sheet and pick up the next patient's report sheet. We want to assign patients to nurses such that all nurses finish their rounds at about the same time (or roughly equivalently, we want to minimize the latest time a patient is visited by a nurse). This problem and other similar settings motive a facility location problem in which the goal is to minimize the maximum load.

Despite the extensive amount of literature on facility-location problems, there is surprisingly a little amount of work on MLkFL and it remains a rather poorly understood problem (see [72]). One can infer that the problem is *NP*-hard, even when the set of open facilities is fixed, via a reduction from the makespan-minimization problem on parallel machines, and that an $O(k)$-approximation can be obtained by running any of the various $O(1)$-approximation algorithms for k-*median* [19, 54, 52, 11, 64] (where one seeks to minimize the *sum* of the facility loads). No better approximation algorithms are known for MLkFL even on line metrics, and the MLkFL problem was mentioned as an open problem in [72].

3.1.1 Summary of results

Our main result is for MLkFL problem in the setting where facilities and clients are located on a line and we completely resolve the status of MLkFL problem in this case: (1) we devise a polynomial-time approximation scheme (PTAS) for this case (Theorem 3.3.1), and (2) complementing our PTAS, we show (Theorem 3.5.1) that MLkFL is *strongly* NP-*hard* on line metrics (and hence, a PTAS is the best approximation that one can hope to achieve in polytime unless *P* = *NP*). This result is the *first* approximation algorithm for MLkFL on line metrics that achieves anything better than an $O(k)$-approximation. Our PTAS for line metrics consists of two main ingredients. First, we prove that there always exists a near-optimal solution possessing some nice structural properties (Section 3.3.1). Second, we show in Section 3.3.2 that these structural properties enable one to find such a structured solution via dynamic programming (DP).

We also consider MLkFL in a special case of tree metrics where the given tree defining the metric is a star. We consider MLkFL in star-metric for a more general setting where clients may have non-uniform integer demands $\{d_j\}_{j\in\mathcal{D}}$ and the demand of a client may be split *integrally* between several open facilities. We now define the load of a facility i to be $\sum_j x_{ij}c(i,j)$, where $x_{ij} \in \mathbb{Z}_{\geq 0}$ is the amount of j's demand that is assigned to i. We devise a 14-approximation algorithm for MLkFL on star metrics with non-uniform demands (Theorem 3.4.1). Notice that when we restrict the metric to be a star metric, we cannot create colocated copies of a client (without destroying the star topology), which makes the setting with non-uniform demands strictly more general than the unit-demand setting.

We show that MLkFL problem is resilient to attack by a variety of techniques that have been successfully applied to facility location problems, e.g., LP-based and local search-based techniques. In Section 3.6, we obtain integrality-gap for configuration-style LP relaxation for MLkFL. In this kind of LP formulation, we "guess" the optimum value B and have a variable $x_{i,C}$ for every facility i and every possible set C of clients such that $\sum_{j\in C} c(i,j) \leq B$. We show the integrality gap of $\Omega(k/\log k)$ even for line metrics (Theorem 3.6.1). Note that the configuration

LP is stronger than the natural LP-relaxation for MLkFL. Moreover, this holds even if the graph consisting of the edges (j, i) such that $c(i, j) \leq B$—call these feasible edges—is connected. This is in contrast with capacitated k-center [26, 8], where a large integrality gap for the natural LP arises due to the fact that the graph of feasible edges is disconnected. In Section 3.7, we show that a multi-swap k-median style local search has an unbounded locality gap.

The results presented in this chapter are part of a joint work between a group at the University of Alberta and the University of Waterloo which was published in 2014 [4]. The published results also include a quasi polynomial algorithm for the tree-metrics case.

3.1.2 Related work

The only prior result for MLkFL are due to Even et al. [32] and Arkin et al. [10] who studied the problem under the name *min-max star cover*. They view the problem as one where we seek to cover the nodes of a graph by stars (hence the name min-max star cover). In this setting, $\mathcal{F} = \mathcal{D}$. Even et al. [32] considered two versions of the problems called rooted and unrooted. In the rooted version, the set of open centers is prespecified and the goal is to find a partition of the nodes of the graph and assign each partition to one the roots so that the maximum load of a root is minimized. They notice that the rooted version is a special case of *minimum makespan scheduling on unrelated machines*, and thus there exists a 2-approximation due to Lenstra et al. [62]. For the rooted version, Even et al. [32] devise a $(4 + \epsilon, 4)$-bicriteria approximation algorithm which produces a solution with the cost within $(4 + \epsilon)$-factor of the optimal cost, and this solution opens at most $4k$ centers (so it violated the bound on the maximum number of open centers). Subsequently, Arkin et al. [10] improved the results by devising a $(3 + \epsilon, 3)$-bicriteria approximation algorithm.

Another close perspective is the objective in which we want to cover the nodes of an underlying graph with using a certain combinatorial object. Even et al. [32] and Arkin et al. [10] both devised 4-approximation algorithm when the covering objects are trees (see also [70]). This approximation was later improved to 3 by Khani and Salavatipour [56]. All these work also consider another variant in which the maximum cost of a covering object is fixed and one seeks to minimize the number of covering objects used. For this variation in the setting that the objects are paths or walks, Arkin et al. [10] give $(2\alpha_{k-med} + 1)$-approximation where α_{k-med} is the approximation ratio of k-median problem. Frederickson et al. [33] obtain an $(e + 1 - \frac{1}{k})$-approximation when the covering objects are tours rooted at a given node.

As with the k-median and k-center problems, MLkFL can also be motivated and viewed as a clustering problem: we seek to cluster points in a metric space around k centers in a way that minimizes the maximum load (or "star cost") of a cluster. Whereas MLkFL and k-center are

min-max clustering problems, where the quality is measured by the *maximum* cost (under some metric) of a cluster, k-median is a min-sum clustering problem, where the clustering quality is measured by summing the cost of each cluster.

3.2 Problem definition and preliminaries

In the minimum-load k-facility location (MLkFL) problem, we are given a set $\mathcal{D}$ of clients and a set $\mathcal{F}$ of facilities located in a common metric space $\{c(i,j)\}_{i,j\in\mathcal{F}\cup\mathcal{D}}$, and an integer $k \geq 1$. A feasible solution is a pair $(F \subseteq \mathcal{F}, \sigma : \mathcal{D} \to F)$ with $|F| = k$, where $\sigma(j) \in F$ indicates that client j is assigned to, or served by, facility $\sigma(j)$. The cost of such a solution is the maximum load of a facility in F where the load of facility i is defined to be sum of connection costs of clients assigned to it, i.e., $cost(F,\sigma) := \max_{i\in F} \sum_{j\in\sigma^{-1}(i)} c(i,j)$. The goal is to find a feasible solution (F,σ) with minimum cost. Throughout, we use OPT to denote an optimum solution and L^{opt} to denote its cost.

In some contexts, MLkFL is called star-cover problem. Consider a solution (F,σ). The function σ in fact gives a partition of client set $\mathcal{D}$, and so the solution can be viewed as a collection of stars where for each star $\mathcal{S}$, the root is a center i in F and leaves are clients in $\sigma^{-1}(i)$. Therefore, MLkFL can be seen as the problem of finding k stars, $\mathcal{S}_1, \mathcal{S}_2, \cdots, \mathcal{S}_k$, with roots in $\mathcal{F}$ and leaves in $\mathcal{D}$, that "cover" all clients so that the maximum load of a facility (or cost of a star) is minimized.

3.3 A PTAS for line metrics

In this section, we focus on MLkFL on line metrics and present a PTAS for it. Let set $V = \mathcal{D}\cup\mathcal{F}$ and let $n := |V|$. Here, each client/facility $p \in V$ is located at some rational point $v_p \in \mathbb{R}$. It may be that $v_p = v_{p'}$ for $p \neq p'$, for instance when we have collocated clients. To simplify notation, we use the term "point" to refer to a client or facility $p \in V$ as well as to its location v_p. The distance $c(p,p')$ between points $p, p' \in V$ is simply $|v_p - v_{p'}|$. We assume that $0 \leq v_1 \leq v_2 \leq \ldots \leq v_n$. The main result of this section is the following theorem.

Theorem 3.3.1. *There is a $(1+\varepsilon)$-approximation algorithm for MLkFL on line metrics for any constant $0 < \varepsilon \leq 1$.*

Our high-level approach is similar to other min-max problems. Namely, we present an algorithm that, given a guess B on the optimum solution value, either certify that $B < L^{opt}$ (recall

that L^{opt} is the cost of an optimal solution) or else find a solution with cost not much more than B. Our main technical result, which immediately yields Theorem 3.3.1 is the following.

Theorem 3.3.2. *Let $\Pi = (\mathcal{D} \cup \mathcal{F}, c, k)$ be a given* MLkFL *instance. For any constant $0 < \epsilon \leq 1$ and any $B \geq 0$, there is a polynomial-time algorithm $\mathcal{A}$ that either finds a feasible solution with the cost at most $(1 + 18\epsilon) \cdot B$ or declares that no feasible solution with the cost at most B exists. Thus, if $B \geq L^{opt}$, then $\mathcal{A}$ always finds a feasible solution with the cost at most $(1 + 18\epsilon) \cdot B$.*

Proof of Theorem 3.3.1. Let Δ be defined such that $v_p\Delta$ is an integer for each point p. So L^{opt} is an integer multiple of $\frac{1}{\Delta}$. Since $n \cdot v_n$ is clearly an upper bound on the cost of an optimal solution, L^{opt} is a multiple of $\frac{1}{\Delta}$ in the interval $[0, nv_n]$. We perform a binary search on this interval, and for each value $B' \in [0, nv_n]$ which is a multiple of $\frac{1}{\Delta}$, we try algorithm $\mathcal{A}$ from Theorem 3.3.2 with value $B = B'$. After $O(\log(nv_n\Delta))$ calls to binary search, we find two multiples of $\frac{1}{\Delta}$, B'_1 and B'_2 with $B'_1 \leq B'_2$ and $B'_2 - B'_1 = \frac{1}{\Delta}$ such that $\mathcal{A}$ certifies $B'_1 < L^{opt}$ and finds a solution with cost $(1 + 18\epsilon)B'_2$. Since L^{opt} is a multiple of $\frac{1}{\Delta}$, we can conclude that $B'_2 \leq L^{opt}$. Therefore, setting $\epsilon := \varepsilon/18$, by $O(\log(nv_n\Delta))$ calls to algorithm $\mathcal{A}$, we can find a solution with cost $\leq (1 + 18\epsilon) \cdot B'_2 \leq (1 + \varepsilon) \cdot L^{opt}$. Hence, we can find a solution with cost $(1 + \varepsilon)L^{opt}$ in polynomial time.

□

In what follows, we describe algorithm $\mathcal{A}$. We will assume that $B \geq L^{opt}$ and show how to find a solution with the cost at most $(1 + 18\epsilon) \cdot B$. Without loss of generality, we assume that $1/\epsilon$ is an integer. Let (F_B, σ_B) be a solution with the cost at most at most B. In the remainder of this section, we will describe some preprocessing steps that simplify the structure of the problem. In Section 3.3.1, we prove that a well-formed near-optimum solution exists, and in Section 3.3.2, we describe a dynamic programming algorithm that finds such a near-optimum well-formed solution.

Preprocessing

The first preprocessing step is to partition the instance into subinstances (and shift the origin suitably) so that for each point p in each subinstance, we have $0 \leq v_p \leq n \cdot B$. Note that solution (F_B, σ_B) has the maximum load at most B, so each facility serves clients that are at distance at most B from it, i.e., $c(j, \sigma_B(j)) \leq B$ for any client j. Therefore, if the distance of two consecutive points on the line is more than B, then we can decompose the instance into smaller instances such that the distance between any two consecutive points is at most B. For each of the resulting instances Π', we find the smallest k' such that running the subsequent algorithm on

the instance with k' instead of k finds a solution with the cost at most $(1+18\epsilon)B$. Since we are assuming $B \geq L^{opt}$, then the sum of these k' values over the subinstances is at most k. Note that in each subinstance Π' by shifting the origin suitably, we can assume $0 \leq v_i \leq n \cdot B$ for each point v_i.

The second preprocessing step is scaling of distances. We move every point $i \in V$ (recall $V = \mathcal{D} \cup \mathcal{F}$) left to its nearest integer multiple of $\frac{\epsilon B}{n}$ and then multiply this new point by $\frac{n}{\epsilon B}$. That is, move p from v_p to $\lfloor v_p \cdot n/\epsilon B \rfloor$. Denote the new position of client/facility p by v'_p. The following lemma describes how the optimum solutions to the original and new locations relate.

Lemma 3.3.3. *The optimum solution has the cost at most $(1+1/\epsilon) \cdot n$ for the instance given by the new positions v'. Furthermore, any solution with the cost at most $(1+\alpha\epsilon) \cdot (1+1/\epsilon) \cdot n$ for the new positions has the cost at most $(1+(2+2\alpha)\epsilon) \cdot B$ in the original instance.*

Proof. After sliding each point v_p left to its nearest integer multiple of $\frac{\epsilon B}{n}$, the distance between any two points changes by at most $\frac{\epsilon B}{n}$. So a load of each facility changes by at most ϵB, therefore each facility has load at most $(1+\epsilon)B$. Finally, after multiplying all points by $\frac{n}{\epsilon B}$, we have that the maximum load of any facility is at most $(1+1/\epsilon) \cdot n$.

Now consider any solution with the cost at most $(1+\alpha \cdot \epsilon) \cdot (1+1/\epsilon) \cdot n$. Scaling the points v' back by $\epsilon B/n$ produces a solution with the cost at most $(1+\alpha \cdot \epsilon)(1+\epsilon) \cdot \epsilon B \leq (1+(1+2\alpha)\epsilon) \cdot B$. Then sliding, any two points i, j back to their original positions v_i, v_j changes their distance by at most $\epsilon B/n$, so doing this for all points changes the load of any facility by at most ϵB. The resulting solution then has the cost at most $(1+(2+2\alpha)\epsilon) \cdot B$. □

In subsequent sections, we describe a $(1+8\epsilon)$-approximation for any one of the subinstances Π' of Π, except we use the new points v'_i. By Lemma 3.3.3, this gives us a solution to Π with the cost at most $(1+18\epsilon)B$, proving Theorem 3.3.2

To simplify notation, we use v_p to refer to the *new* location of point $p \in V$ (i.e. rename v'_p to v_p). Similarly, the notation $c(p, p')$ for $p, p' \in V$ refers to these new distances $|v'_p - v'_{p'}|$ and B denotes the new budget $(1+1/\epsilon) \cdot n$. From now on, we assume our given instance Π of MLkFL satisfies the following properties:

- There is a solution (F_B, σ_B) with the cost at most $B = (1+1/\epsilon) \cdot n$.
- Each point v_p is an integer between 0 and $\frac{n^2}{\epsilon}$.

3.3.1 Structure of near optimum solutions

In this section, we show that there is a near-optimum solution to the instance Π with clients and facilities $\mathcal{D} \cup \mathcal{F}$ that has some suitable structural properties. In Section 3.3.2, we will find such a solution using a dynamic programming approach.

We denote the open interval between two points v_i and v_j on the line by $I_{i,j}$ and call this the *arm* between i and j (assuming that one of i, j is a client and the other is a facility). An arm $I_{i,j}$ is *large* if $c(i, j) > \epsilon B$ and is *small* otherwise. We say that two arms $I_{i,j}$ and $I_{i',j'}$ *cross* if $I_{i,j}$ is not contained in $I_{i',j'}$ or vice versa, and $I_{i,j} \cap I_{i',j'} \neq \emptyset$. Note that the intervals are open so if two intervals share an endpoint that does not count as an intersection and therefore such intervals are non-crossing.

A *well-formed solution* for an $\mathsf{ML}k\mathsf{FL}$ instance is a solution in which the small arms between clients and their assigned facilities (centers) do not cross. To recall, (F_B, σ_B) is a solution with the maximum load B. We show an existence of a low-cost well-formed solution, with facility set F_B, in two steps:

Step 1: **Solution with fractional assignment and no crossing small arms.**
We demonstrate the existence of a *fractional solution* with facility set F_B where the clients are assigned to F_B centers fractionally. This will be such that the fractional load of each facility is at most B, all strictly fractional arms in the support have length at most $2\epsilon B$, and all small arms in the support of the solution do not cross.

Step 2: **Solution with integral assignment and no crossing small arms.**
We use a rounding algorithm for the Generalized Assignment Problem (GAP) by Shmoys and Tardos [75] to round a fractional solution from Step 1 to an integral solution with the cost at most $(1 + 2\epsilon)B$.

We emphasize that this rounding algorithm is not a part of our algorithm, it is only used to demonstrate the existence of a well-formed solution.

Step 1: solution with fractional assignment and no crossing small arms.

For the first step, we will consider a fractional uncrossing argument to eliminate crossings. Instead of proving the fractional uncrossing process eventually terminates, we will instead provide a potential function that strictly decreases in a fractional uncrossing. This potential function is the objective function of a mixed integer-linear program below; thus an optimal solution will not contain any crossings between small arms its support.

We consider the following mixed-integer programming (MIP) with variable x_{ij} which denotes if a client j is assigned to facility $i \in F_B$. The first constraint ensures every client is assigned to some facility. The second constraint ensures the load of each facility in F_B does not exceed B. The third constraint ensures that any arm with length at least $2\epsilon B$ is integral.

$$\min \sum_{i \in F_B} \sum_{j \in \mathcal{D}} c(i,j) \cdot x_{ij} \qquad \text{(MIP)}$$

$$\text{s.t.} \quad \sum_{i \in F_B} x_{ij} = 1 \qquad \forall j \in \mathcal{D}$$

$$\sum_{j \in \mathcal{D}} c(i,j) \cdot x_{ij} \leq B \qquad \forall i \in F_B$$

$$x_{ij} \in \{0,1\} \qquad \forall i,j : c(i,j) \geq 2\epsilon B$$

$$0 \leq x_{ij} \leq 1 \qquad \forall i,j : c(i,j) < 2\epsilon B.$$

We stress that this is *not* a relaxation for MLkFL. The objective function is more similar to the objective function for the k-median problem. Rather, we will only be using this to help demonstrate the existence of a well-formed solution. The objective function acts as a potential function.

Lemma 3.3.4. *The optimal solution x^* of mixed integer-linear program (MIP) does not have any crossing small arms in its support.*

Proof. First observe that there is in fact a feasible solution to (MIP) because the integer solution (F_B, σ_B) defines a feasible solution to (MIP). By the standard theory of mixed-integer programming and the fact that the set of feasible solutions is bounded, there exists an optimal solution x^*. The rest of this proof shows that an optimal solution to (MIP) cannot contain crossings between small arms in its support.

Suppose x^* is a feasible solution such that two small arms $I_{i,j}$ and $I_{i',j'}$ in the support of x^* cross. To simplify notation, let $s_1 = v_i, s_2 = v_{i'}$ be the locations of the centers i, i' and $v_1 = v_j$ and $v_2 = v_{j'}$ be the locations of the clients j, j'. Also let x_1 denote x^*_{ij} and x_2 denote $x^*_{i'j'}$. That is, x_1 is the extent to which the client at location v_1 is assigned to the center at location s_1 and similarly for x_2. Finally, let $\ell_1 = |s_1 - v_1|$ and $\ell_2 = |s_2 - v_2|$ denote the lengths of the two crossing small arms. See Figure 3.1 for an illustration of how this notation is used.

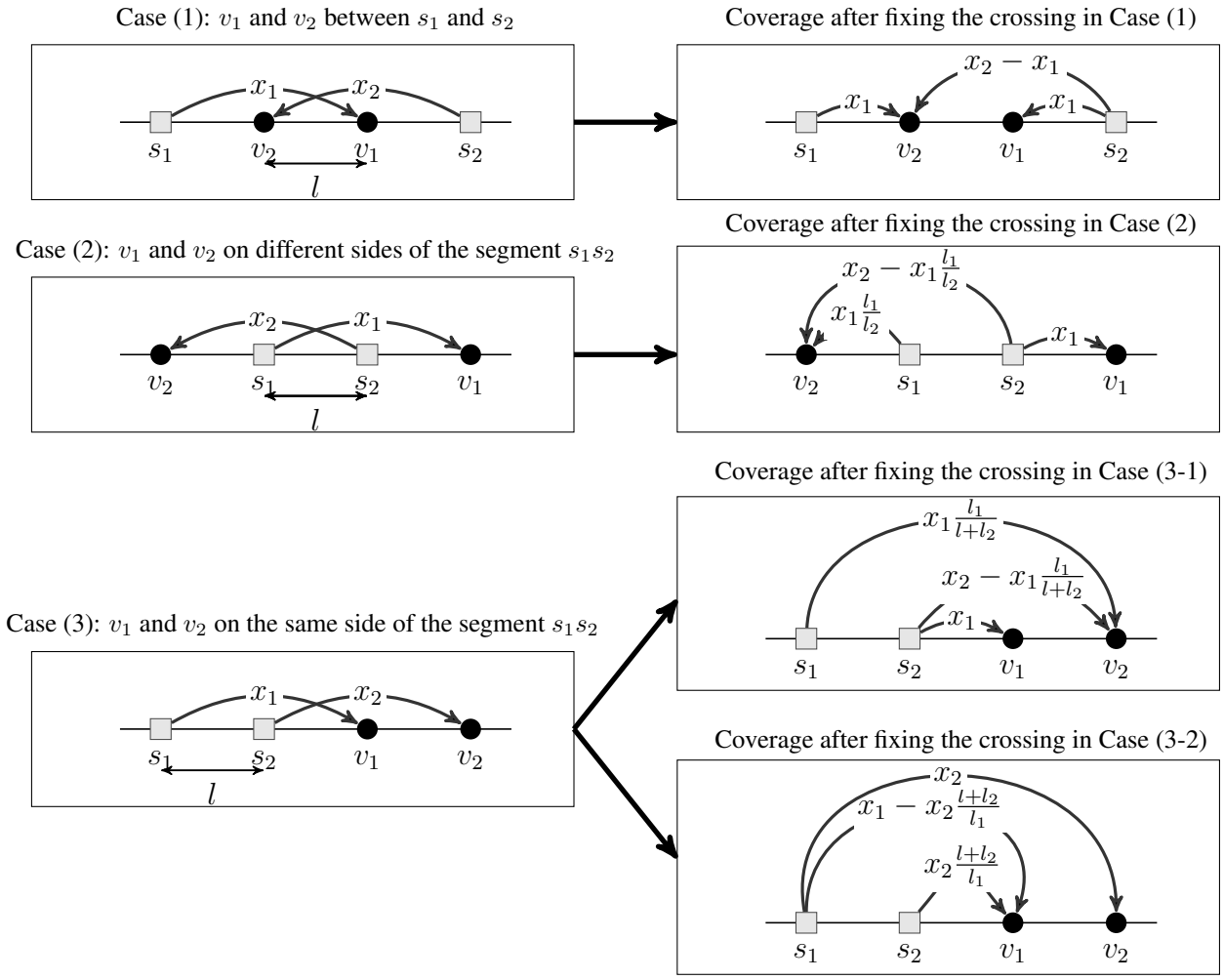


Figure 3.1: Fixing the crossing between two small arms.

We check all possible ways that these two arms can cross. When we say that we shift some value α of coverage from one variable y to another z, we mean increase z by α and decrease y by α. Note that we will always shift value between the $x^*_{ij}, x^*_{i'j}, x^*_{ij'}$ and $x^*_{i'j'}$ values. Since $\ell_1, \ell_2 \leq \epsilon B$ then $c(i, j')$ and $c(i', j) \leq 2\epsilon B$ so such an uncrossing will maintain the constraint that only arms of length at most $2\epsilon B$ may be fractional.

Case (1): v_1 and v_2 lie between s_1 and s_2 (see Figure 3.1). Let $\ell > 0$ be the length of intersecting parts of these arms. Without loss of generality, assume that $x_1 \leq x_2$. Shift x_1 coverage

from x^*_{ij} to $x^*_{ij'}$ and from $x^*_{i'j'}$ to $x^*_{i'j}$ and note that this preserves feasibility, since each client is still covered (fractionally) to the extent of 1. The total cost of the two facilities s_1 and s_2 decreases by $2\ell x_1 > 0$, so the objective function strictly decreases and we are left with an even cheaper solution to (MIP).

Case (2): v_1 and v_2 are on different sides of the segment s_1s_2 (see Figure 3.1). Let $\ell > 0$ be the distance between s_1 and s_2. Without loss of generality, assume that $x_1\ell_1 \leq x_2\ell_2$. Shift $x_1\frac{\ell_1}{\ell_2}$ from $x^*_{i'j'}$ to $x^*_{ij'}$ and shift x_1 from x^*_{ij} to $x^*_{i'j}$.

We verify that the fractional load at each center s_1 and s_2 does not increase. That is, the load at s_1 changes by $x_1\frac{\ell_1}{\ell_2}(\ell_2 - \ell) - x_1\ell_1 = -x_1\frac{\ell_1}{\ell_2}\ell < 0$ and the load at s_2 changes by $x_1(\ell_1 - \ell) - x_1\frac{\ell_1}{\ell_2}\ell_2 = -x_1\ell < 0$. Since the load at both facilities strictly decreases then this also yields a cheaper solution to (MIP).

Case (3): v_1 and v_2 are on the same side of the segment s_1s_2 (see Figure 3.1). Let $\ell > 0$ be the distance between s_1 and s_2. Without loss of generality, assume that v_1 and v_2 are on the right side of segment s_1 and s_2 and the left center is s_1. This means v_1 is between s_2 and v_2 and hence, $\ell_1 < \ell + \ell_2$. As a consequence: $(\ell + \ell_2)(\ell_1 - \ell) < \ell_1\ell_2$.

There are two sub-cases:

Case (3-1): $x_1\ell_1 \leq x_2(\ell + \ell_2)$.
This means $x_1\frac{\ell_1}{\ell+\ell_2} \leq x_2$. We shift x_1 from x^*_{ij} to $x^*_{i'j}$ and shift $x_1\frac{\ell_1}{\ell+\ell_2}$ from $x^*_{i'j'}$ to $x^*_{ij'}$. The fractional load at s_1 changes by $x_1\frac{\ell_1}{\ell+\ell_2}(\ell+\ell_2) - x_1\ell_1 = 0$ and the fractional load at s_2 changes by $x_1(\ell_1 - \ell) - x_1\frac{\ell_1}{\ell+\ell_2}\ell_2 = x_1(\ell_1 - \ell - \frac{\ell_1\ell_2}{\ell+\ell_2}) < 0$. Since the total load strictly decreases, then this also yields a cheaper solution to (MIP).

Case (3-2): $x_1\ell_1 > x_2(\ell + \ell_2)$.
We shift x_2 from $x^*_{i'j'}$ to $x^*_{ij'}$ and shift $\frac{x_2(\ell+\ell_2)}{\ell_1}$ from x^*_{ij} to $x^*_{i'j}$. The fractional load at s_1 changes by $x_2(\ell + \ell_2) - x_2\frac{\ell+\ell_2}{\ell_1}\ell_1 = 0$ and the fractional load at s_2 changes by $x_2\frac{(\ell+\ell_2)(\ell_1-\ell)}{\ell_1} - x_2\ell_2 < 0$. Since the total load strictly decreases, then this is also yields a cheaper solution to (MIP).

In all these cases, the new solution is feasible and has a smaller objective value as required, so an optimal solution x^* of (MIP) does not have any crossing small arms. □

Step 2: solution with integral assignment and no crossing small arms.

In the following, we will use Lemma 3.3.4 to prove the existence of a near-optimum solution to instance Π where the small arms used by clients do not cross. To complete this proof, we rely on

a structural result concerning the polytope of a relaxation for the following scheduling problem.

Definition 3.3.5. *In the* scheduling problem on unrelated machines*, we are given machines* $m_1, \ldots, m_k$*, jobs* $j_1, \ldots, j_n$*, and processing times* $p(m_i, j_a) \geq 0$ *for any job* j_a *and any machine* m_i*. The goal is to assign each job* j_a *to a machine* $\phi(j_a) \in \{m_1, \ldots, m_k\}$ *to minimize the maximum total running time* $\sum_{a:\phi(j_a)=m_i} p(m_i, j_a)$ *of any machine.*

Shmoys and Tardos [75] prove a result concerning the polytope of an LP relaxation for this problem, as a part of a more general result concerning the related *Generalized Assignment Problem* (GAP). The following summarizes the results they obtain that are relevant for our work.

Theorem 3.3.6 (Shmoys and Tardos, [75])**.** *Suppose we have a bound* B *and fractional values* $x(m_i, j_a) \geq 0$ *for each job* j_a *and each machine* m_i *that satisfy the following:*

- $\sum_{i=1}^{k} x(m_i, j_a) = 1$ *for each job* j_a*,*
- $\sum_{a=1}^{n} x(m_i, j_a) \leq B$ *for each machine* m_i*.*

Then there is an assignment ϕ *of jobs to machines such that* $x(\phi(j_a), j_a) > 0$ *for each job* j_a *and the maximum load of any machine under* ϕ *is at most* $B + \max_{a,i:0<x(m_i,j_a)<1} p(m_i, j_a)$*.*

We use the above theorem together with Lemma 3.3.4 to prove the following.

Theorem 3.3.7. *There exists a feasible (integer) solution to the* MLkFL *instance* Π *with facility set* F_B *and the maximum load* $(1 + 2\epsilon)B$ *for each facility such that no two small arms cross.*

Proof. Let x^* be the fractional solution provided by Lemma 3.3.4. We view x^* as a solution to the following scheduling problem on unrelated machines. We have k machines $m_1, \ldots, m_k$, each corresponding to a facility $i \in F_B$. For each client $a \in \mathcal{D}$, there is a single job j_a. The processing time $p(m_i, j_a)$ of job j_a on machine m_i is $|v_i - v_a|$, the distance between the corresponding locations.

Now, x^* fractionally assigns each job j_a to the machines to a total extent of 1 and the maximum (fractional) load at machine m_i is B. Furthermore, the only strictly fractional assignments (i.e. those with $0 < x^*_{ij} < 1$) have $|v_i - v_j| \leq 2\epsilon B$. In the scheduling terminology, the only strictly fractional assignments are between a job j_a and a machine m_i such that $p(m_i, j_a) \leq 2\epsilon B$.

Theorem 3.3.6 shows we can transform this fractional assignment x^* into an integer assignment such that (i) if client j is assigned to facility/center i, then $x^*_{ij} > 0$ and (ii) the maximum load of a facility is $B + \max_{i,j:0<x^*_{ij}<1} |v_i - v_j| \leq B + 2\epsilon B$. In this solution, small arms used by clients do not cross because they come from the support of x^*. □

3.3.2 Finding a well-formed solution

Theorem 3.3.7 shows that there is a solution of the cost at most $(1 + 2\epsilon)B$ such that no two small arms (i.e. length $\leq \epsilon B$) used to assign clients to centers cross. Let (F_B, σ'_B) denote such solution. In this section, we present a dynamic programming approach that finds such a well-formed solution of the cost at most $(1 + 8\epsilon)B$.

The main idea behind our approach is the following. If it were true that a near-optimum solution did not have any crossing arms (large or small) then we can find such a solution using a dynamic programming approach (see Figure 3.2). At a very high-level, we could exploit the laminar structure of the solution by decomposing the solution into a family of nested intervals $\mathcal{I}$ such that for every $I \in \mathcal{I}$ there is one center (supplier) s with $v_s \notin I$ such that clients in I are served either by centers in I or by s. From this, we can consider triples (I, s, β) where $I \in \mathcal{I}$, s is a location outside of I, and β is some integer between 0 and $\mathsf{poly}(n, 1/\epsilon)$ describing the load assigned to s from clients in I. We can look for partial solutions parameterized by these triples and relate them through an appropriate recurrence.

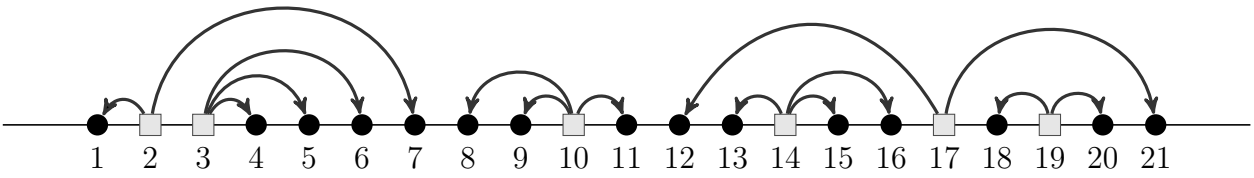


Figure 3.2: Example of a solution without crossing arms.

Unfortunately, we are only guaranteed that the small arms do not cross in our near-optimum solution so the collection of all arms in the solution is not necessarily laminar. To handle this general case, we must carry extra information about large arms through our dynamic programming approach and keep track of how many large arms of each possible length cross an interval. Since each large arm has a length between ϵB and B, if we store the exact length, then this amounts to storing a vector with roughly B coordinates. There are exponentially many vectors with B coordinates that the values in different coordinates sum up to at most n, so we cannot keep track of this information. However, by coarsening the length, we ensure that large arms have only a constant possible number of different perceived length, so we can keep track of this information when we move to perceived length.

First, recall that all large arms have length more than ϵB. Thus, each facility is serving at most $\frac{(1+2\epsilon)B}{\epsilon B} \leq \frac{3}{\epsilon}$ clients that are at distance more than ϵB from it; in other words each facility is

assigned at most $\frac{3}{\epsilon}$ large arms in the solution provided by Theorem 3.3.7. Since large arms have length at least ϵB, if we store their length in $\epsilon^2 B$, we will not lose much information about them. Let $\mathsf{MULT}(\mathsf{i},\mathsf{j})$ denote the number of multiples of $\epsilon^2 B$ in $(v_i, v_j]$ interval for $v_i \leq v_j$, so

$$\mathsf{MULT}(\mathsf{i},\mathsf{j}) = \left\lfloor \frac{v_j}{\epsilon^2 B} \right\rfloor - \left\lfloor \frac{v_i}{\epsilon^2 B} \right\rfloor.$$

Then, we measure the length of a large arm I_{ij} between client j and facility i as $\epsilon^2 B \cdot \mathsf{MULT}(\mathsf{i},\mathsf{j})$ if $v_i \leq v_j$ or $\epsilon^2 B \cdot \mathsf{MULT}(\mathsf{j},\mathsf{i})$ otherwise. We call this the perceived cost of this arm. In this method of measurement, the length of the arm changes by at most $\epsilon^2 B$, and so the total load for each center changes by at most $3\epsilon B$. Now for this method of measurement, since there are $\frac{1}{\epsilon^2}$ coordinates, the number of possible vectors for keeping track of large arms is at most $n^{\frac{1}{\epsilon^2}}$ which is polynomial.

In the dynamic programming algorithm described below, we will use this coarse method to measure the distance of large arms. Since in dynamic programming approach the problem is often broken into subproblem, a large arm might be partitioned with respect to subproblems. We would like to note that for points $v_{p_0} \leq v_{p_1} \leq \cdots \leq v_{p_t}$ with $p_0 = i$ and $p_t = j$, $\mathsf{MULT}(\mathsf{i},\mathsf{j}) = \sum_{q=0}^{t-1} \mathsf{MULT}(\mathsf{p_q},\mathsf{p_{q+1}})$ (Note that $\mathsf{MULT}(\mathsf{a},\mathsf{a}) = 0$ for any a.). We use the term the *perceived cost* of a facility i to denote the total cost of the small arms plus the perceived cost of its large arms. The following is proved using arguments similar to the proof of Lemma 3.3.3, recalling that every facility i n F_B has at most $3/\epsilon$ large arms.

Lemma 3.3.8. *The perceived cost of every facility in (F_B, σ'_B) is at most $(1+5\epsilon)B$. Furthermore, a solution with maximum perceived cost at most $(1+5\epsilon)B$ and at most $3/\epsilon$ large arms per center has the (actual) cost at most $(1+8\epsilon)B$.*

Our dynamic programming algorithm will find a solution with the perceived cost at most $(1+5\epsilon)B$ and at most $3/\epsilon$ large arms per center, so the actual cost will be at most $(1+8\epsilon)B$.

Dynamic programming

Before we formally define the subproblems of dynamic programming, we discuss the structure of a well-formed solution, say (F, σ). We call a client covered by a small (large) arm a *small client* (*large client*), respectively. It will be convenient to associate a direction with each arm, which goes from the center/facility to the client. For a facility i, let $\mathcal{S}_{small}$ denote the clients covered by small arms to i, i.e., small clients in $\sigma^{-1}(i)$. Let the *small-span* of i be the open interval, possibly empty, spanning $(l_{\mathcal{S}}, r_{\mathcal{S}})$ where

$$l_{\mathcal{S}} = \min\left\{v_i, \min_{j \in \mathcal{S}_{\text{small}}} v_j\right\} \quad \text{and} \quad r_{\mathcal{S}} = \max\left\{v_i, \max_{j \in \mathcal{S}_{small}} v_j\right\}$$

are the left most and the right most small clients (or facility) in this star, respectively. Since the small arms do not intersect in (F, σ), for any two small-spans I_1 and I_2 of two facilities, either $I_1 \cap I_2 = \emptyset$ or $I_1 \subseteq I_2$ or $I_2 \subseteq I_1$. Therefore, the $\subseteq$ relation between small-span of facilities in F defines a laminar family. A laminar family can naturally be viewed as a forest where we put a node for each member of the family and each node I has an edge to the minimal member say I' of the family that contains it, i.e., $I \subseteq I'$. If a facility i does not serve any client by a small arm then its small-span is (v_i, v_i) by definition and although this is an empty interval, in the forest corresponding to laminar family, it has an edge to the small-span (interval) that contains v_i. We frequently refer to this forest-view when referring to a laminar family.

Let us try to understand the subproblems that come up in constructing a solution. The dynamic programming in fact stitches together the solutions of these subproblems in order to find the solution to the original problem. Let $V_{i,j}$ denote the set of points $\{v_i, v_{i+1}, \cdots, v_j\}$ with $i \leq j$, so $V = V_{1,n}$. We want to use the dynamic programming to answer the question if it is possible to open k centers in $V_{1,n}$ and assign clients to these centers such that the perceived cost of each center is at most $(1 + 5\epsilon)B$ and the small-span of the centers form a laminar family.

Now consider solution (F_B, σ'_B) which has a maximum perceived load of $(1+5\epsilon)B$. Consider the forest corresponding to the small-span of centers in this solution and let s be the center that its small-span is the leftmost root in the forest. We can guess s as there are at most $O(n)$ possible choices for it. Let k_r and k_l denote the number of centers in F which are on the right and left side of s respectably, so $k_r + k_l = k - 1$. There are $O(k)$ possible choice for k_r and k_l, so we guess k_r and k_l as well (note that k is dominated by n). Let us now focus on the interval $V_{1,s-1}$. Since s corresponds to the interval at the root of the forest, no center in $V_{s+1,n}$ can serve clients in $V_{1,s-1}$ by small arms (otherwise the small-span of s is a subset of small-span of some other center). We can guess the load corresponding to small clients served by s in $V_{1,s-1}$ as this load is an integer in $\mathsf{poly}(n, 1/\epsilon)$. We may have some clients in $V_{1,s-1}$ served by large arms originating in $V_{s,n}$. We cannot guess the large arms serving these clients as the number of possible such arms is not polynomial (there are $O(n^k n^{3/e})$ possible choices) but instead we can bundle large arms entering $V_{1,s-1}$ based on their perceived length past v_{s-1}, that is their perceived length in $V_{1,s-1}$ interval. More precisely, we guess how many large arms have perceived length $q \times \epsilon^2 B$ past v_{s-1} for each $0 \leq q \leq \frac{1}{\epsilon^2}$ as there are $O(n^{\frac{1}{\epsilon^2}})$ possible choices. Similarly, we may have some large arms originating in the interval $V_{1,s-1}$ which serve clients in $V_{s+1,n}$. Again we can bundle these arms based on their perceived length past v_{s+1} (their perceived length in $V_{s+1,n}$) and we guess the number of large arms with perceived length $q \times \epsilon^2 B$ past v_{s+1} for each $0 \leq q \leq \frac{1}{\epsilon^2}$. This give us an idea of what parameters are needed for describing the subproblems.

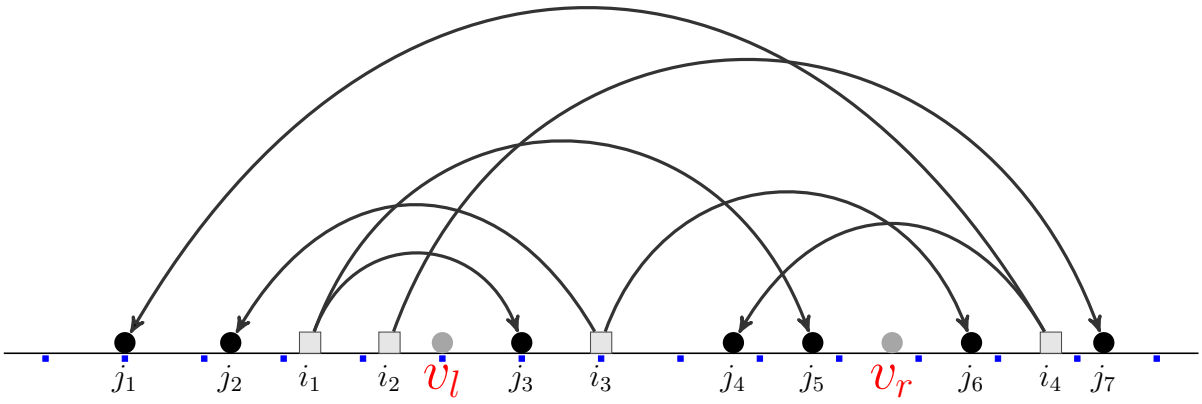
The discussion above gives us a sense of what kind of subproblem we should consider. We consider a subproblem which for an interval $V_{l,r}$ asks if it is possible to open at most k' centers in the interval, for $k' \leq k$, and we may have one extra center s with small arms serving clients in $V_{l,r}$ Now consider some interval $V_{l,r}$ between two arbitrary points v_l, v_r and consider how (F_B, σ'_B) interacts with this interval. There may be some large arms that enter and/or leave this interval from v_l or v_r. The arms that enter the interval can cover the *deficiency* of coverage for some client in the interval and the arms that leave the interval provide coverage for some client outside of interval and can be viewed as *surplus* to the demand of coverage of the clients in the interval. We keep track of all large arms crossing the sides of interval $V_{l,r}$ in terms of deficiency and surplus vectors as follows:

- **Deficiencies:** Vector $\mathbf{D}_l$ is the deficiency vector in $[n]^{\frac{1}{\epsilon^2}}$ for v_l where $\mathbf{D}_l[q]$ for $0 \leq q \leq \epsilon^{-2}$ is the number of large arms from a center location $i < l$ to a client $j \geq l$ such that $\mathsf{MULT}(\mathsf{l}, \mathsf{j}) = q$. So $\mathbf{D}_l[q]$ keeps track of the number of large arms originating in $i < l$ and crossing exactly q multiples of $\epsilon^2 B$ in interval $(v_l, v_j]$. Note that client $j \geq l$ can be located outside of interval $V_{l,r}$, i.e., $j > r$ as well. The vector $\mathbf{D}_r$ is defined similarly for v_r, that is, $\mathbf{D}_r[q]$ is the number of large arms from a center location $i > r$ to a client $j \leq r$ such that $\mathsf{MULT}(\mathsf{j}, \mathsf{r}) = q$. Let $q' = \mathsf{MULT}(\mathsf{l}, \mathsf{r})$, then all arms represented by $\mathbf{D}_l[q]$ for $q < q'$ must end at clients located in $V_{l,r}$ and all arms represented by $\mathbf{D}_l[q]$ for $q > q'$ must end at clients located to the right of $V_{l,r}$. If $q = q'$ then some arms may end at clients in $V_{l,r}$ and some may end at clients located to the right of $V_{l,r}$ (See Figure 3.3).

- **Surpluses:** Vector $\mathbf{S}_l$ is the surplus vector in $[n]^{\frac{1}{\epsilon^2}}$ for v_l where $\mathbf{S}_l[q]$ for $0 \leq q \leq \epsilon^{-2}$ is the number of large arms from a center location $i \geq l$ to a client $j < l$ such that $\mathsf{MULT}(\mathsf{j}, \mathsf{l}) = q$. So $\mathbf{S}_l[q]$ keeps track of large arms originating at $i \geq l$ and crossing exactly q multiples of $\epsilon^2 B$ in interval $(v_j, v_l]$. Note that center $i \geq l$ can be larger than r and does not need to be located in the interval $V_{l,r}$. The vector $\mathbf{S}_r$ is defined similarly, that is, $\mathbf{S}_r[q]$ is the number of large arms from a center location $i \leq r$ to a client $j > r$ such that $\mathsf{MULT}(\mathsf{r}, \mathsf{j}) = q$. Recall that $q' = \mathsf{MULT}(\mathsf{l}, \mathsf{r})$. Note that for $q > q'$, any arm contributing to $\mathbf{D}_l[q]$ also contributes to $\mathbf{S}_r[q - q']$ and similarly, any arm contributing to $\mathbf{D}_r[q]$ also contributes to $\mathbf{S}_l[q - q']$ (See Figure 3.3).

The dynamic programming table. The table we build in our dynamic programming algorithm captures "snapshots" of solutions bound between two given points plus some information on how arms cross these points. We consider the values $A(k', l, r, s, \beta, \mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r)$ corresponding to subproblems. The meanings of the parameters are as follows.

- $0 \leq k' \leq k$ is the number of centers in the interval $V_{l,r}$.



$\mathbf{D}_l[1] = 1$ (arm i_1j_3) $\mathbf{D}_l[4] = 1$ (arm i_1j_5) $\mathbf{D}_l[8] = 1$ (arm i_1j_7)	$\mathbf{D}_r[2] = 1$ (arm i_4j_4) $\mathbf{D}_r[9] = 1$ (arm i_4j_1)	$\mathbf{S}_l[3] = 1$ (arm i_3j_2) $\mathbf{S}_l[4] = 1$ (arm i_4j_1)	$\mathbf{S}_r[1] = 1$ (arm i_3j_6) $\mathbf{S}_r[3] = 1$ (arm i_2j_7)

Figure 3.3: Example of deficiency and surplus vectors. In this example, multiples of $\epsilon^2 B$ are shown by small blue squares, $q' := \mathsf{MULT}(\mathsf{l}, \mathsf{r}) = 5$, and all the arms are large arms. Note that the arm i_2j_7 contributes to $\mathbf{S}_r[3]$ and $\mathbf{D}_l[8]$ and the arm i_4j_1 contributes to $\mathbf{S}_l[4]$ and $\mathbf{D}_r[9]$.

- $1 \leq l \leq r \leq n$ corresponds to the interval $V_{l,r}$.
- $s \in \mathcal{F}$ denotes a single point with either $s < l$ or $s > r$ (i.e. outside of $V_{l,r}$) that is the center of some star, or else $s = \perp$. If $s \neq \perp$ it is the only center outside of $V_{l,r}$ with small arms going into $V_{l,r}$ and the total cost of small arms that s pays to cover vertices in $V_{l,r}$ is β where $0 \leq \beta \leq (1 + 5\epsilon)B$ is an integer.
- $\mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r$ are deficiency and surplus vectors for the endpoints of interval $V_{l,r}$.

Note that in the above, if $s = \perp$ then the value of β can be assumed to be zero.

The value $A[k', l, r, s, \beta, \mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r]$ is TRUE if and only if the following holds. It is possible to open k' centers in the interval $V_{l,r}$ and assign each client $j \in \mathcal{D}$ with $l \leq j \leq r$

- to one of the k' open centers,
- to center s, if $s \neq \perp$,
- or to a large arm entering $V_{l,r}$

and also assign the start of some of the large arms exiting the interval to these open centers in $V_{l,r}$ such that the following hold.

- The perceived load of each of the k' centers is at most $(1 + 5\epsilon)B$.
- The load of s from small arms going to clients $j \in \mathcal{D}$ with $l \leq j \leq r$ is β,
- The large arms entering and/or exiting $V_{l,r}$ are *consistent* with $\mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r$.

By consistent, we mean the following. For each interval $[a, b]$ where a and b are consecutive multiples of $\epsilon^2 B$, we check the number of promised clients to be served by $\mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r$ in this interval (depending on position of a and b with respect to l and r, some of these vectors may not give any information). More precisely, when $r \leq b$, each of $\mathbf{D}_l[q_1]$ and $\mathbf{S}_r[q_2]$ where $q_1 = \mathsf{MULT}(\mathsf{l}, \mathsf{a})$ and $q_2 = \mathsf{MULT}(\mathsf{r}, \mathsf{a})$ gives the number of clients in $[a, b)$ that are supposed to be served by a centers $i \leq l$ and center $i' \leq r$, respectively. Now since $l \leq r$, any large arm counted in $\mathbf{D}_l[q_1]$ must be counted in $\mathbf{S}_r[q_2]$, i.e., $\mathbf{S}_r[q_2] \geq \mathbf{D}_l[q_1]$, and we must have at least $\mathbf{S}_r[q_2]$ clients in $[a, b)$. Note that $\mathbf{S}_r[q_2] - \mathbf{D}_l[q_1]$ correspond to long arms that start in interval $V_{l,r}$. Similar arguments must be made for when $a \leq l$ and $\mathbf{D}_r$ and $\mathbf{S}_l$. For intervals containing l or r, we have to subdivide the interval, e.g., $[a, r)$ and $[r, b)$, and then check the consistency.

The number of table entries is polynomial, because k', l, r, s are in $O(n)$ and β is a polynomial in n and $\frac{1}{\epsilon}$ and the deficiency and surplus vectors, in total, can take one of $O(n^{1+1/\epsilon^2})$ values. We shortly explain how one can compute the values A in polynomial time through dynamic programming. After that, to find out if there is a feasible solution having perceived cost $(1+5\epsilon)B$, one simply needs to look at the value of $A[k, 1, n, \perp, 0, \mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{0}]$, where $\mathbf{0}$ is a vector having $1 + 1/\epsilon^2$ zero components.

The recurrence. In the remaining of the section, we explain how the value of a table entry is calculated. We call a subproblem *feasible* if $A[k', l, r, s, \beta, \mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r]$ is TRUE. The recurrence is somewhat involved, so we explain the main ideas behind it; more details can be found in the paper [4].

Base case. The base case is when $k' = 0$ and $l = r$. Since $k' = 0$, we are not allowed to open a facility at v_r (r might not be a facility anyway). There are two possible main cases based on whether s can serve a possible client at r or not.

- $s = \perp$ and so $\beta = 0$ or $s \neq \perp$ but $\beta = 0$.
  If r is a client, then r has to be served by a large arm coming from outside of $V_{l,r}$. If r is served by a large arm entering from left, so $\mathbf{D}_l[0]$ must be non-zero and moreover, values of $\mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r$ must be consistent, i.e., $\mathbf{D}_l[0] = \mathbf{S}_r[0]+1$, $\mathbf{D}_l[q] = \mathbf{S}_r[q]$ for $1 \leq q \leq \epsilon^{-2}$, and $\mathbf{D}_r = \mathbf{S}_l$. Similarly, if r is served by a large arm entering from right, then we must have $\mathbf{D}_r[0] = \mathbf{S}_l[0] + 1$, $\mathbf{D}_r[q] = \mathbf{S}_l[q]$ for $1 \leq q \leq \epsilon^{-2}$, and $\mathbf{D}_l = \mathbf{S}_r$. So if either of these conditions hold, then the subproblem is feasible.

  If r is a facility, then it cannot be opened as $k' = 0$. So we only check consistency of $\mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r$, i.e., $\mathbf{D}_l = \mathbf{S}_r$ and $\mathbf{D}_r = \mathbf{S}_l$.

- $s \neq \perp$ and $\beta \neq 0$.
  If r is a client, it has to be served by a small arm originating at s, so β must be equal to $c(s, r)$ and it must be smaller than ϵB. The vectors $\mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r$ must be consistent, i.e., $\mathbf{D}_l = \mathbf{S}_r$ and $\mathbf{D}_r = \mathbf{S}_l$.

  If r is a facility, then the subproblem is infeasible by the definition.

Recursive step. Next, we show how to determine if $A[k', l, r, s, \beta, \mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r]$ is TRUE when the parameters do not represent a base case by relating its value to values of smaller problems. In what follows, by *guessing* a parameter, we mean that we try all *polynomially* many possible values of that parameter and if one of them results in a feasible solution, we set the value of the current subproblem to TRUE. We consider two cases regarding the value of s:

Case (1): $s \neq \perp$ and $\beta > 0$.
Without loss of generality, suppose $s < l$. There must be a small client j with $l \leq j \leq r$ covered by s. We guess j to be the leftmost such small client along with how many of k' facilities in $V_{l,r}$ are in $V_{l,j-1}$, call this k'' (the rest of facilities, $k' - k''$, will be in $V_{j+1,r}$). For subproblem constructed for $V_{l,j-1}$, no small arm can enter $V_{l,j-1}$, and for subproblem constructed for $V_{j+1,r}$, the center outside $V_{j+1,r}$ is s with allowed load of $\beta' = \beta - c(s, j)$. We can also guess the large arms leaving and/or entering $V_{l,j-1}$ as well as $V_{j+1,r}$ and in polynomial time, we check if these vectors are consistent with each other as well as $\mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r$. So $A[k', l, r, s, \beta, \mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r]$ is set to TRUE if one of the following expressions is TRUE (see Figure 3.4).

$$A[k'', l, j-1, \perp, 0, \mathbf{D}_l, \mathbf{D}_{j-1}, \mathbf{S}_l, \mathbf{S}_{j-1}] \wedge A[k' - k'', j+1, r, s, \beta - |v_s - v_j|, \mathbf{D}_{j+1}, \mathbf{D}_r, \mathbf{S}_{j+1}, \mathbf{S}_r]$$

For some $l \leq j \leq r$ such that I_{sj} is a small arm and $|v_s - v_j| \leq \beta$, some $0 \leq k'' \leq k'$,

, and $\mathbf{D}_{j-1}, \mathbf{S}_{j-1}, \mathbf{D}_{j+1}, \mathbf{S}_{j+1}$ consistent with $\mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r$,

(using the assumption that j is served with a small arm).

Note that when $j = l$, we just check one other subproblem, namely $A[k', j+1, r, s, \beta - |v_s - v_j|, \mathbf{D}_{j+1}, \mathbf{D}_r, \mathbf{S}_{j+1}, \mathbf{S}_r]$ (we must have I_{sj} is a small arm, and $|v_s - v_j| \leq \beta$). Similarly, when $j = r$, we just check one subproblem, namely $A[k', l, j-1, \perp, 0, \mathbf{D}_l, \mathbf{D}_{j-1}, \mathbf{S}_l, \mathbf{S}_{j-1}]$ (we must have I_{sj} is a small arm, and $|v_s - v_j| = \beta$). If $l = r$ then k' has to be non-zero (otherwise we are in a base case), and since there is no facility to open at j, we say the subproblem is infeasible. Similar argument can be made when $s > r$ (in this case, we guess the rightmost client served by s).

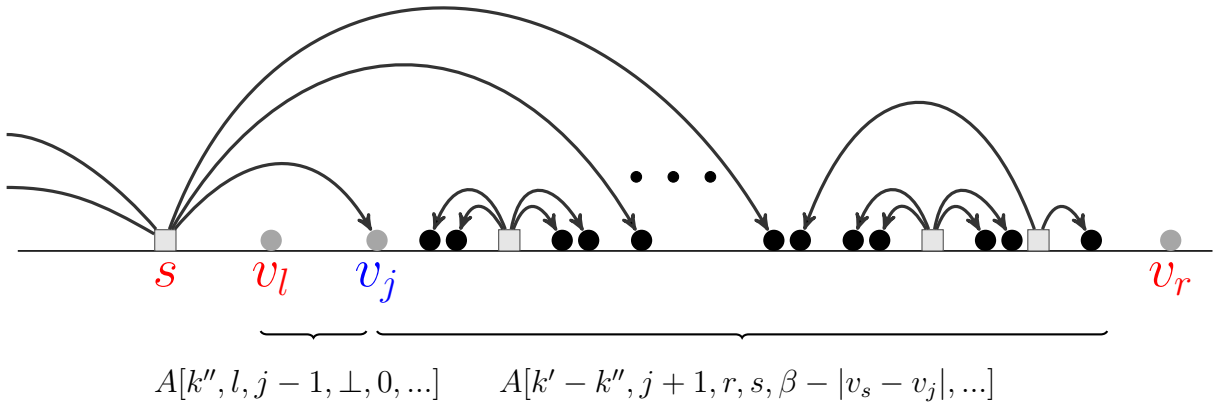


Figure 3.4: Case (1) of recursive step. Only small arms are drawn in the figure, and for clarity large arms are not drawn (large arms can enter and/or leave the interval).

Case (2): $s \neq \perp$ and $\beta = 0$, or $s = \perp$.
We consider two subcases regarding value of k':

Case (2-a): $k' = 0$.
First note that in this case, $l \neq r$ (otherwise we are in a base case). All clients in $V_{l,r}$ must be covered by large arms from centers (facilities) outside the interval. If l is a facility then it must be closed so we recursively check $A[0, l+1, r, s, 0, \mathbf{D}_l', \mathbf{D}_r', \mathbf{S}_l', \mathbf{S}_r']$ where the new deficiency and surplus vectors are obtained from $\mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r$ after taking into account the number α of multiples of $\epsilon^2 B$ between v_l and v_{l+1}. If this is not possible, e.g. if $\mathbf{D}_r(q) > 0$ for some $q < \alpha$ or similarly, then the subproblem is not feasible.
So, suppose that l is a client. First, assume l is covered by a large arm from the left. Then $\mathbf{D}_l[0] > 0$ and we use one to cover client l. In this case, define $\mathbf{D}_l', \mathbf{D}_r', \mathbf{S}_l', \mathbf{S}_r'$

to reflect the fact that this one large arm covers l and the perceived length of the remaining ones accounted for by $\mathbf{D}_l, \mathbf{S}_l$ that entered or exited by v_r have a different perceived length depending on the number of multiples of $\epsilon^2 B$ between v_l and v_{l+1} (again, if this is not possible then the subproblem is not feasible).

Then we declare the subproblem to be feasible if and only if the value of $A[0, l+1, r, s, 0, \mathbf{D}_l{}', \mathbf{D}_r{}', \mathbf{S}_l{}', \mathbf{S}_r{}']$ is TRUE. A similar argument works if l is covered by a large arm from the right.

Case (2-b): $k' > 0$.

Note that since $s \neq \perp$ and $\beta = 0$, or $s = \perp$, no small arm can enter $V_{l,r}$. Consider the set of centers in $V_{l,r}$. The small-span (interval of small arms) of these centers forms a laminar family. Consider the roots of the forest of this laminar family and let s' be the center corresponding to the leftmost root; we guess s' (see Figure 3.5) along with the contribution of small arms originating at s' going to the left (call β') and the right (call β''), and also the number of centers located between l and i, say k''. Observe that the small-span of i is not contained in the small-span of any other center in $V_{l,r}$. Center s' has at most $3/\epsilon$ large arms. We guess the large arms of s' along with large arms entering/leaving $V_{l,s'-1}$ and $V_{s'+1,r}$. For all the guesses that the perceived cost of c' is at most $(1+5\epsilon)B$ and the large arms are consistent with each other as well as $\mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r$. So $A[k', l, r, s, \beta, \mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r]$ is set to TRUE if one the following expressions is TRUE (see Figure 3.5).

$$A[k'', l, s'-1, s', \beta', \mathbf{D}_l, \mathbf{D}_{s'-1}, \mathbf{S}_l, \mathbf{S}_{s'-1}] \wedge A[k'-k''-1, s'+1, r, s', \beta'', \mathbf{D}_{s'+1}, \mathbf{D}_r, \mathbf{S}_{s'+1}, \mathbf{S}_r]$$

For some $l \leq s' \leq r$, $0 \leq k'' \leq k'-1$

guessed $\leq 3/\epsilon$ *long arms such that the perceived cost at* s' *is at most* $(1+5\epsilon)B$,

guessed large arms and $\mathbf{D}_{s'-1}, \mathbf{S}_{s'-1}, \mathbf{D}_{s'+1}, \mathbf{S}_{s'+1}$ *consistent with* $\mathbf{D}_l, \mathbf{D}_r, \mathbf{S}_l, \mathbf{S}_r$.

3.4 A constant-factor approximation algorithm for MLkFL in star metrics

We now consider MLkFL in star metrics, but in the more-general setting where each client j has an integer demand d_j that may be split integrally across various open facilities; we call this

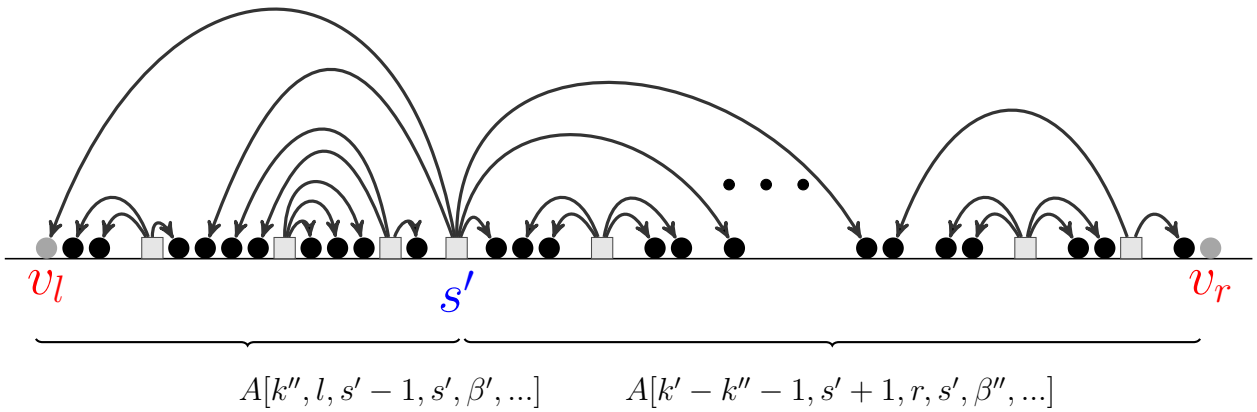


Figure 3.5: Case (2-b) of recursive step.Only small arms are drawn in the figure, and for clarity large arms are not drawn (large arms can enter and/or leave the interval).

an *integer-splittable assignment*. The load of a facility i is now defined as $\sum_j x_{ij}c(i,j)$ where $x_{ij} \in \mathbb{Z}_{\geq 0}$ is the amount of j's demand that is served by i. We devise a 14-approximation algorithm for this problem. At a high level, our approach is similar to the one used to obtain the PTAS for line metrics. We again "guess" the optimal value B. We argue via a slightly different uncrossing technique that if $B \geq L^{opt}$, then there exists a well-structured *fractional* solution with the maximum load at most $6B$, and use DP to obtain a fractional solution with the maximum load at most $12B$. This can then be converted to an integer-splittable assignment with the maximum load at most $14B$ using the GAP-rounding algorithm, since it is easy to ensure via some preprocessing that $c(i,j) \leq 2B$ for every facility i and client j. Thus, we either determine that $B < L^{opt}$ or obtain a solution with the maximum load at most $14B$.

Theorem 3.4.1. *There is a 14-approximation algorithm for* MLkFL *on star metrics with non-uniform demands and integer-splittable assignments.*

Let r be the root of the star graph defining the star metric, V denote the set of all leaf nodes, and let $c_i = c(i,r)$ for leaf i. We may assume that $r \notin \mathcal{F} \cup \mathcal{D}$ since we can add an extra leaf with distance zero to r. Number the nodes of V from 1 to n so that $c_1 \leq c_2 \leq \cdots \leq c_n$. Let d_j be the integer demand of client $j \in \mathcal{D}$. Recall that we consider integer-splittable assignments, where each open facility serves an integer amount of the demand (possibly 0) of each client. As before, we often refer to a pair (i,j), where $i \in \mathcal{F}, j \in C$, as an arm.

Let B be our current guess of the optimal value. Our goal is to either certify that $B < L^{opt}$, or find a solution with the maximum load at most $14 \cdot B$. We may assume that $c_i \leq B$ for all $i = 1, \ldots, n$. Otherwise, if $c_i > B$, then no client may assign any demand to i (if $i \in \mathcal{F}$) in any integer-splittable assignment; also, if $d_i > 0$, then all of d_i must be served by i. Thus, we can remove i from V, and in the latter case, decrease k by 1, and proceed with the smaller instance.

In Section 3.4.1, we show that if $B \geq L^{opt}$, then there exist k facilities, and a well-structured fractional assignment of clients to these facilities of cost (i.e., maximum load) at most $6B$. In Section 3.4.2, we devise a dynamic programming approach that finds k facilities and a well-structured fractional assignment of clients to these facilities of cost at most $12B$ provided there is such a solution of the cost at most $6B$. Combining these results, if $B \geq L^{opt}$, we can find k facilities and a fractional assignment of clients to these facilities that has the maximum load at most $12B$. Finally, using Theorem 3.3.6, we can round this solution to an integer solution while increasing the maximum load by at most $\max_{i \in \mathcal{F}, j \in \mathcal{D}} c(i,j) \leq 2B$.

3.4.1 A well-structured near-optimal solution

We now show that if $B \geq L^{opt}$, then there exists a fractional assignment satisfying various nice structural properties, which will then enable us to find such a solution via DP (Section 3.4.2). Let F_B be the set of open facilities in some integer-splittable solution having the maximum load at most B. Consider the following LP.

$$\min \sum_{i \in F_B} \sum_{j \in \mathcal{D}} c(i,j) \cdot x_{ij} \qquad \text{(S-P)}$$

$$\text{s.t.} \quad \sum_{i \in F_B} x_{ij} = d_j \qquad \forall j \in \mathcal{D}$$

$$\sum_{j \in \mathcal{D}} c(i,j) \cdot x_{ij} \leq B \qquad \forall i \in F_B$$

$$x_{ii} = d_i \qquad \forall i \in F_B$$

$$x_{ij} \geq 0 \qquad \forall i \in F_B, j \in \mathcal{D}.$$

Given a solution x to (S-P), we say that arms (i, j') and (i', j) *cross* in x if $x_{ij'} \cdot x_{i'j} > 0$ and $c(i,j) \cdot c(i',j') < c(i',j) \cdot c(i,j')$. We know that (S-P) is feasible. We prove that the optimal solution to (S-P) does not have any crossing arms in its support.

Lemma 3.4.2. *The optimal solution to* (S-P) *does not have any crossing arms in its support.*

Proof. Let x^* be an optimal solution to (S-P). Suppose (i, j') and (i', j) cross in x^*. If $c(i, j) = 0$ then simply update x by moving all of j's demand to i. Similarly, if $c(i', j') = 0$ then move all of the demand of j' to i'. In both cases, the objective value of x^* decreases, which is a contradiction. So suppose that $c(i, j) \cdot c(i', j') > 0$.

For some $\epsilon, \epsilon' > 0$ to be specified shortly, we create a new assignment x that agrees with x^* in all center-client pairs except that:

- $x_{ij} = x^*_{ij} + \epsilon, \quad x_{i'j} = x^*_{i'j} - \epsilon.$
- $x^*_{i'j'} = x_{i'j'} + \epsilon', \quad x^*_{ij'} = x_{ij'} - \epsilon'.$

It must be that $c(i, j) < c(i, j')$ or $c(i', j') < c(i', j)$ so assume, without loss of generality, that $c(i, j) < c(i, j')$. We chose the largest possible values for ϵ, ϵ' such that the load at i does not change, i.e., $\epsilon \cdot c(i, j) = \epsilon' \cdot c(i, j')$ and $x_{i'j}, x_{ij'}$ are non-negative, so either $x_{i'j} = 0$ or $x_{ij'} = 0$ will be zero while the other remains nonnegative. The change in the load of i' as well as the change in objective value is given by

$$\epsilon' \cdot c(i', j') - \epsilon \cdot c(i', j) = \epsilon \left(\frac{c(i, j) \cdot c(i', j')}{c(i, j')} - c(i', j) \right)$$

which is nonpositive because $c(i, j) \cdot c(i', j') < c(i, j') \cdot c(i', j)$. This yields a contradiction. □

Observe that the above uncrossing property is stronger than the uncrossing that we achieved for line metrics, where we only ensured that small arms do not cross. Figure 3.6 illustrates all the (non-symmetric) cases that count as crossing. The figures on the right show the result after modifying x as described in the above proof (assuming $x_{i'j}$ becomes zero).

Definition 3.4.3. *A fractional solution x to* (S-P) *is* well-structured *if we can partition $V = \{1, \ldots, n\}$ into consecutive subsequences $V_1, V_2, \ldots, V_m$ such that:*

- *For each V_a and each $j \in V_a \cap \mathcal{D}$, we have $x_{ij} = 0$ for $i \notin V_a$. That is, each client is completely served within its partition.*
- *For each V_a, at least one of the following holds:*
  1. $|F_B \cap V_a| = 1$.
  2. $x_{ij} = 0$ *for all* $j \in V_a \cap \mathcal{D}$ *and* $i < j$ *(clients are only satisfied by centers to the right).*

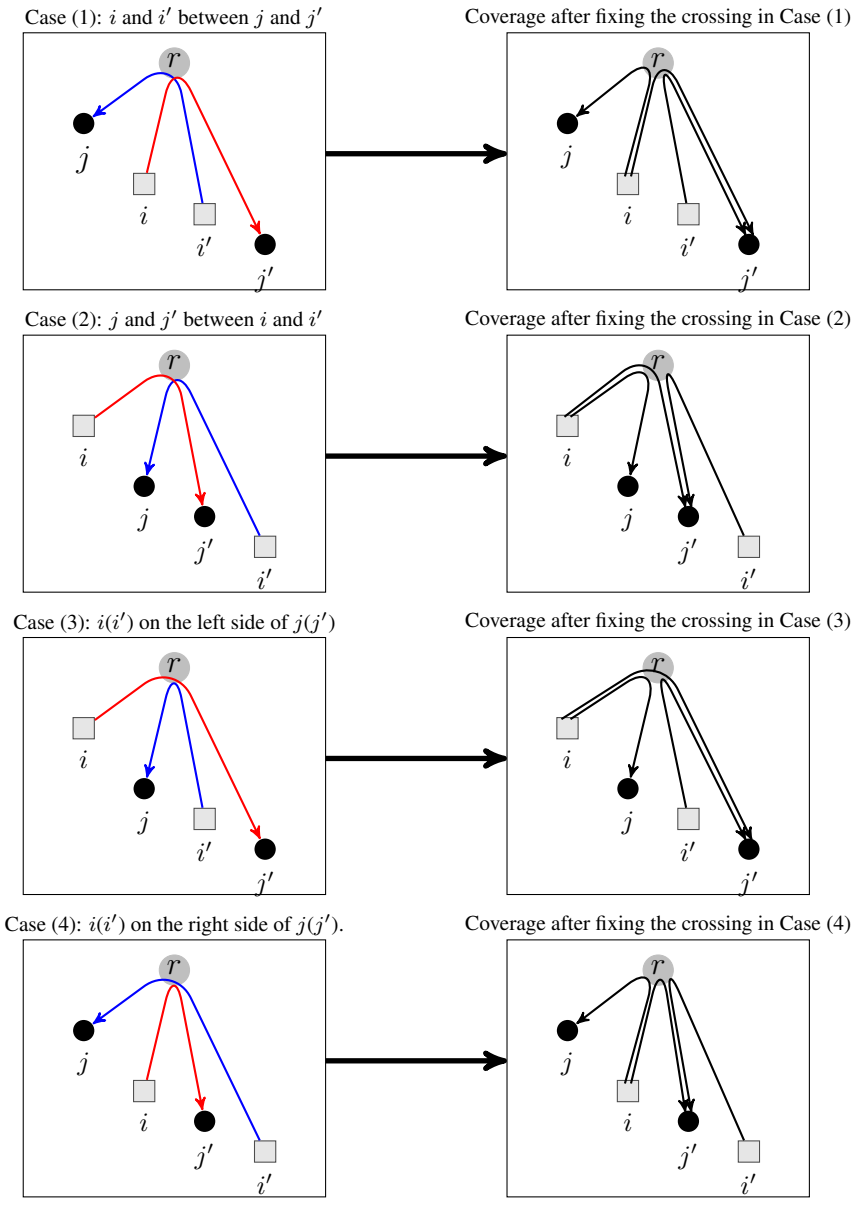


Figure 3.6: Fixing the crossing of two arms.

3. $x_{ij} = 0$ *for all* $j \in V_a \cap \mathcal{D}$ *and* $i > j$ *(clients are only satisfied by centers to the left).*

Lemma 3.4.4. *There is a well-structured fractional solution* x *to* (S-P) *with the maximum load at most* $6B$.

Proof. Let x^* be an optimal solution to (S-P). So x^* has the maximum load at most B, and by Lemma 3.4.2, it does not have any crossings. We initialize variable x to x^*. We modify x in a number of steps that do not increase the maximum load by more than a constant factor.

- **Step 1) Ensuring all clients are served in only one direction.**
  Consider client $j \in \mathcal{D}$. If $j \in F_B$, all the demands at j must be satisfied by the collocated center by the constraint $x_{ii} = d_i$ for all $i \in F_B$, so we can assume $x_{ij} = 0$ for $i \neq j$. If $j \notin F_B$, either $\sum_{i<j} x_{ij} \geq \frac{d_j}{2}$ or $\sum_{i>j} x_{ij} \geq \frac{d_j}{2}$. Suppose the former is true (the latter is similar). Then we simply set $x_{ij} = 0$ for $i > j$ and scale the x_{ij} with $i < j$ uniformly until they sum to d_j again. After doing this for all clients, we have that x_{ij} at most doubles for each center-client pair so the maximum load is at most $2B$.

- **Step 2) Ensuring all centers either serve clients only to the left or only to the right, or form their own consecutive partition.**
  Let i be any center in F_B with $x_{ip}, x_{iq} > 0$ for two clients $p < i < q$. If there is no such center, then this step is done. We will modify the assignment to i and, perhaps, some nearby centers and then form a consecutive subsequence of V whose only center is i.

  Consider the rightmost center $i_L \in F_B$ such that $i_L < i$, see Figure 3.7. If there is no center to the left of i, then the operations in this paragraph are skipped. Otherwise we update the assignment x in the following way: For any client $j < i_L$ with $x_{ij} > 0$, we update $x_{i_L j} \leftarrow x_{i_L j} + x_{ij}$ and $x_{ij} \leftarrow 0$. Note that this modification does not introduce any crossings in x as each client j is now assigned to a closer location. Since $x_{i_L i_L} = d_{i_L}$ by (S-P) constraint, we can now claim that any client $j < i$ with $x_{ij} > 0$, has to be on the right of i_L, i.e., $i_L < j$.

  Similarly, if there is a leftmost center $i_R \in F_B$ with $i_R > i$ then move all assignment x_{ij} with $j > i_R$ to i_R. Again after these modifications, no new crossing is introduced, and any client $j > i$ with $x_{ij} > 0$, has to be on the left of i_R, i.e., $j < i_R$.

  Let j_L be the leftmost client with $x_{ij_L} > 0$, if no such client exists define $j_L = n + 1$. Similarly, let j_R be the rightmost client with $x_{ij_R} > 0$, if no such client exist let $j_R = 0$. Define interval V_a to be $j_1, j_2, \cdots, j_m$ where $j_1 = \min(j_L, i)$ and $j_m = \max(i, j_R)$ (See Figure 3.7). Note that partition V_a satisfies $|F_B \cap V_a| = 1$. We claim that all clients in V_a are completely assigned to i. Let j be an arbitrary client in V_a. Without loss of generality assume $j < i$ (the other case is similar). Note that in this case $j_1 \leq j < i$, so $j_L < i$. We have two possible cases:

  - $j = j_L$. By definition of j_L, $x_{ij_L} > 0$. By step 1, we know that j_L is served in one direction which has to be right as i is on the right of j_L. Suppose facility $i' > i$,

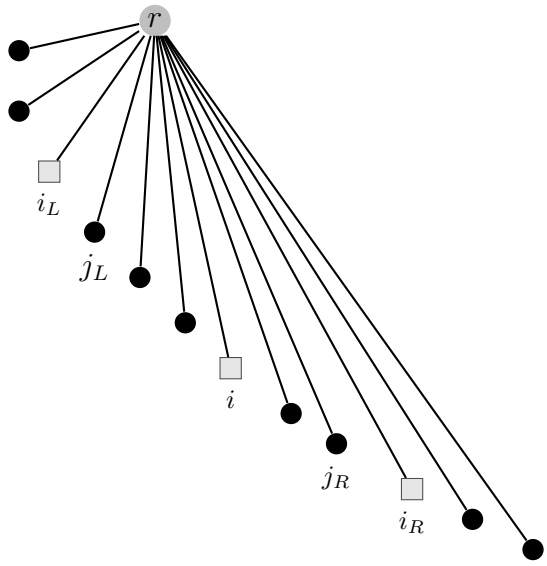


Figure 3.7: Moving client assignments. Black solid circles denote clients with $x_{ij} > 0$.

serves j_L. Since i in x^* serves $q > i$ (initial condition for picking i), this gives us a contradiction as $x^*_{i'j_L} \cdot x^*_{iq} > 0$ and arms (i, q) and (i', j_L) cross (Case (1) or Case (4) depending on the position of q with respect to i', see Figure 3.6).

– $j > j_L$. First, we show j cannot be served by a facility $i' < i$. Let us assume the contrary. By definition of i_L, we have $i' \leq i_L$, and hence arms (i', j) and (i, j_L) cross which gives us a contradiction as $x^*_{i'j} \cdot x^*_{ij_L} > 0$ (Case (2) in Figure 3.6). Second, we show j cannot be served by facility $i' > i$. Again let us the contrary. Since arms (i, q) and (i', j) cross for solution x^* while $x^*_{i'j} \cdot x^*_{iq} > 0$ (Case (1) or Case (4) depending on the position of q with respect to i', see Figure 3.6).

Note that partition V_a satisfies $|F_B \cap V_a| = 1$ and that all clients in V_a are completely assigned to the sole center in V_a. Removing V_a from V effectively divides the instance into two subinstances. We only note that $x_{i'j'} = 0$ for any $j' < i < i'$ or $i' < j < j'$. Otherwise, if (say) $j' < i < i'$ has $x_{i'j'} > 0$ then this would cross arm (i, q) that was assigned to i in x^* which gives us a contradiction.

We claim the maximum load after performing the second step has increased by at most $4B$. The only times the load increases are when some centers of the form i_L or i_R have some x_{ij} reassigned to them. Each center i' can be some center of the form i_L only once and of the form i_R only once. Moreover, the load of i' cannot be increased in both cases.

Suppose the contrary, i.e., i' is i_L for some center i_0 and i_R for some center i_1 where there exist clients $j_0 < i'$ and $j_1 > i'$ with $x^*_{i_0 j_0}, x^*_{i_1 j_1} > 0$. Since the arms (i_0, j_0) and (i_1, j_1) cross, we get a contradiction. So at most one center can increase the load of i', therefore it remains to show that the increase in the load if i' is i_L or i_R of some center is at most $4B$.

First assume that i' is of the form i_L for some facility i. Since $c(i', j) < c(i, j)$ for each client $j \neq i$, the load of i' after the process is increased by the portion of load of i that corresponds to serving clients $j < i_L$ which is at most $2B$. Now assume i' is of the form i_R for some facility i. Note that any client j moved to i' has $c(i, j) < c(i', j)$ but $c(i', j) < 2c(i, j)$ as $j > i'$. So the load increase on i' is at most twice the load on i due to clients $j > i'$; therefore, the load increase is at most $4B$.

- **Step 3) Dividing the remaining instance.**
  Now each $j \in \mathcal{D} \setminus F_B$ assigns all demand either completely to the left or completely to the right. Similarly, any $i \in F_B$ collects demand either completely from the left or completely from the right. Say that $j \in F_B \cup \mathcal{D}$ "goes left" if $j \notin F_B$ and $x_{ij} > 0$ only for $i < j$ **or** $j \in F_B$ and $x_{jj'} > 0$ only for $j' > j$. Say that $j \in F_B \cup \mathcal{D}$ "goes right" if $j \notin F_B$ and $x_{ij} > 0$ only for $i > j$ **or** $j \in F_B$ and $x_{jj'} > 0$ only for $j' < j$. If j is served by a collocated facility and it is the only client served by this facility, we say j is "neutral". Now V naturally breaks up into maximal consecutive intervals of nodes, each of which only includes clients that "go left"/"neutral" or "go right"/"neutral". These form the remaining partitions V_a.

  The only thing left to note is that a client is completely served within its partition. Suppose $j \in V_a$ and that j "goes left". If j is not completely served within V_a, then there exists facility $i < j$ not in V_a which serves j. Since V_a is maximal partition, there exists the preceding partition $V_{a'}$ that "goes right". Since $V_{a'}$ must include a client j' that "goes right" by maximality of intervals, there is some $i' > j'$ with $x_{i'j'} > 0$. Since arms (i, j) and (i', j') cross (Case (1) or Case (4) in Figure 3.6), this cannot happen. Thus, j cannot assign any demand to a center to the left of V_a. □

3.4.2 A dynamic-programming algorithm for finding a well-structured solution

In this section, we describe our dynamic programming approach for finding a well-structured solution with the cost at most $12B$ with fractional assignment. Note that since by the preprocessing done in the beginning, $c_j \leq B$ for $1 \leq j \leq n$, using the GAP-rounding algorithm, we can obtain an integral solution with the maximum load at most $14B$.

Our dynamic programming approach here is much simpler compared to the dynamic programming approach for line metrics. This is due to the useful properties that a well-structured solution has. In the following, we describe a dynamic programming approach, which utilizes the following proxy for measuring the cost of an arm (i, j): we use $\max(c_i, c_j)$ to measure the cost of (i, j). This makes the calculation of a load of a facility for our dynamic programming much easier and since the actual cost of the arm is at most twice this proxy cost, we lose a factor 2 for the actual load of a facility. So in fact, our dynamic programming finds a well-structured solution with the maximum load $6B$ with the new method of measurement.

We describe the dynamic programming approach in two steps. For notational convenience, we set $d_j = 0$ if $j \notin \mathcal{D}$, and think of every node as a client (but with potentially 0 demand). First, for any subsequence $V_{l,r} = \{l, \ldots, r\}$ for $1 \leq l \leq r \leq n$, and any $1 \leq k' \leq k$ we describe a boolean value $I(l, r, k')$ that is TRUE if and only if one of the following is TRUE.

1. $k' = 1$ and there is some facility $l \leq i \leq r$ in $\mathcal{F}$ such that assigning each client from $V_{l,r}$ to i places proxy load at most $6B$ on i.

2. There is a set $F \subseteq V_{l,r} \cap \mathcal{F}$ of k' facilities, and a fractional assignment $(x_{ij})_{i \in F, j \in V_{l,r}}$ such that for every $j \in V_{l,r}$: $x_{ij} = 0$ for $i < j$ (j goes right) and $\sum_{i \in F} x_{ij} = d_j$, and for every $i \in F$: $c_i \cdot \sum_{j \in V_{l,r}: j < i} x_{ij} \leq 6B$.

3. There is a set $F \subseteq V_{l,r} \cap \mathcal{F}$ of k' facilities, and a fractional assignment $(x_{ij})_{i \in F, j \in V_{l,r}}$ such that for every $j \in V_{l,r}$: $x_{ij} = 0$ for $i > j$ (j goes left) and $\sum_{i \in F} x_{ij} = d_j$, and for every $i \in F$: $\sum_{j \in V_{l,r}: j > i} c_j \cdot x_{ij} \leq 6B$.

If we can compute $I(l, r, k')$ for all (l, r, k') tuples (as well as the solution that generates it), then we claim that we are done. Observe that if $I(l, r, k') = \text{TRUE}$ when $k' > 1$, then the fractional assignment x corresponding to $I(l, r, k')$ induces a maximum load of $12B$ on the centers opened from $V_{l,r}$. If $I(l, r, k')$ is TRUE due to the second condition, then this is because if $x_{ij} > 0$, then $c(i, j) \leq 2c_i$, and we have $c_i \cdot \sum_{j \in V_{l,r}} x_{ij} \leq 6B$. Similarly, if $I(l, r, k')$ is TRUE due to the third condition, then $x_{ij} > 0$ implies that $c(i, j) \leq 2c_j$, and we have $\sum_{j \in V_{l,r}} c_j \cdot x_{ij} \leq 6B$.

Now it is a simple matter to determine, using another dynamic program, how to partition V into consecutive intervals $[j_0 + 1, j_1], [j_1 + 1, j_2] \ldots, [j_{m-1} + 1, j_m]$ where $j_0 := 0$ and $j_m := n$ with positive integers $k'_1, \ldots, k'_m$ summing to k such that $I(j_{p-1} + 1, j_p, k'_p) = \text{TRUE}$ for each $1 \leq p \leq m$.

To wrap up the proof, we describe how to compute $I(l, r, k')$. We can associate a table for each case in the definition of $I(l, r, k')$ and describe how to calculate them.

1. Let T_1 be an $n \times n$ table corresponding to the first case. $T_1(l, r)$ is TRUE if there exists a center $l \leq i \leq r$ such that assigning each client from $V_{l,r}$ to i places load at most $6B$ on i which assigns value TRUE. There are $O(n)$ possible choices for the center, and table entry can be computed in $O(n)$ time.

2. For the second case, we consider an $n \times n \times n$ table T_2. In order to compute the value of each table entry $T_2(l, r, k')$, we use an auxiliary table $f(i, k'')$ for $l \leq i \leq r, 0 \leq k'' \leq k'$. $f(i, k'')$ stores the minimum possible excess demand $\sum_{j \leq i} \left(d_j - \sum_{i' \in F'} x_{i'j}\right)$ among all ways of choosing k'' centers F' in $\{l, \ldots, i\}$ and fractionally assigning up to d_j units of demand of each client $j \leq i$ to centers in F' such that no center $i' \in F'$ is assigned more than $6B/c_{i'}$ units of demand from clients $j < i'$ and $x_{i'j} = 0$ if $i' < j$.

   The base cases with $i = l$ are easy: $f(i, 0) = d_i$ and $f(i, k'') = 0$ if $i \in \mathcal{F}$ and $k'' > 0$; we set $f(i, k'') = \infty$ if $k'' > 0$ and $i \notin \mathcal{F}$. Also, for $i > l$ but $k'' = 0$, we have $f(i, k'') = f(i-1, k'') + d_i$. Otherwise, if $i > l$ and $k'' > 0$ we have

$$f(i, k'') = \begin{cases} \min\{\max\{0, f(i-1, k''-1) - 6B/c_i\}, f(i-1, k'') + d_i\}; & \text{if } i \in \mathcal{F} \\ f(i-1, k'') + d_i & \text{otherwise.} \end{cases}$$

   The first term in the $\min$ says that if we open i, then we assign as much leftover demand that we can. The second term says that if we do not open i then all of the demand at i must go to the right of i. Once we compute this, we set $T_2(l, r, k')$ to TRUE if and only if $f(r, p) = 0$.

3. For the last case, we consider a similar dynamic programming algorithm in a "right-to-left" manner, except we are concerned with the minimum value of $\sum_{j \geq i} c_j \cdot \left(d_j - \sum_{i' \in F'} x_{i'j}\right)$. We associate the table $T_3(l, r, k')$ for this case, and we use the auxiliary table $g(i, k'')$ for $l \leq i \leq r, 0 \leq k'' \leq k'$. $g(i, k'')$ is the minimum possible excess load $\sum_{j \geq i} c_j \left(d_j - \sum_{i' \in F'} x_{i'j}\right)$ among all ways to choose k'' centers F' in $\{i, \ldots, j_R\}$ and fractionally assign up to d_j units of demand of each client $j \geq i$ to centers in F' such that no center $i' \in F'$ is assigned more than $6B$ and $x_{i'j} = 0$ if $i' > j$.

   The base cases with $i = r$ are easy: $g(i, 0) = c_i \cdot d_i$ and $g(i, k'') = 0$ if $i \in \mathcal{F}$ and $k'' > 0$; we set $g(i, k'') = \infty$ if $k'' > 0$ and $i \notin \mathcal{F}$. Also, for $i < r$ but $k'' = 0$ we have $g(i, k'') = g(i+1, k'') + c_i \cdot d_i$. Otherwise, if $i < r$ and $k'' > 0$ we have

$$g(i, k'') = \begin{cases} \min\{\max\{0, g(i+1, k''-1) - 6B\}, g(i+1, k'') + c_i \cdot d_i\}; & \text{if } i \in \mathcal{F} \\ g(i+1, k'') + c_i \cdot d_i & \text{otherwise.} \end{cases}$$

The first term in the $\min$ says that if we open i, then we assign as much leftover load that we can. The second term says that if we do not open i then all of the demand at i must go to the left of i. Once we compute this, we set $T_3(l, r, k')$ to TRUE if and only if $g(l, k') = 0$.

Finally, once all of the $T_1(l, r)$, $T_2(l, r, k')$ and $T_3(l, r, k')$ are computed, we can set $I(l, r, 1) = T_1(l, r)$ and $I(l, r, k') = T_2(l, r, k') \vee T_3(l, r, k')$ for $k' > 1$.

3.5 Hardness of the problem

We now present our hardness result and prove that MLkFL is strongly *NP*-hard on line metrics.

Theorem 3.5.1. *Minimum-load k-facility location is strongly NP-hard even in line metrics.*

Proof. We reduce from 3-partition, where we are given $n = 3k$ integers $b_1, \ldots, b_n$ and a bound B such that $\sum_{i=1}^{n} b_i = kB$. The goal is to partition the integers into k groups such that the sum of the integers in any group is at most B. It is NP-complete to determine if there is a feasible solution, even when $b_i \leq 2^{16}n^4$ and $\frac{B}{4} < b_i < \frac{B}{2}$ for each i (e.g. [35]). In particular, any feasible solution will have precisely three integers in each group of the partition.

We create an instance of MLkFL on the line by creating two groups of clients. First, for each point $p \in \{-\frac{k-1}{3k}, -\frac{k-2}{3k}, \ldots, -\frac{1}{3k}, 0\}$, we place $3k^2(B + 1)$ clients at p. Next, for each integer $b_i, 1 \leq i \leq n$, we add a single client at position b_i. Let N be the number of clients in the resulting instance and notice that all values have bit complexity bounded by a polynomial in N. The claim is that there is a solution with cost $B + \frac{k-1}{k}$ if and only if the 3-partition problem is a **yes** instance.

First, suppose there is a partition of the integers $b_1, \ldots, b_n$ into k groups $G_1, \ldots, G_k$ such that the sum of the integers in any group G_i is B. For each $1 \leq i \leq k$ we create a star with center at $-\frac{i-1}{3k}$, assign all clients located at this center to this star, and also assign the clients in group G_i to this star. The only clients that move some positive distance to the center of the star are those from the group G_i, and they move a total distance of $B + \frac{i-1}{k} < B + \frac{k-1}{k}$.

Conversely, suppose there is a solution with the maximum load at most $B + \frac{k-1}{k}$. First, we claim that every point of the form $-\frac{i}{3k}, 0 \leq i < k$ must be the center of a star. Otherwise, the $3k^2(B+1)$ clients at this location must be assigned to other stars. The minimum distance each of these client travels is $\frac{1}{3k}$ and one of the open centers receives at least $3k(B + 1)$ of these clients, so its load is at least $B + 1 > B + \frac{k-1}{k}$. Therefore, the centers are at locations $-\frac{i}{3k}, 0 \leq i < k$.

Since $\frac{B}{4} < b_i < \frac{B}{2}$, then every star must contain exactly three clients corresponding to integers $b_1, \ldots, b_n$ in the 3-partition instance. Without loss of generality, say b_1, b_2, b_3 are the

three integers in some star. The total distance they travel lies between $b_1 + b_2 + b_3$ and $b_1 + b_2 + b_3 + \frac{k-1}{k}$ so $b_1 + b_2 + b_3 \leq B$. Therefore, if we let G_i be the clients corresponding to integers $b_1, \ldots, b_n$ that are in the star with center $-\frac{i-1}{3k}$ for each $1 \leq i \leq k$, then $G_1, \ldots, G_k$ is a feasible solution to the 3-partition problem. □

3.6 Integrality-gap lower bounds

In this section, we demonstrate that a natural configuration-style LP has an unbounded integrality gap in Theorem 3.6.1. Let $(\mathcal{D} \cup \mathcal{F}, c, k)$ be an MLkFL instance. Given a candidate "guess" B of the optimal value, we can consider the following LP-relaxation of the problem of determining if there is a solution with the maximum load at most B. We propose the following linear programming for the MLkFL . For each facility $i \in \mathcal{F}$, define $\mathcal{S}(B; i) := \{C \subseteq \mathcal{D} : \sum_{j \in C} c(i, j) \leq B\}$ to be the set of all stars centered at i that induce load at most B at i. We will often refer to a star in $\mathcal{S}(B; i)$ as a configuration. (Note that $\mathcal{S}(B; i)$ contains $\emptyset$.) Our LP will be a *configuration-style LP*, where for every facility i and star $C \in \mathcal{S}(B; i)$, we have a variable denoting if star C is chosen for facility i. This yields the following natural feasibility LP.

$$(P)$$

$$\sum_{i \in \mathcal{F}} \sum_{C \in \mathcal{S}(B;i): j \in C} x(i, C) \geq 1 \qquad \forall j \in \mathcal{D} \tag{3.1}$$

$$\sum_{C \in \mathcal{S}(B;i)} x(i, C) \leq 1 \qquad \forall i \in \mathcal{F} \tag{3.2}$$

$$\sum_{i \in \mathcal{F}} \sum_{C \in \mathcal{S}(B;i)} x(i, C) \leq k \tag{3.3}$$

$$x \geq 0.$$

Constraint (3.1) ensures that each client belongs to some configuration, and constraints (3.2) and (3.3) ensure that each facility belongs to at most one configuration, and that there are at most k configurations. We show that there is an MLkFL instance on the line metric, where the smallest value B_{LP} for which (P) is feasible is smaller than the optimal value by an $\Omega(k/\log k)$ factor; thus, the "integrality gap" of (P) is $\Omega(k/\log k)$. Moreover, in this instance, the graph containing the (i, j) edges such that $c(i, j) \leq B_{\mathsf{LP}}$ is connected.

Theorem 3.6.1. *The integrality gap of* (P) *is* $\Omega(k/\log k)$ *even for line metrics.*

Proof. Assume for simplicity that k is odd. Consider the following simple MLkFL instance. We have $\mathcal{F} = \{f_1, g_1, f_2, g_2, \ldots, f_m, g_m\}$, where $2m = k+1$. These facilities are located on a line as shown in Figure 3.8, with the distance between any two consecutive nodes being $T/2$. There are $n = 2k$ clients colocated with each facility. Let F_i (respectively G_i) denote the set of clients located at f_i (respectively g_i) for $1 \le i \le m$.

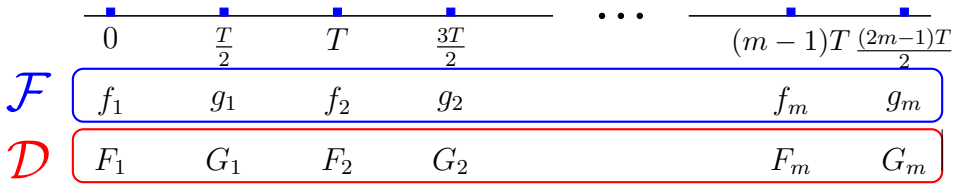


Figure 3.8: Example showing bad integrality gap for the configuration LP in line metrics.

There is a feasible solution to (P) with $B = T$. For all $i = 1, \ldots, m$, we set $x(f_i, F_i \cup \{j, j'\}) = \frac{k}{(k+1)\cdot\binom{n}{2}}$ for all $j, j' \in G_i$; note that all these configurations lie in $\mathcal{S}(T; f_i)$. Similarly, we set $x(g_i, G_i \cup \{j, j'\}) = \frac{k}{k+1\cdot\binom{n}{2}}$ for all $j, j' \in F_i$. It is easy to verify that x is a feasible solution. It is clear that constraints (3.2) and (3.3) hold since every facility belongs to exactly $\binom{n}{2}$ configurations. Consider a client $j \in F_i$. j is covered to an extent of $\frac{k}{k+1}$ by the $\binom{n}{2}$ configurations $\left\{F_i \cup \{k, \ell\}\right\}_{k,\ell \in G_i}$ in $\mathcal{S}(f_i; T)$ and to an extent of $\frac{1}{k+1}$ by the $n-1$ configurations $\left\{F_i \cup \{j, k\}\right\}_{k \in F_i : k \neq j}$. A symmetric argument applies to clients in some G_i set.

Finally, we show that any feasible solution for this instance must have the maximum load at least $T \cdot \frac{k}{2H_k}$, where $H_r := 1 + \frac{1}{2} + \ldots + \frac{1}{r}$ is the r-th harmonic number, which proves the theorem since $H_r = \Theta(\log r)$. In any feasible solution, there is some location v that does not have an open facility. For $i = 1, \ldots, k$, let n_i be the number of clients colocated at v that are assigned to a facility at a location that is i hops away from v; set $n_i = 0$ if there is no such location. Then, $\sum_{i=1}^{k} n_i = n$, and the maximum load L at a facility is at least $\max_{i=1,\ldots,k} \frac{n_i i T}{4}$ since there are at most two facilities that are i hops away from v, and one of them must have at least $\frac{n_i}{2}$ clients assigned to it. Thus, we have $n_i \le \frac{4L}{iT}$ for all $i = 1, \ldots k$, and so $n \le \frac{4L}{T} \cdot H_k$, or $L \ge \frac{nT}{4H_k}$. So integrality gap of (P) is at least $\frac{nT/4H_k}{T} = \frac{2k}{4H_k} = \Omega(k/\log k)$. □

3.7 An unbounded locality gap for the multi-swap local-search algorithm for MLkFL

A natural local-search heuristic for MLkFL is one where given a current set S of k facilities, we may swap out a facility in S and swap in a facility not in S. More generally, we may consider a p-swap heuristic where we swap out and swap in at most p facilities. Note that given a set of k facilities, one can find a good assignment of clients to facilities by solving an instance of the generalized assignment problem [75]. We keep performing such local moves as long as it improves the maximum load of the solution. One can come up with simple examples showing that the *locality gap* of the p-swap heuristic, which is the worst-case ratio between the maximum load at a local optimum and the (global) optimal value, can be arbitrarily large, even on line metrics.

Theorem 3.7.1. *The locality gap of the p-swap heuristic is unbounded, even on line metrics.*

Proof. Choose any $\epsilon < 1$. Consider $3k$ consecutive locations $s_1, j_1, o_1, s_2, j_2, o_2, \ldots, s_k, j_k, o_k$ located on a line with the $d(s_i, j_i) = 1,\ d(j_i, o_i) = \epsilon$ for all $i = 1, \ldots, k$, and $d(o_i, s_{i+1}) = 1 - \epsilon$ for all $i = 1, \ldots, k-1$ (See Figure 3.9). The facility set is $\mathcal{F} = S \cup O$, where $S = \{s_1, \ldots, s_k\}$ and $O = \{o_1, \ldots, o_k\}$, and the client set is $\mathcal{D} = \{j_1, \ldots, j_k\}$.

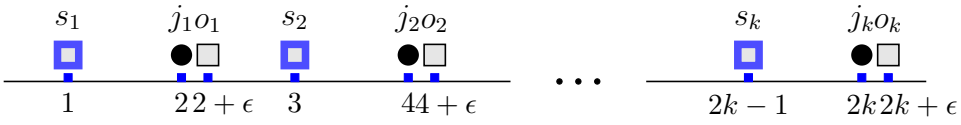


Figure 3.9: Example showing bad locality gap for a simple local search based on multiple swaps.

We claim that the solution S, which has the maximum load of 1, is a local optimum for the p-swap heuristic, for any $p < k$. Consider a p-swap move where we swap out $s_{i_1}, \ldots, s_{i_p}$ and swap in $o_{\ell_1}, \ldots, o_{\ell_p}$. We claim that this move does not decrease the maximum load, and hence is not an improving move. If it were, then the load of every facility in $F = S \setminus \{s_{i_1}, \ldots, s_{i_k}\} \cup \{o_{\ell_1}, \ldots, o_{\ell_k}\}$ must be strictly less than 1. But then none of the facilities in $F \cap S$ may be assigned any clients; thus, no facility in S serves any client. Since $p < k$, there is some i such that o_i is not swapped in. Then, j_i is not assigned to s_i or o_i and hence has connection cost larger than 1, which contradicts the assumption that the maximum load is less than 1.

Thus, S is a local optimum, whereas the global optimum is to open the facilities in O and assign each j_i to o_i incurring a maximum load of ϵ. □

In some sense, the rather simplistic nature of the above example exemplifies the difficulties in applying local search to min-max problems.

Chapter 4

Clustering problems with lower bounds and outliers

4.1 Introduction

Clustering is a practical and well-studied problem in Computer Science that arises in various applications in different areas such as data mining, machine learning, and bioinformatics. The main goal in clustering problems is to partition a set of points into k clusters, for some given k, such that some objective function is minimized. Formally, we consider the following abstraction of clustering problems. We are given a set $\mathcal{D}$ of data points and a set $\mathcal{F}$ of candidate centers (where data points are aggregated) located in a common metric space $\{c(i,j)\}_{i,j\in\mathcal{F}\cup\mathcal{D}}$; the distance between two data points j and j' measures the dissimilarity between j and j'. A cluster is a collection of data points that are assigned to some center in $\mathcal{F}$. For each cluster, the *radius* of the cluster is defined as the distance from the center to the furthest data point in the cluster. We consider two natural clustering objective functions, namely, *minimizing the maximum radius of a cluster* and *minimizing the sum of cluster radii*.

For these two objective functions, we consider clustering problems with some side constraints that arise in the modeling of various applications. More precisely, we consider clustering problems with *(non-uniform) lower bound requirements* on the cluster sizes, and we allow the flexibility of not clustering all points, i.e., some points may be designated as *outliers*, and left unclustered. We use the term k-supplier to denote the objective where we want to open k clusters and minimize the maximum cluster radius (the term k-center is used whenever $\mathcal{F} = \mathcal{D}$). We use LB$k$Sup and LB$k$SupO to denote the lower-bounded k-supplier problem without outliers and with outliers, respectively. Analogous problems to LBkSup and LBkSupO problems when $\mathcal{F} = \mathcal{D}$,

are LBkCent and LBkCentO, respectively. We use the notation LBkSR to denote the clustering problem with a lower bound requirement where we want to open k clusters to minimize the sum of radii; the term LBkSRO is used when outliers are allowed. More formal definitions of these problems appear in Section 4.2.

Various applications motivate these problems. One motivation for considering lower bounds comes from an *anonymity* perspective. Suppose we want to publish data points obtained from a test run on the patients of a hospital while preserving the identity of patients. In order to achieve data privacy, [73] proposed an anonymity problem where we perturb some of (the attributes of) the data points and then cluster them so that every cluster has at least L identical perturbed data points, thus making it difficult to identify a specific individual's data from the clustering. Aggrawal et al. [3, 2] observed that this anonymization problem can be abstracted as a lower-bounded clustering problem where the clustering objective captures the cost of perturbing data. Another motivation comes from a facility location perspective and is inspired by a model in which it is infeasible or unprofitable to provide a service from a facility unless we know that there are at least a certain number of clients served by this facility (see, e.g., [65]); an example is the *lower-bounded facility location* (LBFL) problem.

The motivation for considering outliers is that there could be data points that are quite dissimilar from the rest of the data, which can disproportionately degrade the cost of any clustering that is required to cluster all data points. Instead, allowing for outliers allows one to ignore such points and focus on data points of our interest.

4.1.1 Summary of Results

We obtain the *first* results for clustering problems with *non-uniform lower bounds and outliers*. We develop various techniques for tackling these problems using which we obtain *constant factor approximation guarantees* for LBkSRO and LBkSupO. Note that we need to ensure that none of the three types of *hard* constraints involved here—at most k clusters, non-uniform lower bounds, and at most m outliers—are violated, which is somewhat challenging.

For the k-supplier objective (Section 4.3), we obtain an approximation factor of 5 for LBkSupO (Theorem 4.3.2), and 3 for LBkSup (Theorem 4.3.1). These are the *first* approximation results for the k-supplier problem with non-uniform lower bounds. Previously, [2] obtained approximation factors of 4 and 2 respectively for LBkCentO and LBkCent for the special case of *uniform* lower bounds (recall that LBkCentO and LBkCent are special cases of LBkSupO and LBkSup when $\mathcal{F} = \mathcal{D}$, respectively). Complementing our approximation bounds, we prove a factor-3 hardness of approximation for LBkSup (Theorem 4.3.3), which shows that our approximation factor of 3 is optimal for LBkSup. We also show (Section 4.8) that LBkSupO and LBkCentO are equivalent

in terms of approximation, so we have a factor-3 hardness of approximation for LBkCentO and LBkSupO, as well.

For the min-sum-of-radii problem, we obtain an approximation factor of 12.365 for LBkSRO (Theorem 4.6.1,Section 4.6), which improves to 3.83 for the non-outlier version LBkSR (Theorem 4.5.2, Section 4.5). These also constitute the *first* approximation results for the min-sum-of-radii objective when we consider: (a) lower bounds (even uniform bounds) but no outliers (LBkSR); and (b) outliers but no lower bounds. Previously, an $O(1)$-approximation was known only in the setting where there are *no lower bounds and no outliers* [22].

The results presented in this chapter are part of a joint work with Chaitanya Swamy and it was published in 2016 [7].

4.1.2 Related Work

The only prior work on clustering problems to incorporate both lower bounds *and* outliers is by Aggarwal et al. [2]. They obtain approximation ratios of 4 and 2 respectively for LBkCentO and LBkCent with *uniform* lower bounds. They also prove factor-2 approximation hardness result for LBkCent with uniform lower bounds via a reduction from 3-SAT problem which complements their result.

Other related problems either consider lower bounds or outliers but not both. Charikar et al. [20] consider clustering problems with outliers such as uncapacitated FL, k-supplier and k-median problems. They devise constant factor approximations for the first two problems, and a bicriteria approximation for the k-median problem with outliers. They also proved a factor-3 approximation hardness result for k-supplier with outliers. This nicely complements our factor-3 hardness result for k-supplier with lower bounds but no outliers. The result for k-median with outliers was improved by Chen [23] who presented the first (and only) true approximation via a sophisticated combination of the primal-dual algorithm for k-median and local search that yields a large (unspecified) $O(1)$-approximation. Some of the challenges encountered in devising an approximation algorithm for this problem are similar in spirit to the challenges we encounter in devising an algorithm for LBkSRO. Another closely related problem is the *capacitated k-supplier with outliers* problem wherein instead of a lower bound requirement on the size of a cluster we have an upper bound on the size of a cluster. This problem was studied by Cygan and Kociumaka [27] and they devised a 25-approximation algorithm. Some of the ideas for our algorithm for LBkSup are inspired by their ideas.

Clustering problems with lower bounds have received some attention but are not so well understood. As it was mentioned LBkCent was studied by [2] and subsequently by Ene et al.[31] in

Euclidean spaces. The problem has been considered in the facility location setting as well under the name *lower-bounded facility location* (LBFL) [55, 41], wherein we seek to open (any number of) facilities (which have lower bounds) and assign each client j to an open facility $\sigma(j)$ so as to minimize $\sum_{j\in\mathcal{D}} c(\sigma(j), j)$. Svitkina [78] obtained the first true approximation for LBFL, achieving an $O(1)$-approximation; the $O(1)$-factor was subsequently improved by [6]. Both results apply to LBFL with uniform lower bounds, and can be adapted to yield $O(1)$-approximations to the k-median variant (where we may open at most k facilities).

We now discuss work related to our clustering objectives that does not consider lower bounds or outliers. Doddi et al. [30] introduced the min-sum-of-diameters problem (kSD), which is closely related to the min-sum-of-radii problem (kSR), and they showed that the kSD-cost is at least the kSR-cost and it is at most twice the kSR-cost. The kSD problem is better understood in terms of hardness results and it is *NP*-hard to obtain a polynomial time algorithm with approximation ratio better than 2 for kSD problem even when the metric is the shortest path metric of an unweighted graph [30]. However, kSR is only known to be *NP*-hard to approximate in general metrics, and its complexity for shortest-path metrics of unweighted graphs is not yet settled, with only a quasipolynomial time (exact) algorithm known for this case [36]. On the positive side, Charikar and Panigrahi [22] devised the first (and currently the best) $O(1)$-approximation algorithms for these problems, obtaining approximation ratios of 3.53 and 7.06 for kSR and kSD, respectively. Various other results are known for specific metric spaces and when $\mathcal{F} = \mathcal{D}$, such as Euclidean spaces [37, 17] and metrics with bounded aspect ratios [36, 14].

The k-supplier and k-center (i.e., k-supplier with $\mathcal{F} = \mathcal{D}$) objectives have a rich history of study. Hochbaum and Shmoys [47, 48] obtained optimal approximation ratios of 3 and 2 for these problems respectively. Capacitated versions of k-center and k-supplier have also been studied: [57] devised a 6-approximation for uniform capacities, [26] obtained the first $O(1)$-approximation for non-uniform capacities, and this $O(1)$-factor was improved to 9 in [8].

4.2 Problem Definition and Preliminaries

Various clustering problems can be considered in the framework where each facility has a lower bound requirement on the number of clients assigned to it (if opened), and a bounded number of points may be designated as *outliers* and left unclustered. More precisely, we consider two clustering problems with different objective functions with the same input and feasible solutions. In these problems, we are given a facility-set $\mathcal{F}$, client-set $\mathcal{D}$ located in a metric space $\{c(i,j)\}_{i,j\in\mathcal{F}\cup\mathcal{D}}$, lower bounds $\{L_i\}$ for each facility $i \in \mathcal{F}$, a non-negative integer k specifying the number of allowed clusters, and a non-negative integer m specifying the maximum number of allowed outliers. A feasible solution is a pair $\big(F \subseteq \mathcal{F}, \sigma : \mathcal{D} \mapsto F \cup \{\mathsf{out}\}\big)$, where $\sigma(j) \in F$

indicates that j is assigned to facility $\sigma(j)$, and $\sigma(j) = \mathsf{out}$ designates j as an outlier, such that $|F| \leq k$, $|\sigma^{-1}(i)| \geq L_i$ for all $i \in F$, and $|\sigma^{-1}(\mathsf{out})| \leq m$. That is, the lower bound requirements are met for the facilities in F, and at most m points are designated as outliers. Let the *radius* of facility i, denoted by r_i be defined as the distance between facility i and the furthest client assigned to i, i.e., $r_i := \max_{j \in \sigma^{-1}(i)} c(i, j)$. We define two different problems based on the objective function we are interested in:

- The *lower-bounded k-supplier with outliers* (LBkSupO) problem is the min max-radius problem where we want to find a feasible solution (F, σ) to minimize the maximum radius among the opened facilities in F, i.e, minimize

$$\max_{i \in F} r_i = \max_{i \in F} \max_{j \in \sigma^{-1}(i)} c(i, j).$$

  The special case where $m = 0$ is called the *lower-bounded k-supplier* (LBkSup) problem, and the setting where $\mathcal{D} = \mathcal{F}$ is often called the *k-center* version.

- The *lower-bounded min-sum-of-radii with outliers* (LBkSRO) problem is the min-sum version where we want to find a feasible solution (F, σ) to minimize the sum of cluster radii of clusters, i.e., minimize

$$\sum_{i \in F} r_i = \sum_{i \in F} \max_{j \in \sigma^{-1}(i)} c(i, j).$$

  The special case where $m = 0$ is called the *lower-bounded min-sum-of-radii* (LBkSR) problem.

One piece of notation used in both problems is $B(i, r)$ which denotes the ball of clients centered at i with radius r, i.e., $B(i, r) = \{j : c(i, j) \leq r\}$.

4.3 Minimizing the maximum radius with lower bounds and outliers

As defined above, in the *lower-bounded k-supplier with outliers* (LBkSupO) problem, we seek a feasible solution, i.e., a pair ($F \subseteq \mathcal{F}, \sigma : \mathcal{D} \mapsto F \cup \{\mathsf{out}\}$) with $|F| \leq k$, $|\sigma^{-1}(i)| \geq L_i$ for all $i \in F$, $|\sigma^{-1}(\mathsf{out})| \leq m$, that minimizes the maximum radius r_i of a facility in F (where $r_i = \max_{j \in \sigma^{-1}(i)} c(i, j)$). The two special cases are *lower-bounded k-supplier* (LBkSup) problem in which $m = 0$, and the *lower-bounded k-center with outliers* (LBkCentO) in which $\mathcal{D} = \mathcal{F}$.

Although LBkCentO is a special case of LBkSupO, it can be shown that if there exists an α-approximation for LBkCentO then there exists an α-approximation for LBkSupO as well (see Section 4.8).

Let τ^* denote the optimal value; note that there are only polynomially many choices for τ^* as $\tau^* \in \{c(i,j) : i \in \mathcal{F}, j \in \mathcal{D}\}$. As is common in the study of min-max problems, we reduce the problem to a "graphical" instance, where given some value τ, we try to find a solution of cost $O(\tau)$ or deduce that $\tau^* > \tau$. We construct a bipartite unweighted graph $G_\tau = \big(V_\tau, E_\tau\big)$ where the vertex set V_τ consists of the bipartitions $\mathcal{D} \cup \mathcal{F}_\tau$, where $\mathcal{F}_\tau = \{i \in \mathcal{F} : |B(i,\tau)| \geq L_i\}$, i.e., $\mathcal{F}_\tau$ contains facility $i \in \mathcal{F}$ if there are at least L_i clients with distance at most τ from i. Edge set E_τ is a collection of edges ij between client j and facility $i \in \mathcal{F}_\tau$ that are at distance at most τ from each other, i.e., $E_\tau = \{ij : c(i,j) \leq \tau, i \in \mathcal{F}_\tau, j \in \mathcal{D}\}$. Let $dist_\tau(i,j)$ denote the shortest-path distance in G_τ between i and j, so $c(i,j) \leq dist_\tau(i,j) \cdot \tau$ by triangle inequality. We say that an assignment $\sigma : \mathcal{D} \mapsto \mathcal{F}_\tau \cup \{\mathsf{out}\}$ is a *distance-α assignment* if $dist_\tau(j, \sigma(j)) \leq \alpha$ for every client j with $\sigma(j) \neq \mathsf{out}$. We call such an assignment feasible, if it yields a feasible LBkSupO solution, and we say that G_τ is feasible if it admits a feasible distance-1 assignment. It is not hard to see that given $F \subseteq \mathcal{F}_\tau$, the problem of finding a feasible distance-α-assignment $\sigma : \mathcal{D} \mapsto F \cup \{\mathsf{out}\}$ in G_τ (if one exists) can be solved by creating a network-flow instance with lower bounds and capacities. [1]

Observe that an optimal solution yields a feasible distance-1 assignment in G_{τ^*}. We devise an algorithm that for every τ, either finds a feasible distance-α assignment in G_τ for some constant α, or detects that G_τ is not feasible. This immediately yields an α-approximation algorithm since the smallest τ for which the algorithm returns a feasible LBkSupO solution must be at most τ^*. We obtain Theorems 4.3.1 and 4.3.2 via this template.

Theorem 4.3.1. *There is a 3-approximation algorithm for* LBkSup.

Theorem 4.3.2. *There is a 5-approximation algorithm for* LBkSupO.

Before introducing our algorithms for these problems, let us prove a result on the hardness of LBkSupO. The following hardness result complements the above, and shows that our approximation factor for LBkSup is tight.

Theorem 4.3.3. *It is NP-hard to approximate* LBkSup *within a factor better than 3, unless $P = NP$.*

[1] We add source s, sink t, and another node o. We have edges (s,j) for all $j \in \mathcal{D}$ with lower bound and capacity 1, edges (i,t) for all $i \in F$ with lower bounds L_i and infinite capacity, and edge (o,t) with lower bound 0 and capacity m. We also have edges (j,i) if $dist(i,j) \leq \alpha$, and edges (j,o); these edges have lower bound 0 and infinite capacity. The source s has demand $-|\mathcal{D}|$, the sink t has demand $|\mathcal{D}|$, and every other vertex has demand 0. A feasible distance-α assignment in G_τ exists if and only if there exists a feasible circulation satisfying the demands of all the vertices.

Proof. The result is shown via a reduction from the set cover problem. Suppose we have a set cover instance with set $\mathcal{U} = [n]$ of elements and collection $\mathcal{S} = \cup_{q=1}^{n'}\{S_q\}$ of subsets of $\mathcal{U}$, and we want to know if there exists k subsets of $\mathcal{U}$ in $\mathcal{S}$ that cover all elements of $\mathcal{U}$. Let $j_1, j_2, \cdots, j_n$ represent the elements and $i_1, i_2, \cdots, i_{n'}$ represent subsets of $\mathcal{U}$ in $\mathcal{S}$. Construct an LBkSup instance $\mathcal{I}$ with client set $\mathcal{D} = \cup_{p=1}^{n}\{j_p\}$, facility set $\mathcal{F} = \cup_{q=1}^{n'}\{i_q\}$, define $c(j_p, i_q)$ for $j_p \in \mathcal{D}, i_q \in \mathcal{F}$ to be 1 if $p \in S_q$, 3 otherwise, and let $L_i = 1$ for each $i \in \mathcal{F}$. Suppose there exists a collection F of k subsets in $\mathcal{S}$ that cover all elements. First, remove any set i in F, if i does not cover an element that is not covered by $F \setminus i$. Let $\sigma : \mathcal{D} \to F$ be defined for element j to be some set in F that covers j. Since each set i in F covers at least one element that is not covered by $F \setminus i$, $|\sigma^{-1}(i)| \geq 1$, so (F, σ) is a feasible solution to $\mathcal{I}$ with radius 1. Conversely, if (F, σ) is a feasible solution to $\mathcal{I}$ with radius 1, then it is easy to see that the sets corresponding to F yield k subsets that cover $\mathcal{U}$. Thus, there exist k subsets that cover $\mathcal{U}$ if and only if $\text{OPT}(\mathcal{I})$ is 1. It is also easy to see that if $\text{OPT}(\mathcal{I}) > 1$, then it is at least 3. Therefore, it is *NP*-hard to approximate LBkSup within a factor better than 3 as otherwise the algorithm can be used to answer the decision problem. □

4.3.1 Finding a distance-3 assignment for LBkSup.

In this section, we present our algorithm for finding a solution with a distance-3 assignment for the LBkSup problem. Consider the graph G_{τ^*}. Note that there exists an optimal center among the neighbors of each client in G_{τ^*}. Moreover, two clients at distance at least 3 are served by two distinct centers. These insights motivate the following algorithm.

Let $N(v)$ denote the neighbors of vertex v in the given graph G_τ. Find a maximal subset Γ of clients such that every pair of distinct clients in Γ are at distance at least 3 from each other. If $|\Gamma| > k$ or there exists a client j with $N(j) = \emptyset$, then return G_τ is not feasible. (i.e., there is no feasible distance-1 assignment in G_τ). For each $j \in \Gamma$, let i_j denote the center in $N(j)$ with minimum lower bound. If there exists a feasible distance-3 assignment σ of clients to $F = \bigcup_{j\in\Gamma}\{i_j\}$, return σ (this can be checked by solving a simple network flow problem), otherwise return G_τ is not feasible. The following lemma yields Theorem 4.3.1.

Lemma 4.3.4. *If G_τ is feasible, then the above algorithm finds a feasible distance-3 assignment in G_τ.*

Proof. Let $\sigma^* : \mathcal{D} \mapsto F^*$ be a feasible distance-1 assignment in G_τ. This means that $\left|\sigma^{*^{-1}}(i)\right| \geq L_i$ for each $i \in F^*$. Moreover, since each client is assigned to a facility at distance at most 1 from it, every client has a non-empty neighbor set. Since each client in Γ has to be served by a

distinct center in F^*, $|\Gamma| \leq |F^*| \leq k$. For each client $j \in \Gamma$, let $i_j^* = \sigma^*(j)$. Note that $i_j^* \in N(j)$, so $L_{i_j} \leq L_{i_j^*}$ by the choice of i_j, and every client in ${\sigma^*}^{-1}(i_j^*)$ is at distance at most 3 from i_j.

We show that there is a feasible distance-3 assignment $\sigma : \mathcal{D} \mapsto F$. For each $j \in \Gamma$, we assign all clients in ${\sigma^*}^{-1}(i_j^*)$ to i_j. As argued above this satisfies the lower bound of i_j by choice of i_j. For any unassigned client j, let $j' \in \Gamma$ be a client at distance at most 2 from j (which must exist by maximality of Γ); we assign j to $i_{j'}$. □

The above lemma shows that the algorithm returns a distance-3 assignment for all $\tau \geq \tau^*$ (since G_τ is feasible for all $\tau \geq \tau^*$). Therefore, as mentioned before this yields a 3-approximation as the smallest τ for which the algorithm returns a distance-3 solution must be smaller than or equal to τ^*, hence the algorithm return a solution which assigns each clients to a facility at distance at most 3τ which is at most $3\tau^*$.

4.3.2 Finding a distance-5 assignment for LBkSupO.

In this section, we present our algorithm that given an input graph $G_\tau = (\mathcal{D} \cup \mathcal{F}_\tau, E_\tau)$, returns an LB$k$SupO solution with a distance-5 assignment or returns that G_τ is not feasible. The main idea here is to find a set $F \subseteq \mathcal{F}_\tau$ of at most k centers that are close to the centers in $F^* \subseteq \mathcal{F}_\tau$ where (F^*, σ^*) is a feasible LBkSupO solution and $\sigma^* : \mathcal{D} \mapsto F^* \cup \{\mathsf{out}\}$ is a distance-1 assignment in G_τ. The non-outlier clients of (F^*, σ^*) are close to F, so there are at least $|\mathcal{D}| - m$ clients close to F. If centers in F do not share a neighbor in G_τ, then clients in $N(i)$ can be assigned to i for each $i \in F$ to satisfy the lower bounds (recall that for each $i \in \mathcal{F}_\tau$, $|N(i)| \geq L_i$). We cannot check if F satisfies the above properties, but using an idea similar to that in [27], we will find a sequence of facility sets such that at least one of these sets will have the desired properties when G_τ is feasible.

Definition 4.3.5. *Given the bipartite graph G_τ, a set $F \subseteq \mathcal{F}$ is called a* skeleton *if it satisfies the following properties.*

(a) (Separation property) For $i, i' \in F$, $i \neq i'$, we have $dist_\tau(i, i') \geq 6$;

(b) There exists a feasible LBkSupO *solution (F^*, σ^*) with distance-1 assignment $\sigma^* : \mathcal{D} \mapsto F^* \cup \{\mathsf{out}\}$ in G_τ such that*

- *(Covering property) For all $i^* \in F^*$, $dist_\tau(i^*, F) \leq 4$, where $dist_\tau(i^*, F)$ is defined as $\min_{i \in F} dist_\tau(i^*, i)$.*
- *(Injection property) There exists $f : F \mapsto F^*$ such that $dist_\tau(i, f(i)) \leq 2$ for all $i \in F$.*

If F satisfies the separation and injection properties, it is called a pre-skeleton.

Note that if $F \subseteq \mathcal{F}_\tau$ is a skeleton (or pre-skeleton), then G_τ is feasible. Suppose $F \subseteq \mathcal{F}_\tau$ is a skeleton and satisfies the properties with respect to a feasible distance-1 assignment (F^*, σ^*). The separation property ensures that the neighbor sets of any two locations $i, i' \in F$ are disjoint. The covering property ensures that F^* is at distance at most 4 from F, so there are at least $|\mathcal{D}|-m$ clients at distance at most 5 from F. Finally, the injection and separation properties together ensure that $|F| \leq k$ since no two locations in F can be mapped to the same location in F^*. Thus, if F is a skeleton, then we can obtain a feasible distance-5 assignment $\sigma : \mathcal{D} \mapsto F \cup \{\mathsf{out}\}$.

Lemma 4.3.6. *Let F be a pre-skeleton in G_τ. Define $U = \{i \in \mathcal{F}_\tau : dist_\tau(i, F) \geq 6\}$ and let $i = \arg\max_{i' \in U} |N(i')|$. Then, either F is a skeleton, or $F \cup \{i\}$ is a pre-skeleton.*

Proof. Suppose F is not a skeleton and $F \cup \{i\}$ is not a pre-skeleton. Let $\sigma^* : \mathcal{D} \mapsto F^* \cup \{\mathsf{out}\}$ be a feasible distance-1 assignment in G_τ such that F satisfies the injection property with respect to (F^*, σ^*). Let $f : F \mapsto F^*$ be the mapping given by the injection property. Since $F \cup \{i\}$ is not a pre-skeleton and $dist_\tau(i, F) \geq 6$, this implies that $dist_\tau(i, F^*) > 2$, and hence, $dist_\tau(i, F^*) \geq 4$ as G_τ is bipartite. This means that all clients in $N(i)$ are outliers in (F^*, σ^*). Moreover, since F is not a skeleton, there exists a center $i^* \in F^*$ with $dist_\tau(i^*, F) > 4$, and so $dist(i^*, F) \geq 6$ (see Figure 4.1). Therefore, $i^* \in U$. By the choice of i, we know that $|N(i)| \geq |N(i^*)|$. Now consider $F' = F^* \setminus \{i^*\} \cup \{i\}$, and define $\sigma' : \mathcal{D} \mapsto F' \cup \{\mathsf{out}\}$ as follows: $\sigma'(j) = \sigma^*(j)$ for all $j \notin N(i) \cup N(i^*)$, $\sigma'(j) = i$ for all $j \in N(i)$, and $\sigma'(j) = \mathsf{out}$ for all $j \in N(i^*)$. Note that the F' covers as many clients as F^*, and so $\sigma' : \mathcal{D} \mapsto F' \cup \{\mathsf{out}\}$ is another feasible distance-1 assignment. But this yields a contradiction since $F \cup \{i\}$ now satisfies the injection property with respect to (F', σ') as certified by the function $f' : F \cup \{i\} \to F'$ defined by $f'(s) = f(s)$ for $s \in F$, $f'(i) = i$.

□

If G_τ is feasible, then $\emptyset$ is a pre-skeleton. A skeleton can have size at most k. So using Lemma 4.3.6, we can find a sequence $\mathcal{F}'$ of at most $k+1$ subsets of $\mathcal{F}_\tau$ by starting with $\emptyset$ and repeatedly applying Lemma 4.3.6 until we either have a set of size k or the set U in Lemma 4.3.6 is empty. By Lemma 4.3.6, if G_τ is feasible then one of these sets must be a skeleton. So if for some $F \in \mathcal{F}'$, there exists a feasible distance-5 assignment $\sigma : \mathcal{D} \mapsto F \cup \{\mathsf{out}\}$, then, we return (F, σ). Otherwise we return that G_τ is not feasible.

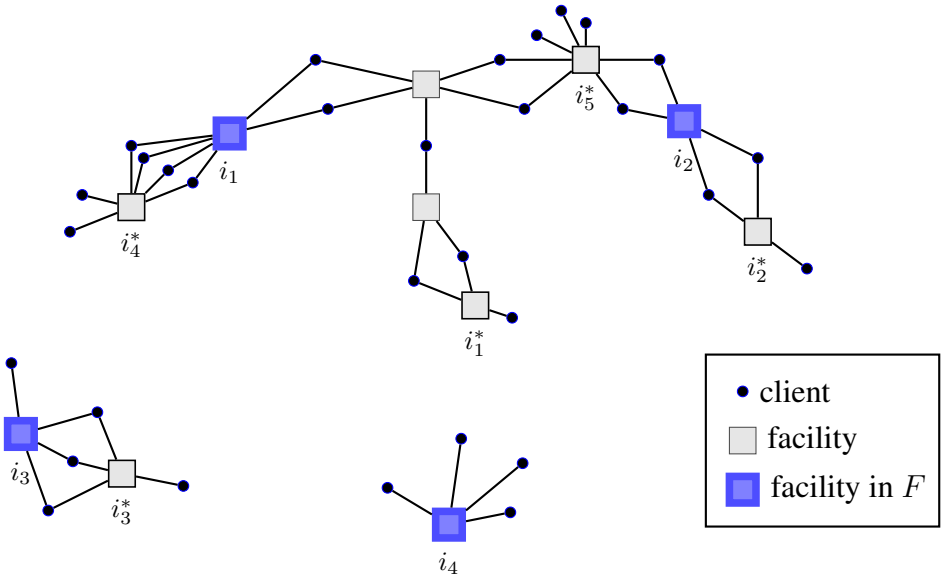


Figure 4.1: An example of pre-skeleton construction: $F = \{i_1, i_2, i_3, i_4\}$ and i_4 is the lastly added facility. $F \setminus \{i_4\}$ is a pre-skeleton with respect to $F^* = \{i_1^*, i_2^*, \cdots, i_5^*\}$ but F is not a skeleton as $dist_\tau(i_1^*, F) > 4$. F is a pre-skeleton with respect to an optimal solution $F^* \setminus \{i_1^*\} \cup \{i_4\}$.

4.4 Minimizing sum of radii with lower bounds and outliers

Recall that in the *lower-bounded min-sum-of-radii with outliers* (LBkSRO) problem, the input $\mathcal{I} = (\mathcal{F}, \mathcal{D}, \{c(i,j)\}_{i\in\mathcal{F}, j\in\mathcal{D}}, \{L_i\}_{i\in\mathcal{F}}, k, m)$ and the space of feasible solutions remain the same as in LBkSupO (i.e., $\big(F \subseteq \mathcal{F}, \sigma : \mathcal{D} \mapsto F \cup \{\mathsf{out}\}\big)$ is feasible if $|F| \le k$, $|\sigma^{-1}(i)| \ge L_i$ for all $i \in F$, and $|\sigma^{-1}(\mathsf{out})| \le m$.); our objective is to find a feasible solution (F, σ) that minimizes $cost(F, \sigma) := \sum_{i\in F} r_i$. A special case is when all clients have to be clustered and $m = 0$ (LBkSR problem).

It will be convenient to consider a relaxation of LBkSRO that we call the *k-ball-selection* (k-BS) problem, which focuses on selecting at most k balls centered at facilities of minimum total radius. Recall that $B(i,r)$ denotes the ball of clients centered at i with radius r. Let $\mathcal{L}_i := \{(i,r) : |B(i,r)| \ge L_i\}$ denote the collection of pairs (i,r) for a fixed facility $i \in \mathcal{F}$ where $B(i,r)$ has enough clients to satisfy the lower bound of i, and let $\mathcal{L} := \bigcup_{i\in\mathcal{F}} \mathcal{L}_i$ be the set of all allowed pairs. A feasible solution to k-BS is a set $F \subseteq \mathcal{L}$ of at most k balls (i.e., $|F| \le k$) that covers all but at most m clients, i.e., $\big|\mathcal{D} \setminus \bigcup_{(i,r)\in F} B(i,r)\big| \le m$. The goal in k-BS is to find a feasible solution F so that $cost(F) := \sum_{(i,r)\in F} r$ is minimized. When formulating the

LP-relaxation of the k-BS problem, we equivalently view $\mathcal{L}$ as containing at most $|\mathcal{F}| \cdot |\mathcal{D}|$ pairs of the form $(i, c(i,j))$ for some client j, which makes $\mathcal{L}$ finite.

It is easy to see that any LBkSRO solution yields a k-BS solution of no greater cost: let (F, σ) be a feasible solution of LBkSRO, define $F = \{(i, c(i, j_i)) : i \in F, j_i = \arg\max_{j \in \sigma^{-1}(i)} c(i,j)\}$ (note that $F \subseteq \mathcal{L}$ as for each $(i,r) \in F$, $\sigma^{-1}(i) \subseteq B(i,r)$ and $|\sigma^{-1}(i)| \geq L_i$). The key advantage of working with k-BS is that we do not explicitly consider the lower bounds (they are folded into the $\mathcal{L}_i$s) and we do not require the balls $B(i,r)$ for $(i,r) \in F$ to be disjoint. While a k-BS solution F need not directly translate to a feasible LBkSRO solution, one can show that it does yield a feasible LBkSRO solution of cost at most $2 \cdot cost(F)$. We prove a stronger version of this statement in Lemma 4.4.1. In the following two sections, we utilize this relaxation to devise the *first* constant-factor approximation algorithms for LBkSR and LBkSRO.

The first step of our algorithm for both problems involves "guessing" the t facilities in the optimal solution with the largest radii, and their radii, where $t \geq 1$ is some constant. This will help us in filtering the set of balls considered for the underlying k-BS problem (we only consider balls with radius bounded by the smallest radius among the guessed balls), which is needed in order to bound the cost of final solution.[2]We perform this guessing step by enumerating over all $O\big((|\mathcal{F}| + |\mathcal{D}|)^{2t}\big)$ choices $F^O = \{(i_1, r_1), \ldots, (i_t, r_t)\}$ of t (i,r) pairs from $\mathcal{L}$ (this can be done by guessing facility i along with the furthest client j assigned to it, and checking if $|B(i,r)| \geq L_i$). For each such selection, we set $\mathcal{D}' = \mathcal{D} \setminus \bigcup_{(i,r) \in F^O} B(i,r)$, $\mathcal{L}' = \{(i', r') \in \mathcal{L} : r' \leq \min_{(i,r) \in F^O} r\}$ and $k' = k - |F^O|$, and run our k-BS-algorithm on the modified k-BS-instance $(\mathcal{F}, \mathcal{D}', c, \mathcal{L}', k', m)$ to obtain a k-BS solution F. We translate $F \cup F^O$ to an LBkSRO solution, and return the best of these solutions. The following lemma, and the procedure described therein, is repeatedly used to bound the cost of translating $F \cup F^O$ to a feasible LBkSRO solution (see Algorithm 2 for an overview of the general algorithm). We call pairs $(i,r), (i',r') \in \mathcal{F} \times \mathbb{R}_{\geq 0}$ *non-intersecting*, if $c(i,i') > r + r'$, and *intersecting* otherwise. Note that $B(i,r) \cap B(i',r') = \emptyset$ if (i,r) and (i',r') are non-intersecting. For a set $P \subseteq \mathcal{F} \times \mathbb{R}_{\geq 0}$ of pairs, define $\mu(P) := \{i \in \mathcal{F} : \exists r \text{ s.t. } (i,r) \in P\}$.

Lemma 4.4.1. *Let $F^O \subseteq \mathcal{L}$, and $\mathcal{D}', \mathcal{L}', k'$ be as defined above. Let $F \subseteq \mathcal{L}$ be a k-BS solution for the k-BS instance $(\mathcal{F}, \mathcal{D}', c, \mathcal{L}', k', m)$. Suppose for each $i \in \mu(F)$, we have a radius $r'_i \leq \max_{r:(i,r) \in F} r$ such that the pairs in $U := \bigcup_{i \in \mu(F)} (i, r'_i)$ are non-intersecting and $U \subseteq \mathcal{L}$. Then there exists a feasible LBkSRO solution (F, σ) with $cost(F, \sigma) \leq cost(F) + \sum_{(i,r) \in F^O} 2r$.*

Proof. Pick a maximal subset $P \subseteq F^O$ to add to U such that all pairs in $U' = U \cup P$ are non-intersecting. For each $(i,r) \in F^O \setminus P$, define $\kappa(i,r)$ to be some intersecting pair $(i', r') \in U'$

[2]This is similar in spirit to the preprocessing for the k-*minimum spanning tree*($k-$MST) problem where we guess the furthest node (or $1/\epsilon$ furthest nodes) from the root that is covered by the optimal tree.

Algorithm 2 Algorithm for constructing feasible assignment LBkSRO solution (F, σ)

Input: An LBkSRO instance $\mathcal{I} = \big(\mathcal{F}, \mathcal{D}, \{c(i,j)\}, \{L_i\}, k, m\big)$, parameter $\epsilon > 0$.
Output: A feasible LBkSRO solution (F, σ).

$t = \min\big\{k, \lceil \frac{1}{\epsilon} \rceil\big\}$.
for each $F^O = \{(i_1, r_1), \ldots, (i_{t'}, r_t)\} \subseteq \mathcal{L}$ **do**
  $\mathcal{D}' \leftarrow \mathcal{D} \setminus \bigcup_{(i,r)\in F^O} B(i,r)$.
  $\mathcal{L}' \leftarrow \{(i,r) \in \mathcal{L} : r \le R^* = \min_{(i,r)\in F^O} r\}$.
  $k' \leftarrow k - t$.
  $\mathcal{I}' \leftarrow (\mathcal{F}, \mathcal{D}', c, \mathcal{L}', k', m)$.
  $(F, \{r'_i\}_{i\in\mu(F)}) \leftarrow k$-BS algorithm's output on instance $\mathcal{I}'$ satisfying Lemma 4.4.1.
  if LP relaxation of the k-BS instance is infeasible (P_3 or P_1 when $m = 0$) **then**
    Reject this guess, i.e., skip to the next guess.
  else
    $(F, \sigma) \leftarrow$ the output of the procedure in Lemma 4.4.1 with $(F, \{r'_i\}_{i\in\mu(F)})$.
return min-cost solution (F, σ) found in the for loop.

(see Figure 4.2). Define $F = \mu(U')$. Assign each client j to $\sigma(j) \in F$ as follows. If $j \in B(i,r)$ for some $(i,r) \in U'$, set $\sigma(j) = i$. Note that $U' \subseteq \mathcal{L}$ and sets in U' are non-intersecting, so this satisfies the lower bounds for all $i \in F$. Otherwise, if $j \in B(i,r)$ for some $(i,r) \in F$, set $\sigma(j) = i$. Otherwise, if $j \in B(i,r)$ for some $(i,r) \in F^O \setminus P$ and $(i',r') = \kappa(i,r)$, set $\sigma(j) = i'$. Any remaining unassigned client is not covered by the balls corresponding to pairs in $F \cup F^O$. There are at most m such clients (since F is a feasible solution for the k-BS instance), and we set $\sigma(j) = \mathsf{out}$ for each such client j. Thus (F, σ) is a feasible LBkSRO solution.

For any $i \in F$ and $j \in \sigma^{-1}(i)$ either $j \in B(i,r)$ for some $(i,r) \in F \cup U'$, or $j \in B(i',r')$ where $\kappa(i',r') = (i,r) \in U'$, in which case $c(i,j) \le r + 2r'$. So

$$cost(F, \sigma) \le cost(F) + \sum_{(i,r)\in F^O} 2r.$$

□

In the next two sections, we describe the algorithms for k-BS problem for non-outlier and outlier versions and analyze the resulting algorithms for LBkSR (Section 4.5) and LBkSRO (Section 4.6), respectively.

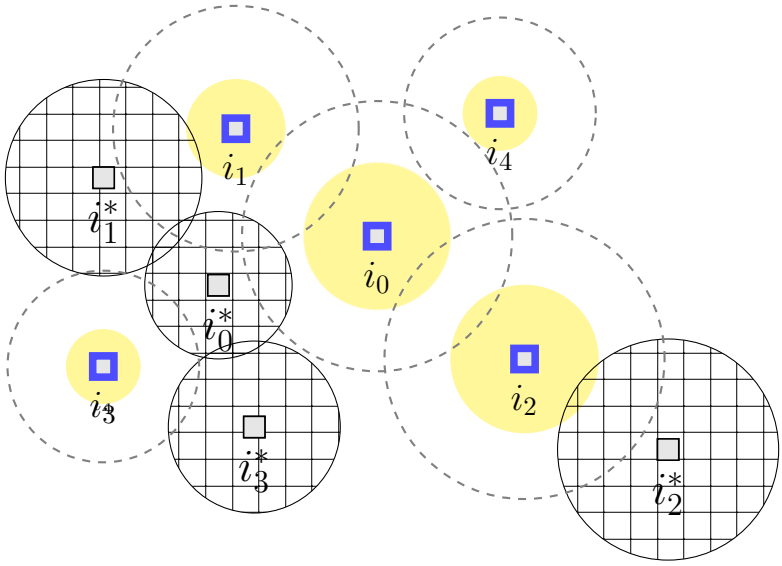


Figure 4.2: In this example $F = \{(i_l, r_l) : 0 \leq l \leq 4\}$, each ball in F is shown by a dashed-bordered circle, and each ball in U is shown by solid (yellow) circle. Balls in the set $F^O = \{(i_l^*, r_l^*) : 0 \leq l \leq 4\}$ are shown by grid-filled circles. In this example $P = \{(i_0^*, r_0^*)\}$, $\kappa(i_1^*, r_1^*) = \kappa(i_3^*, r_3^*) = (i_0^*, r_0^*)$, and $\kappa(i_2^*, r_2^*) = (i_2, r_2)$.

4.5 Approximation algorithm for LBkSR

We now present our algorithm for the non-outlier version, LBkSR, which will introduce many of the ideas underlying our algorithm for LBkSRO described in Section 4.6. The main results of this section are as follows.

Theorem 4.5.1. *There exists a* $(6.18 + \epsilon)$*-approximation algorithm for* LBkSR *problem for any* $\epsilon > 0$.

Theorem 4.5.2. *There exists a* $(3.83 + \epsilon)$*-approximation algorithm for* LBkSR *problem for any* $\epsilon > 0$.

As discussed before, we focus on a k-BS problem with client set $\mathcal{D}'$ and set $\mathcal{L}'$ of balls, and consider the k-BS problem of picking at most k', which is equal to $k - t$ where t is the number of guessed balls, pairs from $\mathcal{L}'$ whose corresponding balls cover $\mathcal{D}'$ incurring the minimum cost. We point out that our algorithm k-BSAlg in fact returns pairs from $\mathcal{L}$ (i.e., the superset of $\mathcal{L}'$ where we do not place any upper bounds on the radii) but this is compatible with Lemma 4.4.1.

We consider the following natural LP-relaxation (P_1) of this problem, and its dual (D_1). (As remarked earlier, we now consider $\mathcal{L}'$ to be a finite set of pairs of the form $(i, c(i, j))$ for some facility i and some client j.) We have an indicator variable $y_{i,r}$ for each $(i, r) \in \mathcal{L}'$ that indicates whether $B(i, r)$ is selected or not. We have to cover all clients with at most k balls, so the LP is

$$\begin{aligned} \min \quad & \sum_{(i,r)\in\mathcal{L}'} r \cdot y_{i,r} && (P_1)\\ \text{s.t.} \quad & \sum_{(i,r)\in\mathcal{L}':j\in B(i,r)} y_{i,r} \geq 1 \quad \forall j \in \mathcal{D}' \\ & \sum_{(i,r)\in\mathcal{L}'} y_{i,r} \leq k' && (4.1)\\ & y \geq 0. \end{aligned}$$

$$\begin{aligned} \max \quad & \sum_{j\in\mathcal{D}'} \alpha_j - k' \cdot z && (D_1)\\ \text{s.t.} \quad & \sum_{j\in B(i,r)\cap\mathcal{D}'} \alpha_j - z \leq r \quad \forall (i, r) \in \mathcal{L}' \\ & \alpha \geq 0 \\ & z \geq 0. \end{aligned}$$

In our algorithm for LBkSR problem, if (P_1) is infeasible then we discard the choice of t pairs and move to the next selection. In the remainder of this section, we assume (P_1) is feasible. Let OPT denote the common optimal value of (P_1) and (D_1). Our algorithm k-BSAlg follows the Jain-Vazirani (JV) template for k-median. As in the JV algorithm for k-median, we Lagrangify constraint (4.1) and consider the facility location version of the problem where we do not bound the number of pairs we may pick, but we incur a fixed cost z for each pair (i, r) that we pick (in addition to r). So the new LP along with its dual are as follows

$$\begin{aligned} \min \quad & \sum_{(i,r)\in\mathcal{L}'} (r + z) \cdot y_{i,r} && (P_2)\\ \text{s.t.} \quad & \sum_{(i,r)\in\mathcal{L}':j\in B(i,r)} y_{i,r} \geq 1 \quad \forall j \in \mathcal{D}' \\ & y \geq 0. \end{aligned}$$

$$\begin{aligned} \max \quad & \sum_{j\in\mathcal{D}'} \alpha_j && (D_2)\\ \text{s.t.} \quad & \sum_{j\in B(i,r)\cap\mathcal{D}'} \alpha_j \leq r + z \quad \forall (i, r) \in \mathcal{L}' && (4.2)\\ & \alpha \geq 0. \end{aligned}$$

In the following, we first present our primal-dual algorithm for facility location version of the k-BS problem (Section 4.5.1), then we present our algorithm k-BSAlg in Section 4.5.2 followed

by the analysis of the LBkSR algorithm in Section 4.5.3 that employs this k-BSAlg in line 6 of Algorithm 2. Finally in Section 4.5.4, we present ideas for improving approximation ratio of k-BSAlg which results in a better approximation ratio for the LBkSR problem.

While our approach is similar to the one in [22] for the min-sum-of-radii problem without lower bounds (although our combination step is notably simpler), an important distinction that arises is the following. In the absence of lower bounds, the ball-selection problem k-BS is equivalent to the min-sum-of-radii problem, but (as noted earlier) this is no longer the case when we have lower bounds since in k-BS problem, we do not insist that the balls we pick be disjoint. Consequently, moving from overlapping balls in a k-BS solution to an LBkSR solution incurs, in general, a factor-2 blowup in the cost (see Lemma 4.4.1). It is interesting that we are able to avoid this blowup and obtain an approximation factor that is quite close to the approximation factor (of 3.504) achieved in [22] for the min-sum-of-radii problem without lower bounds.

4.5.1 Primal Dual Algorithm

In this section, we present our primal-dual algorithm PDAlg for the facility location version of k-BS problem, which utilizes the LPs (P_2) and (D_2). The primal dual algorithm takes input $\mathcal{I} = (\mathcal{F}, \mathcal{D}', c, \mathcal{L}', k')$ (since $m = 0$ in this problem, we remove it from the input) and a fixed opening cost z, and outputs a solution $F \subseteq \mathcal{L}$, radius $\mathsf{rad}(i)$ for all $i \in \mu(F)$, and a dual solution α to (D_2). The algorithm consists of two main phases:

PD1 **Dual-ascent phase.** We start with a feasible dual solution $\alpha = 0$ ($\alpha_j = 0$ for all $j \in \mathcal{D}'$), the set of *active clients* initialized to $\mathcal{D}'$, and the set T of *tight pairs* initialized to $\emptyset$. We repeat the following until all clients become inactive: we raise the α_js of all active clients uniformly until constraint (4.2) becomes tight for some (i, r); we add (i, r) to T and mark all active clients in $B(i, r)$ as inactive.

PD2 **Pruning phase.** Let T_I be a maximal subset of non-intersecting pairs in T picked by a greedy algorithm that scans pairs in T in non-increasing order of radius (see Figure 4.3). Note that for each $i \in \mu(T_I)$, there is exactly one pair $(i, r) \in T_I$; we set $\mathsf{rad}(i) := r$ and set $r_i := \max\{c(i, j) : j \in B(i', r'),\ (i', r') \in T,\ r' \leq r,\ (i', r')$ intersects $(i, r)\}$. Note that r_i denotes the distance from i to the furthest client in a T-ball of radius at most r intersecting with $B(i, r)$. Let $F = \{(i, r_i)\}_{i \in \mu(T_I)}$. Return F, $\{\mathsf{rad}(i)\}_{i \in \mu(T_I)}$, and α.

Claim 4.5.3. *Solution F found by* PDAlg *covers all clients in $\mathcal{D}'$ and $F \subseteq \mathcal{L}$. Moreover, $\{(i, \mathsf{rad}(i))\}_{i \in \mu(F)} \subseteq \mathcal{L}'$, is a set of non-intersecting pairs, and $\mathsf{rad}(i) \leq r_i \leq 3\mathsf{rad}(i) \leq 3R^*$ for all $i \in \mu(F)$,*

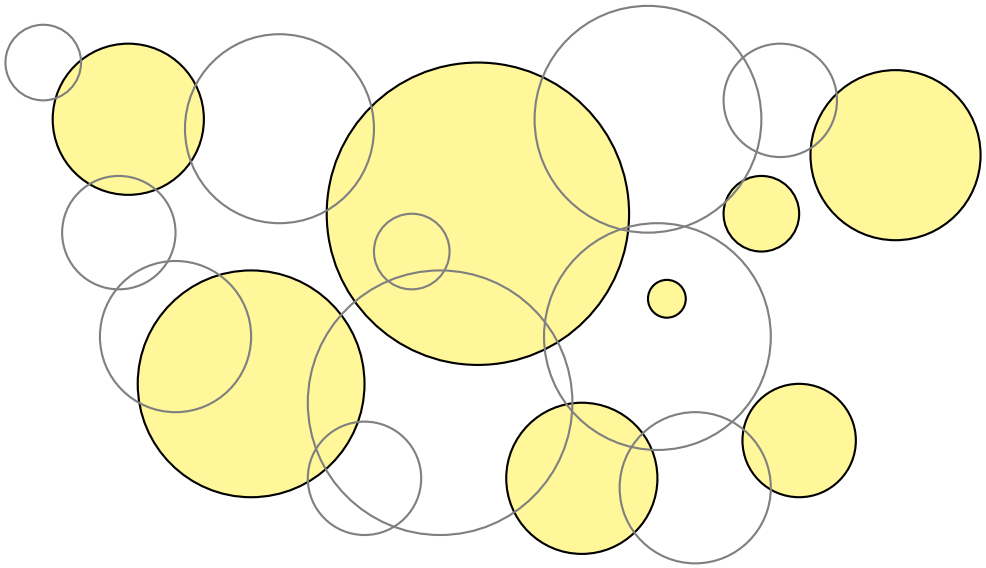


Figure 4.3: Example for the pruning phase: all the tight pairs in T are shown by circles and the tight pairs in T_I are shown by solid (yellow) circles.

Proof. We first show that all clients in $\mathcal{D}'$ are covered by F. The Dual-ascent phase terminates when all clients are inactive. Consider client $j \in \mathcal{D}'$, and let (i', r') denote the tight pair that causes j to become inactive. Then there must be a pair $(i, r) \in T_I$ that intersects (i', r') such that $r \geq r'$ (we could have $(i, r) = (i', r')$). Since by definition $r_i \geq c(i, j)$, $j \in B(i, r_i)$ and j is covered by F-balls.

Note that $\{(i, \mathsf{rad}(i))\}_{i \in \mu(F)} = T_I$, so all these pairs are tight and non-intersecting. Since all tight sets are subsets of $\mathcal{L}'$, $\{(i, \mathsf{rad}(i))\}_{i \in \mu(F)} \subseteq \mathcal{L}'$. Now consider $(i, r_i) \in F$. Note that $(i, \mathsf{rad}(i)) \in T_I$ and since $T_I \subseteq T$, $(i, \mathsf{rad}(i))$ is among the T-balls with radius at most $\mathsf{rad}(i)$ intersecting with $(i, \mathsf{rad}(i))$, so $r_i \geq \mathsf{rad}(i)$. In other words, each $(i, r) \in F$ is obtained by expanding the radius of the $(i, \mathsf{rad}(i))$-ball so that clients in intersecting balls of smaller radius are covered. Clearly, expanding the radius by $2\mathsf{rad}(i)$ is enough to cover any client in an intersecting ball with radius smaller than $\mathsf{rad}(i)$, so $r_i \leq 3\mathsf{rad}(i)$. Since $(i, \mathsf{rad}(i))$ is in $\mathcal{L}'$, $|B(i, \mathsf{rad}(i))| \geq L_i$ which means that (i, r) is in $\mathcal{L}$. Moreover, $\mathsf{rad}(i) \leq R^*$ for each $(i, \mathsf{rad}(i)) \in \mathcal{L}'$, so $3\mathsf{rad}(i) \leq 3R^*$.

□

Next, we prove that PDAlg is a *Lagrangian-multiplier-preserving* (LMP) 3-approximation algorithm : if F is the primal solution constructed, then $3\sum_j \alpha_j$ can pay for $cost(F) + 3|F|z$.

Theorem 4.5.4. *Suppose* $\mathsf{PDAlg}(\mathcal{F}, \mathcal{D}', c, \mathcal{L}', z)$ *returns* $(F, \{\mathsf{rad}(i)\}, \alpha)$ *where* (α, z) *is a feasible solution to* (P_1)*. Then*

$$cost(F) + 3|F|z \leq 3 \sum_{j \in \mathcal{D}'} \alpha_j \leq 3(OPT + k'z)$$

Moreover, If $|F| \geq k'$*, then* $cost(F) \leq 3 \cdot OPT$ *and if* $|F| > k'$*, then* $z \leq OPT$*.*

Proof. Consider a pair $(i, \mathsf{rad}(i)) \in T_I$, since this pair is tight, we have $\sum_{j \in B(i, \mathsf{rad}(i))} \alpha_j = \mathsf{rad}(i) + z$. Since all pairs in T_I are non-intersecting,

$$\sum_{j \in \mathcal{D}'} \alpha_j \geq \sum_{(i, \mathsf{rad}(i)) \in T_I} \mathsf{rad}(i) + |T_I| \cdot z$$

Since by Claim 4.5.3, $\mathsf{rad}(i) \geq \frac{1}{3} r_i$ for each $(i, r_i) \in F$ and $|T_I| = |F|$, we get the first inequality in the lemma. The last inequality in the lemma follows by weak duality and the fact that (α, z) is a feasible solution to (D_1).

For the second part, since $cost(F) + 3|F|z \leq 3OPT + 3k'z$, if $|F| \geq k'$, $cost(F) \leq 3OPT$. Moreover, $3(|F| - k')z \leq 3OPT$, so if $|F| > k'$, then $z \leq OPT$. □

Corollary 4.5.5. *Let* $(F, \{\mathsf{rad}(i)\}, \alpha) = \mathsf{PDAlg}(\mathcal{F}, \mathcal{D}', c, \mathcal{L}', z)$ *for* $z = 2k' c_{\max}$ *where* $c_{\max} = \max_{i \in \mathcal{F}, j \in \mathcal{D}} c(i, j)$*. Then* $|F| \leq k'$*.*

Proof. We prove this corollary by showing that $OPT \leq k' c_{\max}$, so using Theorem 4.5.4, $|F| \leq k'$. Since (P_1) is feasible (by assumption), all balls in $\mathcal{L}'$ have radius at most $c_{\max}$ and any feasible solution of (P_1) satisfies $\sum_{(i,r) \in \mathcal{L}'} y_{i,r} \leq k'$, the optimal solution of (P_1) has value at most $k' c_{\max}$. □

Corollary 4.5.6. *Let* $(F, \{\mathsf{rad}(i)\}, \alpha) = \mathsf{PDAlg}(\mathcal{F}, \mathcal{D}', c, \mathcal{L}', z)$ *for* $z = 0$*. If* $|F| \leq k'$*, then* F *is a feasible* k-BS *solution with cost at most* $3OPT$*.*

Proof. Since F covers all clients in $\mathcal{D}'$ (Claim 4.5.3), and $|F| \leq k'$, F is a feasible k-BS solution. Using Theorem 4.5.4, $cost(F) + 3|F|z \leq 3OPT + 3k'z$. Since $z = 0$, $cost(F) \leq 3OPT$. □

4.5.2 Algorithm k-BSAlg

In this section, we explain our algorithm k-BSAlg for obtaining a k-BS solution utilizing PDAlg. We follow similar framework as JV algorithm for k-median. We use PDAlg within a binary-search procedure for z to obtain two solutions F_1 and F_2 with $|F_1| > k' > |F_2|$ and then show how these two solutions can be combined to extract a feasible k-BS solution with cost at most $6.183OPT + O(R^*)$. This combination step is more involved than in k-median.

Step 1: Binary search for z.

We start with $z_1 = 0$ and $z_2 = 2k'c_{\max}$. Let $(F_p, \{\mathsf{rad}_p(i)\}, \alpha^p) \leftarrow \mathsf{PDAlg}(\mathcal{F}, \mathcal{D}', c, \mathcal{L}', z_p)$ for $p = 1, 2$, and let $k_p = |F_p|$. If $k_1 \leq k'$, we stop and return $\big(F_1, \{\mathsf{rad}_1(i)\}\big)$ as the output of k-BSAlg (Steps 2 and 3 below are skipped). In Corollary 4.5.5, we proved that that $k_2 \leq k'$. If $k_2 = k'$, we stop and return $\big(F_2, \{\mathsf{rad}_2(i)\}\big)$ as the output of k-BSAlg (Steps 2 and 3 are skipped).

Now if we have not returned any solution yet, it means that we have our initial F_1 and F_2 solutions with $|F_1| > k' > |F_2|$. We repeat the following until $z_2 - z_1 \leq \delta_z = \frac{\epsilon OPT}{3n}$, where $n = |\mathcal{F}| + |\mathcal{D}|$: Set $z = \frac{z_1+z_2}{2}$. Let $(F, \{\mathsf{rad}(i)\}, \alpha) \leftarrow \mathsf{PDAlg}(\mathcal{F}, \mathcal{D}', c, \mathcal{L}', z)$. If $|F| = k'$, we stop and return $\big(F, \{\mathsf{rad}(i)\}\big)$ as an output of our k-BSAlg (combination step is skipped), otherwise if $|F| > k'$, update $z_1 \leftarrow z$ and $(F_1, \mathsf{rad}_1, \alpha^1) \leftarrow (F, \mathsf{rad}, \alpha)$, else update $z_2 \leftarrow z$ and $(F_2, \mathsf{rad}_2, \alpha^2) \leftarrow (F, \mathsf{rad}, \alpha)$.

Step 2: Combining F_1 and F_2.

If in Step 1, no solution is returned then we perform this step to combine two solutions F_1 and F_2 found in Step 1. The main idea is to use F_2 as a guide to merge some F_1-pairs. We cluster F_1 pairs around F_2-pairs and setup a *covering-knapsack problem* whose solution determines for each F_2-pair (i, r), whether to "merge" the F_1-pairs clustered around (i, r) or select all these F_1-pairs.

Let $\pi : F_1 \mapsto F_2$ be any map such that (i', r') and $\pi(i', r')$ intersect $\forall (i', r') \in F_1$. This map exists since every $j \in \mathcal{D}'$ is covered by $B(i, r)$ for some $(i, r) \in F_2$, so each (i', r')-pair in F_1 intersects with all the F_2-balls covering clients in $B(i', r')$. Define star $\mathcal{S}_{i,r} = \pi^{-1}(i, r)$ for all $(i, r) \in F_2$ (see Fig. 4.4).

In order to decide how to merge F_1-pairs, we use the following LP with an indicator variable $x_{i,r}$ for $(i, r) \in F_2$ which takes value 1 if all pairs in $\mathcal{S}_{i,r}$ are to be merged and takes value 0 if they are not to be merged.

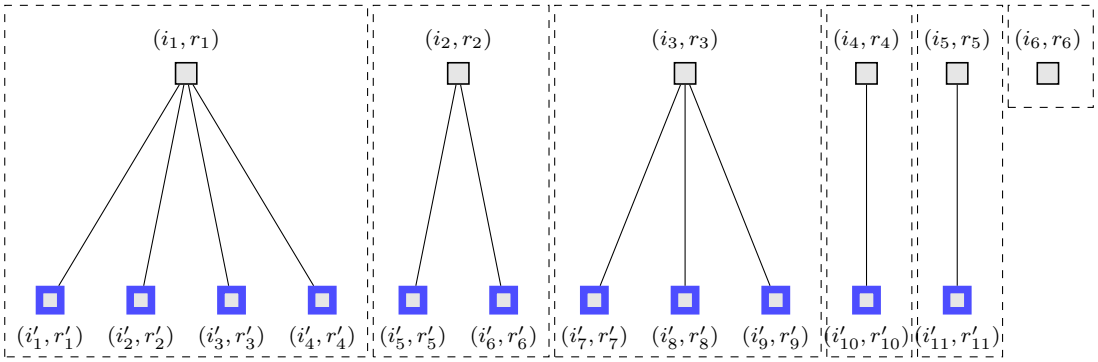


Figure 4.4: An example of stars formed by F_1 and F_2 where $F_1 = \{(i'_l, i'_l) : 1 \leq l \leq 11\}$ and $F_2 = \{(i_l, r_l) : 1 \leq l \leq 6\}$ depicted on the bottom row and the top row, respectively.

$$\min \sum_{(i,r)\in F_2} \left(x_{i,r}(2r + \textstyle\sum_{(i',r')\in\mathcal{S}_{i,r}} 2r') + (1 - x_{i,r}) \textstyle\sum_{(i',r')\in\mathcal{S}_{i,r}} r' \right) \qquad \text{(C-P)}$$

$$\text{s.t.} \quad \sum_{(i,r)\in F_2} \left(x_{i,r} + |\mathcal{S}_{i,r}|(1 - x_{i,r}) \right) \leq k \qquad \text{(SZ BND)}$$

$$0 \leq x_{i,r} \leq 1 \qquad \forall (i,r) \in F_2.$$

Let x^* be an extreme-point optimal solution to (C-P). We construct solution F' using x^* as follows. If $x^*_{i,r} = 0$, then we select all pairs in $\mathcal{S}_{i,r}$. Otherwise, if $\mathcal{S}_{i,r} \neq \emptyset$, we pick a pair in $(i', r') \in \mathcal{S}_{i,r}$, and include $(i', 2r + r' + \max_{(i'',r'')\in\mathcal{S}_{i,r}\setminus\{(i',r')\}} 2r'')$ in our solution. Notice that by expanding the radius of i' to $2r + r' + \max_{(i'',r'')\in\mathcal{S}_{i,r}\setminus\{(i',r')\}} 2r''$, we cover all the clients in $\bigcup_{(i'',r'')\in\mathcal{S}_{i,r}} B(i'', r'')$ (the coefficient of $x_{i,r}$ in LP objective function dominates this value and is used as an upper bound on this value).

Step 3: Returning a solution

In this step we return solution F' or F_2 depending on which has a smaller cost. So if $cost(F_2) \leq cost(F')$, return (F_2, rad_2), else return $\left(F', \{\mathsf{rad}_1(i)\}_{i\in\mu(F')}\right)$.

Analysis. Let $\left(F, \{\mathsf{rad}(i)\}\right) = k\text{-BSAlg}(\mathcal{F}, \mathcal{D}', c, \mathcal{L}', k')$. If k-BSAlg terminates before reaching combination step, then $cost(F) \leq 3 \cdot OPT$ due to Theorem 4.5.4 or Corollary 4.5.6,

so let us assume assume otherwise. So the binary-search procedure finds $(F_1, \mathsf{rad}_1, \alpha^1)$ and $(F_2, \mathsf{rad}_2, \alpha^2)$ which are the outputs of PDAlg for z_1 and z_2 respectively. Let $a, b \geq 0$ be such that $ak_1 + bk_2 = k'$, $a + b = 1$. Let $C_1 = cost(F_1)$ and $C_2 = cost(F_2)$. We first prove the following claim on the cost of a fractional solution obtained from linear combination of F_1 and F_2 solutions.

Claim 4.5.7. *We have $aC_1 + bC_2 \leq (3+\epsilon)OPT$, and moreover, $OPT_{\text{C-P}} \leq 2bC_2 + (1+b)C_1$.*

Proof. By Theorem 4.5.4, we have $C_1 + 3k_1z_1 \leq 3(OPT + k'z_1)$ and $C_2 + 3k_2z_2 \leq 3(OPT + k'z_2)$. Multiplying the first inequality by a, and the second by b, and adding and rearranging, we get

$$aC_1 + bC_2 \;\leq\; 3OPT + 3k'(az_1 + bz_2) - 3(ak_1z_1 + bk_2z_2)$$

Since $0 \leq z_2 - z_1 \leq \delta_z$, we get

$$\begin{aligned} aC_1 + bC_2 &\leq 3OPT + 3k'(az_2 + bz_2) - 3(ak_1(z_2 - \delta_z) + bk_2z_2) \\ \Rightarrow aC_1 + bC_2 &\leq 3OPT + 3k'(z_2) + 3ak_1\delta_z - 3z_2k' \end{aligned}$$

where the inequalities uses the fact that $a + b = 1$ and $ak_1 + bk_2 = k'$. Note that $k \leq n$ and $0 \leq a \leq 1$, so $3ak_1\delta_z \leq \epsilon OPT$ and the lemma follows.

In order to bound $OPT_{\text{C-P}}$, we present a feasible solution and use its cost to bound $OPT_{\text{C-P}}$. Consider $\bar{x}_{i,r} = b$ for all $(i,r) \in F_2$. $\bar{x}$ yields a feasible solution to (C-P) as $0 \leq b \leq 1$ and $\sum_{(i,r)\in F_2} b + |\mathcal{S}_{i,r}|(1-b) = b \cdot k_2 + a \cdot k_1 = k'$. The objective function evaluates to

$$\sum_{(i,r)\in F_2} b\cdot(2r + \sum_{r'\in\mathcal{S}_{i,r}} 2r') + (1-b) \sum_{(i',r')\in\mathcal{S}_{i,r}} r' = b \sum_{(i,r)\in F_2} 2r + \sum_{(i',r')\in F_1} b\cdot 2r' + (1-b)\cdot r' = 2bC_2 + (1+b)C_1$$

□

Lemma 4.5.8. *The solution (F', rad) found in the combination step consists of at most k balls and has the cost of at most $OPT_{\text{C-P}} + 15R^*$.*

Proof. In (C-P), there is only one constraint in addition to the bound constraints $0 \leq x_{i,r} \leq 1$, so the extreme-point optimal solution x^* has at most one fractional component, and if it has a fractional component, then $\sum_{(i,r)\in F_2} \left(x^*_{i,r} + |\mathcal{S}_{i,r}|(1 - x^*_{i,r})\right) = k'$. For any $(i,r) \in F_2$ with $x^*_{i,r} \in \{0,1\}$, the number of pairs we include is exactly $x^*_{i,r} + |\mathcal{S}_{i,r}|(1 - x^*_{i,r})$. If x^* has a fractional component $(i',r') \in F_2$, then $x^*_{i',r'} + |\mathcal{S}_{i',r'}|(1 - x^*_{i',r'})$ is a *positive* integer. Since we include at most one pair for (i',r'), this implies that $|F'| \leq k'$.

Now let us focus on the cost of F'. Again for any $(i,r) \in F_2$ with $x^*_{i,r} \in \{0,1\}$, the radius of the pair included is at most $x^*_{i,r}(2r + \sum_{(i',r')\in\mathcal{S}_{i,r}} 2r') + (1 - x^*_{i,r})\sum_{(i',r')\in\mathcal{S}_{i,r}} r'$ (the contribution to the objective function of (C-P) from the $x^*_{i,r}$ and $(1 - x^*_{i,r})$ terms). For a pair (i,r) with $0 < x^*_{i,r} < 1$, if $S_{i,r}$ is non-empty, we pick an arbitrary pair (i',r') in $\mathcal{S}_{i,r}$ and include a pair $(i', 2r + r' + \max_{(i'',r'')\in\mathcal{S}_{i,r}\setminus\{(i',r')\}} 2r'')$. Since all $(i,r) \in F_1 \cup F_2$ satisfy $r \le 3R^*$, the cost of the included pair in F' is at most $15R^*$. Therefore, $cost(F') \le OPT_{\text{C-P}} + 15R^*$.

□

Theorem 4.5.9. *k-BSAlg$(\mathcal{D}', \mathcal{L}', k')$ returns a feasible solution $\big(F, \{\mathsf{rad}(i)\}\big)$ with $cost(F) \le \big(6.183 + O(\epsilon)\big) \cdot OPT + O(R^*)$ where $\big\{(i, \mathsf{rad}(i))\big\}_{i\in\mu(F)} \subseteq \mathcal{L}'$ is a set of non-intersecting pairs.*

Proof. The radii $\{\mathsf{rad}(i)\}_{i\in\mu(F)}$ are simply the radii obtained from some execution of PDAlg, so $\big\{(i, \mathsf{rad}(i))\big\}_{i\in\mu(F)} \subseteq \mathcal{L}'$ and comprises non-intersecting pairs. If k-BSAlg terminates in the binary search step, then either $|F| = k'$, or $z_1 = 0$ and $|F| \le k'$; in both cases, we have $cost(F) \le 3 \cdot OPT$ (by Theorem 4.5.4 or Corollary 4.5.6). So in this cases we have a better bound (than what is claimed in the lemma) on $cost(F)$. If the algorithm does not return a solution in the binary search step, then it returns solution F' found in the combination step or solution F_2 whichever has a smaller cost. So when we terminate in step 3 (returning a solution), we return a solution F with $cost(F) \le \min\{C_2, 2bC_2 + (1+b)C_1 + 15R^*\}$. Claim 4.5.10 proves that $\min\{C_2, 2bC_2 + (1+b)C_1\} \le 2.0607(aC_1 + bC_2)$ for all $a, b \ge 0$ with $a + b = 1$. Combining this with Claim 4.5.7 yields the bound in the theorem. □

Claim 4.5.10. $\min\{C_2, 2bC_2 + (1+b)C_1\} \le (\frac{b+1}{3b^2-2b+1})(aC_1 + bC_2) \le 2.0607(aC_1 + bC_2)$ *for all $a, b \ge 0$ such that $a + b = 1$.*

Proof. Since the minimum is less than any convex combination,

$$\begin{aligned}
\min(C_2, 2bC_2 + bC_1 + C_1) &\le \frac{3b^2 - b}{3b^2 - 2b + 1} C_2 + \frac{1 - b}{3b^2 - 2b + 1}(2bC_2 + bC_1 + C_1) \\
&= \frac{(1-b)(1+b)}{3b^2 - 2b + 1} C_1 + \frac{b^2 + b}{3b^2 - 2b + 1} C_2 \\
&= \frac{b+1}{3b^2 - 2b + 1}((1-b)C_1 + bC_2)
\end{aligned}$$

Since $a = 1 - b$, the first inequality in the claim follows.

The expression $\frac{b+1}{3b^2-2b+1}$ is maximized at $b = -1 + \sqrt{2}$, and has value $1 + \frac{3}{2\sqrt{2}} \approx 2.0607$, which yields the second inequality in the claim. □

4.5.3 Analysis of LBkSR Algorithm employing k-BSAlg

Now that we have the bounds on the performance of k-BSAlg, we can analyze what Algorithm 2 will give us using this k-BSAlg (in line 6). We show that the right selection of F^O combined with Lemma 4.4.1 yields Theorem 4.5.1.

Proof of Theorem 4.5.1. It suffices to show that when the selection $F^O = \{(i_1, r_1), \ldots (i_t, r_t)\}$ in Algorithm 2 corresponds to the t facilities in an optimal solution with largest radii, we obtain the desired approximation bound. If $k \leq \frac{1}{\epsilon}$, then $t = k$, and so we obtain an optimal solution. Otherwise, $t \geq \frac{1}{\epsilon}$ and $R^* \leq \frac{O^*}{t} \leq \epsilon O^*$ and $OPT \leq O^* - \sum_{p=1}^{t} r_p$. Using Lemma 4.4.1, the cost of solution (F, σ) found is at most $cost(F) + \sum_{p=1}^{t} 2r_p$. By Theorem 4.5.9, $cost(F)$ is at most $\big(6.183 + O(\epsilon)\big) \cdot OPT + O(\epsilon O^*)$. Note that the remaining facilities in optimal solution (excluding the guessed facilities), yield a feasible solution to (P_1), so $OPT \leq O^* - \sum_{p=1}^{t} r_p$. Combining this we get that the cost of solution F is at most

$$\big(6.183 + O(\epsilon)\big) \cdot (O^* - \sum_{p=1}^{t} r_p) + O(\epsilon O^*) + \sum_{p=1}^{t} 2r_p \leq \big(6.183 + O(\epsilon)\big) \cdot O^*.$$

□

4.5.4 Improved Approximation Ratio for LBkSR

We now present a better way of combining solution F_1 and solution F_2 which leads to the improved approximation ratio stated in Theorem 4.5.2. We begin by representing the key insight that encourages us to modify our combination subroutine. This insight was originally observed by Charikar et. al [22], and we generalize it for LBkSRO problem in Lemma 4.6.9, called *continuity lemma*. The Lemma below is a corollary of Lemma 4.6.9 in Section 4.6.2, which proves the continuity property for the dual solution constructed for outlier problem. We defer the proofs of these lemmas to the end of this chapter.

Lemma 4.5.11. *Let* $(F_p, \ldots, \alpha^p) = \mathsf{PDAlg}(\mathcal{F}, \mathcal{D}', c, \mathcal{L}', z_p)$ *for* $p = 1, 2$, *where* $0 \leq z_2 - z_1 \leq \delta_z$. *Then,* $\|\alpha_j^1 - \alpha_j^2\|_\infty \leq 2^n \delta_z$. *Thus, if* (4.2) *is tight for some* $(i, r) \in \mathcal{L}'$ *in one execution, then* $\sum_{j \in B(i,r) \cap \mathcal{D}'} \alpha_j^p \geq r + z_p - 2^n \delta_z$ *for* $p = 1, 2$.

Proof. This follows from Lemma 4.6.9 since the dual-ascent process in PDAlg° when $m = 0$ is the same as the dual-ascent process in PDAlg. □

The above lemma essentially states that if a pair is tight in one execution of $z = z_1$ or $z = z_2$, then this pair is almost tight (might even be tight) in the other execution. We use this insight to improve our combination step by constructing stars based on whether $(i, \mathsf{rad}_1(i))$ intersects with $(i', \mathsf{rad}_2(i'))$ for pairs $(i, r) \in F_1$ and $(i', r') \in F_2$, rather than whether (i, r) intersects with (i', r'). Let Ik-BSAlg denote our modified combination step (replacing Step 2 in Section 4.5.2). A minor change in **Binary search for z** is that the process is continued until $z_2 - z_1 \leq \delta_z = \frac{\epsilon OPT}{3n2^n}$, so this adds up to n iterations. As before, we return solution F_2 or solution F' from the modified combination step Ik-BSAlg whichever has a smaller cost. Recall that we require that every pair $(i', r') \in F_1$ is in some star $\mathcal{S}_{i,r}$, but now a pair $(i', \mathsf{rad}_1(i'))$ may not intersect with any $(i, \mathsf{rad}_2(i))$ for $(i, r) \in F_2$, so first we modify solution F_2 to ensure this does not happen. Let $F = F_2$ and $T_F = T_{2,I}$. We consider pairs in F_1 one by one. For each pair $(i, r) \in F_1$, if $(i, \mathsf{rad}_1(i))$ does not intersect any pair in T_F, add $(i, \mathsf{rad}_1(i))$ to T_F and add (i, r) to F. We continue this process until all pairs in F_1 are scanned or $|F| = k'$. Now using Lemma 4.5.11, we get the following result.

Lemma 4.5.12. *If $|F| = k'$ after above process, then F is a feasible* kSR *solution with $cost(F) \leq (3 + \epsilon)OPT$.*

Proof. Using Lemma 4.5.11, each pair in T_F is almost tight, so for each $(i, \mathsf{rad}(i)) \in T_F$, we have $r + z_1 - 2^n\delta_z \leq \sum_{j \in B(i,\mathsf{rad}(i)) \cap \mathcal{D}'} \alpha_j^1$. Since all pairs in T_F are non-intersecting, $\sum_{(i,\mathsf{rad}(i)) \in T_F}(r + z_1) \leq \sum_{j \in \mathcal{D}} \alpha_j^1 + 2^n\delta_z \cdot |T_F|$. Since by construction of F, we have $F_2 \subseteq F$, all clients in $\mathcal{D}'$ are covered by balls corresponding to F-pairs. Now we know by PDAlg process that any (i, r) in F_2 or F_1 has $r \leq 3\mathsf{rad}(i)$, so the cost of solution F can be bounded as follows:

$$cost(F) = \sum_{(i,r) \in F} r \leq \sum_{(i,\mathsf{rad}(i)) \in T_F} 3\mathsf{rad}(i) \leq 3(\sum_{j \in \mathcal{D}'} \alpha_j^1 - |F|z_1 + 2^n\delta_z|F|).$$

Since $\sum_{j \in \mathcal{D}'} \alpha_j^1 - k'z_1 \leq OPT$ (by weak duality and feasibility of (α^1, z_1) to (D_1)) and $2^n\delta_z|F| \leq \epsilon OPT$ by choice of δ_z, we get $cost(F) \leq (3 + \epsilon)OPT$.

□

If preprocessing finds a solution F of size k', we just return this solution. Otherwise, $|F| < k'$ From this point, we use $F_2 := F$, $T_{2,I} = T_F$ and $\mathsf{rad}_2(i)$ is defined based on T_F. We combine solutions F_1 and F_2. We construct assignment $\pi : F_1 \to F_2$ similar to before with the small modification that $\pi(i, r) = (i', r')$ if $(i, \mathsf{rad}_1(i))$ intersects with $(i', \mathsf{rad}_2(i'))$. By our preprocessing step, π is well-defined. Let star $\mathcal{S}_{i,r} = \pi^{-1}(i, r)$ for each $(i, r) \in F_2$ (see Figure 4.5).

Again we use an LP with an indicator variable $x_{i,r}$. If $x_{i,r} = 0$, we select all pairs in $\mathcal{S}_{i,r}$. If $x_{i,r} > 0$ and $|\mathcal{S}_{i,r}| > 1$, we select a pair $(i', r') \in \mathcal{S}_{i,r}$ with the second largest radius, and include $(i', \mathsf{rad}_1(i') + 2\mathsf{rad}_2(i) + 4\max_{(i'',r'') \in \mathcal{S}_{i,r}} \mathsf{rad}_1(i''))$. If $x_{i,r} > 0$ and $|\mathcal{S}_{i,r}| = 1$, we select

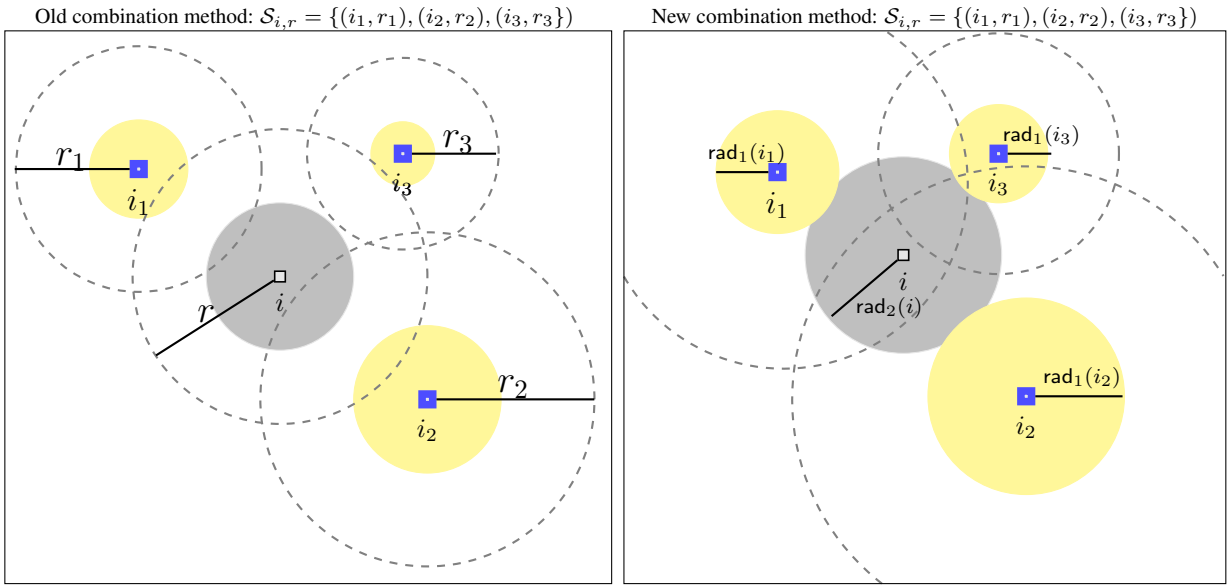


Figure 4.5: Comparison of old and new combination methods.

the $(i', r') \in \mathcal{S}_{i,r}$. Note by the given choices all clients in $\cup_{(i'',r'')\in\mathcal{S}_{i,r}} B(i'', r'')$ are covered and the radius of the chosen pair is dominated by $2\mathsf{rad}_2(i) + 4\sum_{(i',r')\in nstar_{i,r}} \mathsf{rad}_1(i')$ when $x_{i,r} > 0$. Since for each $(i, r) \in F_1$, $r \leq 3\mathsf{rad}_1(i)$, the new LP can be written as

$$\min \sum_{(i,r)\in F_2} \Big(x_{i,r}(2\mathsf{rad}_2(i) + \textstyle\sum_{(i',r')\in\mathcal{S}_{i,r}} 4\mathsf{rad}_1(i')) + (1 - x_{i,r}) \sum_{(i',r')\in\mathcal{S}_{i,r}} 3\mathsf{rad}_1(i') \Big) \qquad \text{(C-P')}$$

$$\text{s.t.} \quad \sum_{(i,r)\in F_2} \big(x_{i,r} + |\mathcal{S}_{i,r}|(1 - x_{i,r}) \big) \leq k, \qquad \text{(SZ BND)}$$

$$0 \leq x_{i,r} \leq 1 \qquad \forall (i, r) \in F_2.$$

Let x^* be an extreme point of (C-P'). Let F' be corresponding pairs obtained by the above process. Similar to before an optimal solution of this LP has at most one fractional component and the above rounding process chooses at most k' pairs.

Let us now analyze Ik-BSAlg. Let $C'_1 = \sum_{(i,r\in F_1)} \mathsf{rad}_1(i)$ and $C'_2 = \sum_{(i',r')\in F_2} \mathsf{rad}_2(i')$. We have claims analogous to Claim 4.5.7 and Claim 4.5.10 for our new analysis.

Claim 4.5.13. *We have* $aC'_1 + bC'_2 \leq (1 + \frac{\epsilon}{3})OPT$

Proof. Using Lemma 4.5.11,

$$\begin{aligned} aC'_1 + bC'_2 &\leq a \cdot (\sum_{j \in \mathcal{D}'} \alpha^1_j - k_1 z_1) + b \cdot (|F_2| 2^n \delta_z + \sum_{j \in \mathcal{D}'} \alpha^1_j - k_2 z_1) \\ &\leq \sum_{j \in \mathcal{D}'} (a\alpha^1_j + b\alpha^1_j) - (ak_1 + bk_2) \cdot z_1 + b \cdot k_2 2^n \delta_z \\ &\leq \sum_{j \in \mathcal{D}'} \alpha^1_j - k \cdot z + \frac{\epsilon}{3} OPT \leq (1 + \frac{\epsilon}{3}) OPT, \end{aligned}$$

where the first inequality uses Lemma 4.5.11, and the third inequality uses that $b \leq 1$, $k_2 \leq n$, and $\delta_z = \frac{\epsilon OPT}{3n2^n}$. □

Lemma 4.5.14. Ik-BSAlg$(\mathcal{F}, \mathcal{D}', c, \mathcal{L}', k')$ *returns a feasible solution* $(F, \{\mathsf{rad}(i)\})$ *with* $cost(F) \leq (3.83 + O(\epsilon))OPT + O(R^*)$ *where* $\{(i, \mathsf{rad}(i))\}_{i \in \mu(F)} \subseteq \mathcal{L}'$ *is a set of non-intersecting pairs.*

Proof. First note that if Ik-BSAlg returns solution $F = F_2$, then $\{\mathsf{rad}(i)\}$ correspond to $\{\mathsf{rad}_2(i)\}$. If Ik-BSAlg returns solution resulted from combination step, i.e., $F = F'$, then $\{\mathsf{rad}(i)\} \subseteq \{\mathsf{rad}_1(i)\}$ by construction of F' from optimal (C-P') solution. So in both cases $\{(i, \mathsf{rad}(i))\}$ corresponds to non-intersecting pairs in $\mathcal{L}'$.

Let us first bound the cost of solution F' obtained in the combination step. We claim the cost of the (possible) pair corresponding to a fractional component of x^* is at most $7R^*$. First note that $\mathsf{rad}_p(i) \leq R^*$ for each $(i, r) \in F_p$ for $p \in \{1, 2\}$. Now if $0 < x^*_{i,r} < 1$ and $|\mathcal{S}_{i,r}| > 1$, the included pair has radius $\mathsf{rad}_1(i') + 2\mathsf{rad}_2(i) + 4\mathsf{rad}_1(i'') \leq 7R^*$ where (i'', r'') and (i', r') are the pairs with the largest and the second largest radius in $\mathcal{S}_{i,r}$, respectively. if $0 < x^*_{i,r} < 1$ and $\mathcal{S}_{i,r} = \{(i', r')\}$, the included pair has radius $r' \leq 3\mathsf{rad}_1(i') \leq 7R^*$. Since x^* has at most one fractional component, cost of solution F' is at most $OPT_{\text{C-P'}} + 7R^*$. Also, $OPT_{\text{C-P'}} \leq 2bC'_2 + (4b + 3a)C'_1 = 2bC'_2 + (3 + b)C'_1$, since setting $x_{i,r} = b$ for all $(i, r) \in F_2$ yields a feasible solution to (C-P') of this cost. So the combination step returns solution F' with $cost(F') \leq 2bC'_2 + (b + 3)C'_1 + 7R^*$.

The algorithm returns one of solutions F' or F_2 depending on which has smaller cost. So the cost of solution F return by the algorithm is at most $\min(3C'_2, 2bC'_2 + (b + 3)C'_1) + 7R^*$ which is at most $3.83(aC'_1 + bC'_2) + 7R^*$ for all $a, b \geq 0$ and $a + b = 1$ by the following Claim 4.5.15. Using Claim 4.5.13, the cost of solution F is at most $(3.83 + O(\epsilon))OPT + O(R^*)$. □

Claim 4.5.15. $\min\{3C'_2, 2bC'_2 + (3 + b)C'_1\} \leq \frac{3(b+3)}{3b^2 - 2b + 3}(aC'_1 + bC'_2) \leq 3.83(aC'_1 + bC'_2)$ *for all* $a, b \geq 0$ *such that* $a + b = 1$.

Proof. Since the minimum is less than any convex combination,

$$\begin{aligned}\min(3C'_2, 2bC'_2 + bC'_1 + 3C'_1) &\leq \frac{3b^2+b}{3b^2-2b+3}(3C'_2) + \frac{-3b+3}{3b^2-2b+3}(2bC'_2 + bC'_1 + 3C'_1) \\ &= \frac{3(1-b)(b+3)}{3b^2-2b+3}(C'_1) + \frac{3b(b+3)}{3b^2-2b+3}C'_2 \\ &= \frac{3(b+3)}{3b^2-2b+3}((1-b)C'_1 + bC'_2) \leq \frac{3(b+3)}{3b^2-2b+3}OPT\end{aligned}$$

Since $a = 1 - b$, the first inequality in the claim follows. The expression $\frac{3(b+3)}{3b^2-2b+3}$ is maximized at $b = -3+2\sqrt{3}$, and has value $\frac{3}{8}(5+3\sqrt{3}) \approx 3.8235$, which yields the second inequality in the claim. □

We now have all the ingredients needed for proving our main result for the approximation of the LBkSR problem.

Proof of Theorem 4.5.2. Again when $F^O = \{(i_1, r_1), \ldots (i_t, r_t)\}$ in Algorithm 2 corresponds to the t facilities in an optimal solution with largest radii, we obtain the desired approximation bound. In this case, we either have $t = k$ and F^O is then an optimal solution, or $tgeq\frac{1}{\epsilon}$ and $R^* \leq \frac{O^*}{t} \leq \epsilon O^*$ and $OPT \leq O^* - \sum_{p=1}^{t} r_p$. Combining Lemma 4.5.14 and Lemma 4.4.1 then yields the theorem. □

4.6 Approximation algorithm for LBkSRO

In this section, we present our algorithm for the LBkSRO problem and its analysis. We prove the following result.

Theorem 4.6.1. *There exists a* $\big(12.365 + O(\epsilon)\big)$*-approximation algorithm for* LBkSRO *that runs in time* $n^{O(1/\epsilon)}$ *for any* $\epsilon > 0$.

The algorithm is built on the ideas presented in Section 4.5 and follows a similar line of attack. The presence of outliers introduces various complications and requires us to slightly modify the primal-dual algorithm and the binary search process. More significantly, it also leads to major modifications in the combination step. In the following, we first give an overview of the algorithm and necessary modification needed to be made to primal-dual and binary search step. Then we present two different combination steps employed by our algorithm. Finally, we present the analysis of the algorithm.

The high level approach of the algorithm is similar to the one in Section 4.5. We again "guess" the t (i,r) pairs F^O corresponding to the facilities with largest radii in an optimal solution, and consider the modified k-BS instance $(\mathcal{F}, \mathcal{D}', c, \mathcal{L}', k', m)$ (where $\mathcal{D}', \mathcal{L}', k'$ are defined as before). We design a primal-dual algorithm for the Lagrangian relaxation of the k-BS problem where we are allowed to pick any number of pairs from $\mathcal{L}'$ (leaving at most m uncovered clients) incurring a fixed cost of z for each pair picked. Then this algorithm is utilized to obtain two solutions F_1 and F_2, and finally, we combine these solutions to extract a low-cost solution.

We consider the following LP-relaxation of the k-BS problem and its dual (analogous to (P_1) and (D_1)). We use an indicator variable $y_{i,r}$ for each $(i,r) \in \mathcal{L}'$ that indicates whether the ball $B(i,r)$ is selected ($y_{i,r} = 1$) or not ($y_{i,r} = 0$). We also use an indicator variable w_j for each client j that indicates whether client j is an outlier ($w_j = 1$) or not ($w_j = 0$).

$$\begin{aligned}
\min \quad & \sum_{(i,r)\in\mathcal{L}'} r \cdot y_{i,r} && (P_3)\\
\text{s.t.} \quad & \sum_{(i,r)\in\mathcal{L}': j\in B(i,r)} y_{i,r} + w_j \geq 1 && \forall j \in \mathcal{D}'\\
& \sum_{(i,r)\in\mathcal{L}'} y_{i,r} \leq k' && (4.3)\\
& \sum_{j\in\mathcal{D}'} w_j \leq m\\
& y, w \geq 0.
\end{aligned}$$

$$\begin{aligned}
\max \quad & \sum_{j\in\mathcal{D}'} \alpha_j - k' \cdot z - m \cdot \gamma && (D_3)\\
\text{s.t.} \quad & \sum_{j\in B(i,r)\cap\mathcal{D}'} \alpha_j - z \leq r && \forall (i,r) \in \mathcal{L}'\\
& \alpha_j \leq \gamma && \forall j \in \mathcal{D}'\\
& \alpha \geq 0\\
& z, \gamma \geq 0.
\end{aligned}$$

If (P_3) is infeasible, then we reject this guess. So in the remainder of this section, we assume that (P_3) is feasible. Let OPTdenote the optimal value of (P_3). As in JV algorithm (and as before), we Lagrangify constraint (4.3) and consider the facility location version of the problem. So the new LP with its dual are:

$$\min \quad \sum_{(i,r)\in\mathcal{L}'} (r+z)\cdot y_{i,r} \qquad (P_4)$$

$$\text{s.t.} \quad \sum_{(i,r)\in\mathcal{L}':j\in B(i,r)} y_{i,r} + w_j \geq 1 \qquad \forall j \in \mathcal{D}'$$

$$\sum_{j\in\mathcal{D}'} w_j \leq m$$

$$y, w \geq 0.$$

$$\max \quad \sum_{j\in\mathcal{D}'} \alpha_j - m\cdot\gamma \qquad (D_4)$$

$$\text{s.t.} \quad \sum_{j\in B(i,r)\cap\mathcal{D}'} \alpha_j \leq r+z \qquad \forall (i,r)\in\mathcal{L}'$$

$$\alpha_j \leq \gamma \qquad \forall j\in\mathcal{D}'$$

$$\alpha, z, \gamma \geq 0.$$

As before we first use a primal-dual algorithm to find a feasible solution to (P_4) along with a feasible dual solution to (D_4). The natural modification of the earlier primal-dual algorithm PDAlg is to now stop the dual-ascent process when the number of active clients is at most m and set $\gamma = \max_{j\in\mathcal{D}'} \alpha_j$. This introduces the significant complication that one may not be able to completely pay for the $(r+z)$-cost of the non-intersecting tight pairs selected in the pruning phase by the dual objective value $\sum_{j\in\mathcal{D}'} \alpha_j - m\cdot\gamma$, since clients with $\alpha_j = \gamma$ may be needed to pay for the $r+z$-cost of the last tight pair $f = (i_f, r_f)$ but their contribution gets canceled by the $-m\cdot\gamma$ term. This issue affects us at various levels. First, we no longer obtain an LMP approximation for the unconstrained problem since we have to account for the $(r+z)$-cost of f separately. Second, unlike Claim 4.5.7, given solutions F_1 and F_2 obtained via binary search for $z_1, z_2 \approx z_1$ respectively with $|F_2| \leq k' \leq |F_1|$, we now only obtain a fractional k-BS solution of cost $O(OPT + z_1)$. While one can modify the covering-knapsack-LP based procedure of k-BSAlg to combine F_1, F_2, this only yields a good solution when $z_1 = O(OPT)$. The chief technical difficulty is that z_1 may however be much larger than OPT. We design a second combination procedure that is guaranteed to return a good solution when $z_1 = \Omega(OPT)$. This requires establishing certain structural properties for F_1 and F_2, using which we argue that one can find a good solution in the neighborhood of F_1 and F_2.

We now detail the changes to the primal-dual algorithm and k-BSAlg in Section 4.5, and analyze them to prove that our LBkSRO algorithm has an approximation guarantee of 12.365 (see Theorem 4.6.1 in Section 4.6.3).

Modified primal-dual algorithm PDAlg$^o(\mathcal{F}, \mathcal{D}', c, \mathcal{L}', z)$. This is quite similar to PDAlg (and we again return pairs from $\mathcal{L}$). We stop the dual-ascent process when there are at most m active clients. We set $\gamma = \max_{j\in\mathcal{D}'} \alpha_j$. Let $f = (i_f, r_f)$ be the last tight pair added to the tight-pair

set T, and $B_f = B(i_f, r_f)$. For a set P of (i, r) pairs, define $\mathsf{uncov}(P) := \mathcal{D}' \setminus \bigcup_{(i,r)\in P} B(i, r)$. Note that $|\mathsf{uncov}(T \setminus f)| > m \geq |\mathsf{uncov}(T)|$. Let OUT be a set of exactly m clients such that $\mathsf{uncov}(T) \subseteq \mathsf{OUT} \subseteq \mathsf{uncov}(T \setminus f)$. Note that $\alpha_j = \gamma$ for all $j \in \mathsf{OUT}$.

The pruning phase is similar to before, but we only use f if necessary. Let T_I be a maximal subset of non-intersecting pairs picked by greedily scanning pairs in $T \setminus f$ in non-increasing order of radius. For $i \in \mu(T_I)$, set $\mathsf{rad}(i)$ to be the unique r such that $(i, r) \in T_I$, and let r_i be the smallest radius ρ such that $B(i, \rho) \supseteq B(i', r')$ for every $(i', r') \in T \setminus f$ such that $r' \leq \mathsf{rad}(i)$ and (i', r') intersects $(i, \mathsf{rad}(i))$. Let $F' = \{(i, r_i)\}_{i\in\mu(T_I)}$. If $\mathsf{uncov}(F') \leq m$, set $F = F'$. If $\mathsf{uncov}(F') > m$ and there exists $i \in \mu(F')$ such that $c(i, i_f) \leq 2R^*$, then increase r_i so that $B(i, r_i) \supseteq B_f$ and let F be the set obtained from F' after this extension. Otherwise, set $F = F \cup f$ and $r_{i_f} = \mathsf{rad}(i_f) = r_f$. We return $(F, f, \mathsf{OUT}, \{\mathsf{rad}(i)\}_{i\in\mu(F)}, \alpha, \gamma)$.

Claim 4.6.2. *Solution F found by* PDAlg° *covers at least $|\mathcal{D}'| - m$ clients and $F \subseteq \mathcal{L}$. Moreover, $\big\{(i, \mathsf{rad}(i))\big\}_{i\in\mu(F)} \subseteq \mathcal{L}'$, is a set of non-intersecting pairs, and $\mathsf{rad}(i) \leq r_i \leq 3R^*$ for each $i \in \mu(F)$. Moreover, there is at most one $i \in \mu(F)$ for which $r_i > 3\mathsf{rad}(i)$.*

Proof. Let $F' = \{(i, r'_i)\}_{i\in\mu(T_I)}$ be the set of pairs obtained from the set T_I in the pruning phase. By the same argument as in the proof of Claim 4.5.3, we have $\mathsf{rad}(i) \leq r'_i \leq 3\mathsf{rad}(i) \leq 3R^*$ for all $i \in \mu(T_I)$, and $\mathsf{uncov}(F') \subseteq \mathsf{uncov}(T \setminus f)$. If we return $F = F'$, then $|\mathsf{uncov}(F)| \leq m$ by definition. If $\mathsf{uncov}(F') > m$ and there exists $i \in \mu(F')$ such that $c(i, i_f) \leq 2R^*$, we increase the radius of $(i, r) \in F'$ to include $B(i_f, r_f)$, so then we have $r_i \leq \max\{r'_i, 3R^*\} \leq 3R^*$ and $\mathsf{uncov}(F) \subseteq \mathsf{uncov}(T)$, so $|\mathsf{uncov}(F)| \leq m$. If $f \in F$, then we again have $\mathsf{uncov}(F) \subseteq \mathsf{uncov}(T)$. In all cases, F is feasible, and we have $\mathsf{rad}(i) \leq r_i \leq 3R^*$ for all $i \in \mu(F)$, and $r_i \leq 3\mathsf{rad}(i)$ for all but at most one $i \in \mu(F)$.

Notice that $\big\{(i, \mathsf{rad}(i))\big\}_{i\in\mu(F)}$ is T_I if $f \notin F$, and $T_I + f$ otherwise. In the latter case, we know that $c(i, i_f) > 2R^*$ for all $i \in \mu(T_I)$, so f does not intersect $(i, \mathsf{rad}(i))$ for any $i \in \mu(T_I)$. Thus, all pairs in $\big\{(i, \mathsf{rad}(i))\big\}_{i\in\mu(F)}$ are non-intersecting. □

Next, we prove a bound on the cost of solution returned by PDAlg°.

Theorem 4.6.3. *Suppose* $\mathsf{PDAlg}^\circ(\mathcal{F}, \mathcal{D}', c, \mathcal{L}', z)$ *returns* $(F, \{\mathsf{rad}(i)\}, \alpha, \gamma)$ *where* (α, z, γ) *is a feasible solution to* (P_3). *Then*

$$cost(F \setminus f) + 3|F \setminus f|z - 3R^* \leq 3(\sum_{j\in\mathcal{D}'} \alpha_j - m\gamma) \leq 3(OPT + k'z)$$

If $|F \setminus f| \geq k'$ then $cost(F) \leq 3OPT + 4R^*$, and if $|F \setminus f| > k'$ then $z \leq OPT$.

Proof. The argument in proof of Claim 4.6.2 shows that $cost(F \setminus f) \leq \sum_{i \in \mu(T_I)} 3 \cdot \mathsf{rad}(i) + 3R^*$. All pairs in T_I are tight and non-intersecting and $|F \setminus f| = |T_I|$. Also, $\mathsf{OUT} \subseteq \mathsf{uncov}(T \setminus f) \subseteq \mathsf{uncov}(T_I)$. (Recall that $|\mathsf{OUT}| = m$ and $\alpha_j = \gamma$ for all $j \in \mathsf{OUT}$.) So

$$cost(F \setminus f) + 3|F \setminus f|z - 3R^* \leq \sum_{i \in \mu(T_I)} (3 \cdot \mathsf{rad}(i) + 3z) = \sum_{\substack{i \in \mu(T_I) \\ j \in B(i, \mathsf{rad}(i)) \cap \mathcal{D}'}} 3\alpha_j \tag{4.4}$$

$$\leq 3\Big(\sum_{j \in \mathcal{D}'} \alpha_j - \sum_{j \in \mathsf{OUT}} \alpha_j\Big) = 3\Big(\sum_{j \in \mathcal{D}'} \alpha_j - m\gamma\Big) \leq 3(OPT + k'z).$$

The last inequality follows since (α, γ, z) is a feasible solution to (D_3).

If $|F \setminus f| \geq k'$ then since $r_f \leq R^*$ then $cost(F) \leq 3OPT + 4R^* + 3z(k' - |F \setminus f|)$ which is at most $3OPT + 4R^*$. Inequality (4.4) implies that , $\sum_{i \in \mu(T_I)} (\mathsf{rad}(i) + z) \leq OPT + k'z$ and since $|T_I| = |F \setminus f|$, we have $z \leq OPT$ if $|F \setminus f| > k'$. □

Modified algorithm k-BSAlg°$(\mathcal{F}, \mathcal{D}', c, \mathcal{L}', k', \epsilon)$. As before, first, we use binary search to find solutions F_1 and F_2 with $|F_1| > k'$ and $F_2 \leq k'$, and then extract a low-cost solution from combining these two solutions. The binary search is essentially the same with the modification that we start with $z_1 = 0$ and $z_2 = 2nk'c_{\max}$; we argue in Claim 4.6.4 that for this z_2, PDAlg° returns at most k' pairs. We stop when $z_2 - z_1 \leq \delta_z := \frac{\epsilon OPT}{3n2^n}$. Note that We *do not stop even if* PDAlg° *returns a solution* $(F, \ldots)$ *with* $|F| = k'$ for some $z = \frac{z_1 + z_2}{2}$, since *Theorem 4.6.3 is not strong enough to bound* $cost(F)$ *even when this happens!*.[3] If $|F| > k'$, we update $z_1 \leftarrow z$ and the F_1 solution; otherwise, we update $z_2 \leftarrow z$ and the F_2 solution. Thus, we maintain that $k_1 = |F_1| > k'$, and $k_2 = |F_2| \leq k'$.

Claim 4.6.4. *When* $z = z_2 = 2nk'c_{\max}$, PDAlg° *returns at most* k' *pairs.*

Proof. Let $(F, f, \mathsf{out}, \{\mathsf{rad}(i)\}_{i \in \mu(F)}, \alpha, \gamma)$ be the output of PDAlg° for this z. Let T be the set of tight pairs after the dual-ascent process. Observe that $\gamma \geq 2k'c_{\max}$, since for any tight pair $(i, r) \in T$, we have that $n\gamma \geq \sum_{j \in B(i,r) \cap \mathcal{D}'} \alpha_j \geq z$. We have $\sum_{j \in \mathcal{D}'} \alpha_j - m\gamma \leq OPT + k'z \leq k'c_{\max} + k'z$. On the other hand, since $\mathsf{uncov}(T \setminus f) \setminus \mathsf{OUT} \neq \emptyset$ and $\alpha_j = \gamma$ for all $j \in \mathsf{uncov}(T \setminus f)$, we also have the lower bound

$$\sum_{j \in \mathcal{D}'} \alpha_j - m\gamma \geq \sum_{\substack{i \in \mu(F \setminus f) \\ j \in B(i, \mathsf{rad}(i)) \cap \mathcal{D}'}} \alpha_j + \gamma \geq |F \setminus f|z + \gamma.$$

So if $|F| > k'$, we arrive at the contradiction that $\gamma \leq k'c_{\max}$. □

[3] Note that this is an artifact that never arises for k-median, or for other problems using the Lagrangian-relaxation technique coupled with LMP approximations (such as k-MST)

The main change is in the way solutions F_1, F_2 are combined. We modify the combination step in k-BSAlg to handle outliers (procedure $\mathcal{A}$ in Section 4.6.1), but the key extra ingredient is that we devise an alternate combination procedure $\mathcal{B}$ (Section 4.6.2) that returns a low-cost solution when $z_1 = \Omega(OPT)$. We return the better of the solutions output by the two procedures. In Section 4.6.3, we summarize these changes in Algorithm k-BSAlg$^\circ(\mathcal{D}', \mathcal{L}', k', \epsilon)$ and state the approximation bound for k-BSAlg$^\circ$ and LBkSRO. This immediately yields Theorem 4.6.1.

4.6.1 Combination subroutine $\mathcal{A}\big((F_1, \mathsf{rad}_1), (F_2, \mathsf{rad}_2)\big)$

As in the combination step in k-BSAlg, we cluster the F_1-pairs around F_2-pairs in stars. However, unlike before, some $(i', r') \in F_1$ may remain *unclustered* as (i', r') may only cover outliers of solution F_2 and thus it does not intersect with any pair in F_2. In the previous combination subroutine, for each pair $(i', r') \in F_1$, either (i', r') or some pair (i, r) close to (i', r') was picked (the case corresponding to merging pairs in a star). The reason behind this selection was to make sure that we cover all clients. Since now we do not need to cover all clients, we allow the flexibility of neither choosing $(i', r') \in F_1$ nor a pair close to it.

We again set up an LP to obtain a suitable collection of pairs. Let uc_p denote $\mathsf{uncov}(F_p)$ and $\mathcal{D}_p := \mathcal{D}' \setminus \mathsf{uc}_p$ for $p = 1, 2$. Let $\pi : F_1 \to F_2 \cup \{\emptyset\}$ be defined as follows: for each $(i', r') \in F_1$, if $(i', r') \in F_1$ intersects some F_2-pair, pick such an intersecting $(i, r) \in F_2$ and set $\pi(i', r') = (i, r)$; otherwise, set $\pi(i', r') = \emptyset$. In the latter case, (i', r') is unclustered, and $B(i', r') \subseteq \mathsf{uc}_2$. Define $\mathcal{S}_{i,r} = \pi^{-1}(i, r)$ for all $(i, r) \in F_2$. Let $\mathcal{Q} = \pi^{-1}(\emptyset)$. Let $\{\mathsf{uc}_1(i, r)\}_{(i,r)\in F_2}$ denote a partition of set $\mathsf{uc}_1 \cap \mathcal{D}_2$ such that for each $(i, r) \in F_2$, $\mathsf{uc}_1(i, r) \subseteq B(i, r)$. In essence, $\{\mathsf{uc}_1(i, r)\}_{(i,r)\in F_2}$ is a partition of F_1-outliers covered by F_2 that is induced by the F_2-balls. Similarly, let $\{\mathsf{uc}_2(i', r')\}_{(i',r')\in F_1}$ denote a partition of $\mathsf{uc}_2 \cap \mathcal{D}_1$ such that for each $(i', r') \in F_1$, $\mathsf{uc}_2 \subseteq B(i', r')$. We use an indicator variable $x_{i,r}$ for each $(i, r) \in F_2$ to indicate whether the F_1-pairs in $\mathcal{S}_{i,r}$ should be merged ($x_{i,r} = 1$) or not ($x_{i,r} = 0$). For each unclustered pair $(i', r') \in \mathcal{Q}$ (so $\pi(i', r') = \emptyset$), we use an indicator variable $q_{i',r'}$ to indicate whether this pair should be picked ($q_{i',r'} = 1$) or not ($q_{i',r'} = 0$). We consider the following LP, which can be written as 2-dimensional covering knapsack LP.

$$\min \sum_{(i,r)\in F_2} \Big(x_{i,r}(2r + \textstyle\sum_{(i',r')\in\mathcal{S}_{i,r}} 2r') + (1 - x_{i,r}) \sum_{(i',r')\in\mathcal{S}_{i,r}} r' \Big) + \sum_{(i',r')\in\mathcal{Q}} q_{i',r'} \cdot r' \qquad \text{(2C-P)}$$

$$\text{s.t.} \quad \sum_{(i,r)\in F_2} \big(x_{i,r} + |\mathcal{S}_{i,r}|(1 - x_{i,r}) \big) + \sum_{(i',r')\in\mathcal{Q}} q_{i',r'} \le k' \qquad (4.5)$$

$$- \sum_{(i,r)\in F_2} x_{i,r} |\mathsf{uc}_1(i,r)| + \sum_{(i',r')\in\mathcal{Q}} (1 - q_{i',r'}) |\mathsf{uc}_2(i',r')| \le 0 \qquad (4.6)$$

$$0 \le x_{i,r} \le 1 \qquad \forall (i,r) \in F_2$$
$$0 \le q_{i',r'} \le 1 \qquad \forall (i',r') \in \mathcal{Q}.$$

The interpretation of the variable $x_{i,r}$ is similar to before with a small modification. If $x_{i,r} = 0$, we select all pairs in $\mathcal{S}_{i,r}$. If $x_{i,r} = 1$, $\mathcal{S}_{i,r} \neq \emptyset$, we pick some $(i',r') \in \mathcal{S}_{i,r}$ and expand its radius suitably. Finally, (unlike before) if $x_{i,r} = 1$, $\mathcal{S}_{i,r} = \emptyset$, then we also pick (i,r) (see Lemma 4.6.5). Inequality (4.5) ensures that there are at most k' pairs picked. Inequality (4.6) ensures that our solution leaves at most m clients uncovered. To see this, note that the number of uncovered clients is

$$|\mathsf{uc}_1| + |\text{clients from } \mathcal{D}_1 \text{ that are not covered}| - |\text{clients in } \mathsf{uc}_1 \text{ that are covered}|.$$

The clients in $\mathcal{D}_1$ that are not covered correspond to unclustered pairs $(i',r') \in \mathcal{Q}$ that are not picked. The total number of such clients is at most $\sum_{(i',r')\in\mathcal{Q}} (1 - q_{i',r'}) \cdot |\mathsf{uc}_2(i',r')|$. The number of clients from uc_1 that are covered, is at least $\sum_{(i,r)\in F_2} x_{i,r} \cdot |\mathsf{uc}_1(i,r)|$. Thus, inequality 4.6 ensures that there are at most m uncovered clients (note that $|\mathsf{uc}_1| = m$).

The general idea in the combination subroutine $\mathcal{A}$ is similar to before; we first solve the LP-formulation (2C-P) to obtain an optimal (fractional) solution (x^*, q^*), then this solution is rounded to an integral solution $(\tilde{x}, \tilde{y})$ (see Lemma 4.6.5), and finally a good set F' of pairs is constructed from $(\tilde{x}, \tilde{y})$ (see Lemma 4.6.6). Subroutine $\mathcal{A}$ returns one of solutions $(F_2, \{\mathsf{rad}_2(i)\}_{i\in\mu(F_2)})$ or $(F', \{\mathsf{rad}(i)\}_{i\in\mu(F')})$ whichever has a smaller cost. We now describe these steps in more details and proceed with the analysis. The resulting combination subroutine $\mathcal{A}$ is summarized towards the end of this Section (after Claim 4.6.7), and Theorem 4.6.8 gives a bound on the cost of the solution returned.

Recall that $k_1 = |F_1|$, $k_2 = |F_2|$. Let $a, b \ge 0$ be such that $ak_1 + bk_2 = k'$, $a + b = 1$. Let $C_1 = cost(F_1)$ and $C_2 = cost(F_2)$.

Claim 4.6.5. *(x^*, q^*) can be rounded to a feasible integer solution $(\tilde{x}, \tilde{q})$ to (2C-P) of objective value at most $\text{OPT}_{\text{2C-P}} + 30R^*$.*

Proof. Let S be the set of fractional components of (x^*, q^*). As noted earlier, $|F|$ is at most the number of tight constraints from (4.5), (4.6). Let

$$l^* := \sum_{(i,r)\in S\cap F_2} \left(x^*_{i,r} + |\mathcal{S}_{i,r}|(1 - x^*_{i,r})\right) + \sum_{(i',r')\in S\cap\mathcal{Q}} q^*_{i',r'}$$

denote the contribution of the fractional components of (x^*, q^*) to the LHS of (4.5). Note that if (4.5) is tight, then l^* must be an integer. For a vector $v = (v_j)_{j\in I}$ where I is some index-set, let $\lceil v \rceil$ denote $\left(\lceil v_j \rceil\right)_{j\in I}$. Let $\mathcal{P} = \{(i, r) \in F_2 : \mathcal{S}_{i,r} = \emptyset\}$. We round (x^*, q^*) as follows.

- If $l^* \geq 2$ or $|S| \leq 1$ or $|S \cap (F_2 \setminus \mathcal{P})| \geq 1$, set $(\tilde{x}, \tilde{q}) = \lceil (x^*, q^*) \rceil$.
- Otherwise (i.e., $l^* < 2, S \subseteq \mathcal{P} \cup \mathcal{Q}, |S| = 2$), we set $\tilde{x}_{i,r} = x^*_{i,r}$, $\tilde{q}_{i',r'} = q^*_{i',r'}$ for all the integer-valued coordinates. We set the fractional component with larger absolute coefficient value on the LHS of (4.6) equal to 1 and the other fractional component to 0.

We prove that $(\tilde{x}, \tilde{q})$ is a feasible solution to (2C-P). Note that (4.6) holds for $(\tilde{x}, \tilde{q})$ since we always have

$$\sum_{(i,r)\in F_2} (1 - \tilde{x}_{i,r})|\mathsf{uc}_1(i, r)| + \sum_{(i',r')\in\mathcal{Q}} (1 - \tilde{q}_{i',r'})|\mathsf{uc}_2(i', r')| \leq \sum_{(i,r)\in F_2} (1 - x^*_{i,r})\mathsf{uc}_1(i, r) + \sum_{(i',r')\in\mathcal{Q}} (1 - q^*_{i',r'})|\mathsf{uc}_2(i', r')|.$$

Clearly, the contribution to the LHS of (4.5) from the components not in S is the same in both $(\tilde{x}, \tilde{q})$ and (x^*, q^*). Let l denote the contribution from $(\tilde{x}, \tilde{q})$ to the LHS of (4.5) from the components in S. Clearly, l is an integer.

If $l^* \geq 2$, then $l = 2$. If $|S| \leq 1$, then $l = 1$. If $l^* \geq 1$, then in these cases the LHS of (4.5) evaluated at $(\tilde{x}, \tilde{q})$ is at most the LHS of (4.5) evaluated at (x^*, q^*). If $l^* < 1$ and $|S| \leq 1$ (so $l = 1$), then since l^* is fractional, we know that (4.5) is not tight for (x^*, q^*). So despite the increase in LHS of (4.5), we have that (4.5) holds for $(\tilde{x}, \tilde{q})$. If $|S| = 2$ and $|S \cap (F_2 \setminus \mathcal{P})| \geq 1$, then we actually have $l^* > 1$ and $l = 2$. Again, since l^* is fractional, we can conclude that $(\tilde{x}, \tilde{q})$ satisfies (4.5) despite the increase in LHS of (4.5). Finally, suppose $l^* < 2$, $|S| = 2$, and $S \cap (F_2 \setminus \mathcal{P}) = \emptyset$. Then, given any (x, q), the contribution from S to the LHS of (4.5) is

$\sum_{(i,r)\in S\cap F_2} x_{i,r} + \sum_{(i',r')\in S\cap\mathcal{Q}} q_{i',r'}$, and at most one of the components in S is set to 1 in $(\tilde{x},\tilde{q})$. So $l = 1$, and either $l \le l^*$ or $l^* < 1$, and in both cases (4.5) holds for $(\tilde{x},\tilde{q})$.

To bound the objective value of $(\tilde{x},\tilde{q})$, notice that compared to (x^*,q^*), the solution $(\tilde{x},\tilde{q})$ pays extra only for the components that are rounded up. There are at most two such components, and their objective-function coefficients are bounded by $15R^*$, so the objective value of $(\tilde{x},\tilde{q})$ is at most $\text{OPT}_{\text{2C-P}} + 30R^*$.

□

Lemma 4.6.6. *A k-BS solution $\left(F', \{\mathsf{rad}(i)\}_{i\in\mu(F')}\right)$ can be obtained from $(\tilde{x},\tilde{q})$ with $cost(F') \le \text{OPT}_{\text{2C-P}} + 30R^*$ where $\left\{(i,\mathsf{rad}(i))\right\}_{i\in\mu(F')} \subseteq \mathcal{L}'$ is a set of non-intersecting pairs.*

Proof. We first construct F'' from $(\tilde{x},\tilde{q})$ as follows. If $\tilde{q}_{i',r'} = 1$, we include $(i',r') \in F''$ and set $\mathsf{rad}(i') = \mathsf{rad}_1(i')$. If $\tilde{x}_{i,r} = 0$, we include all pairs in $\mathcal{S}_{i,r}$ in F'' and set $\mathsf{rad}(i') = \mathsf{rad}_1(i')$ for all $(i',r') \in \mathcal{S}_{i,r}$. If $\tilde{x}_{i,r} = 1$ and $\mathcal{S}_{i,r} \neq \emptyset$, we pick a pair in $(i',r') \in \mathcal{S}_{i,r}$, and include $(i', 2r + r' + \max_{(i'',r'')\in\mathcal{S}_{i,r}\setminus\{(i',r')} 2r'')$ in F''. We set $\mathsf{rad}(i') = \mathsf{rad}_1(i')$. Now we initialize $F' = F''$ and consider all $(i,r) \in \mathcal{P}$ with $\tilde{x}_{i,r} = 1$. If (i,r) does not intersect any $(i',r') \in F''$ then we add (i,r) to F', and set $\mathsf{rad}(i) = \mathsf{rad}_2(i)$. Otherwise, if (i,r) intersects some $(i',r') \in F''$, then we replace $(i',r') \in F'$ with $(i',r'+2r)$. We have thus ensured that $\left\{(i,\mathsf{rad}(i))\right\}_{i\in\mu(F')} \subseteq \mathcal{L}'$ and consists of non-intersecting pairs. Note that in all the cases above, the total cost of the pairs we include when we process some $\tilde{q}_{i',r'}$ or $\tilde{x}_{i,r}$ term is at most the total contribution to the objective function from the $\tilde{q}_{i',r'}$ term, or the $\tilde{x}_{i,r}$ and $1-\tilde{x}_{i,r}$ terms. Therefore, $cost(F')$ is at most the objective value of $(\tilde{x},\tilde{q})$ which is at most $\text{OPT}_{\text{2C-P}} + 30R^*$. □

Claim 4.6.7. *We have $aC_1 + bC_2 \le (3+\epsilon)OPT + 4R^* + 3z_1$ and moreover, $\text{OPT}_{\text{2C-P}} \le 2bC_2 + (1+b)C_1$.*

Proof. This follows easily from Theorem 4.6.3 and since $cost(F_p) \le cost(F_p \setminus f_p) + R^*$ for $p = 1,2$. So we have $C_1 + 3(k_1-1)z_1 \le 3(OPT + k'z_1) + 4R^*$ and $C_2 + 3(k_2-1)z_2 \le 3(OPT + k'z_2) + 4R^*$. Combining these, we obtain

$$\begin{aligned} aC_1 + bC_2 &\le 3OPT + 3k'(az_1 + bz_2) - 3(ak_1z_1 + bk_2z_2) + 3(az_1 + bz_2) + 4R^* \\ &\le 3(OPT + k'z_2) - 3k'z_2 + 3ak_1\delta_z + 3z_1 + 3b\delta_z + 4R^* \\ &\le (3+\epsilon)OPT + 4R^* + 3z_1. \end{aligned}$$

where the second inequality follows since $0 \le z_2 - z_1 \le \delta_z = \frac{\epsilon OPT}{3n2^n}$.

In order to prove the second part, we present a feasible solution and use its cost to upper bound $\text{OPT}_{\text{2C-P}}$. We claim that setting $x_{i,r} = b$ for all $(i,r) \in F_2$, and $q_{i',r'} = a$ for all $(i',r') \in \mathcal{Q}$

yields a feasible solution to (2C-P). The LHS of (4.5) evaluates to $ak_1 + bk_2$, which is exactly k'. The first term on the LHS of (4.6) evaluates to $b\sum_{(i,r)\in F_2} |\mathsf{uc}_1(i,r)| = b|\mathsf{uc}_1|$. The second term on the LHS of (4.6) evaluates to at most $(1-a)\sum_{(i',r')\in\mathcal{Q}} \mathsf{uc}_2(i',r') = b|\mathsf{uc}_2|$. Since the number of F_1-outliers is equal to the number of F_2-outliers covered by F_2 (both solutions have exactly m outliers), we have $|\mathsf{uc}_1| = |\mathsf{uc}_2|$, so the inequality holds trivially. The objective value of this solution is

$$\sum_{(i,r)\in F_2} b\cdot 2r + \sum_{(i',r')\in F_1,\pi(i',r')\neq\emptyset} (1+b)\cdot r' + \sum_{(i',r')\in F_1,\pi(i',r')=\emptyset} a\cdot r' \leq 2bC_2 + (1+b)C_1$$

using the fact that $0 \leq a \leq 1$. □

Now that all the ingredients of subroutine $\mathcal{A}$ are presented, we can summarize this subroutine as follows. First, we solve the LP-formulation (2C-P) to obtain an optimal (fractional) solution (x^*, q^*). The number of fractional components in (x^*, q^*) is at most the number of tight constraints from (4.5), (4.6). We exploit this to round (x^*, q^*) to an integer solution $(\tilde{x}, \tilde{q})$ of good objective value (see Lemma 4.6.5), and then use $(\tilde{x}, \tilde{q})$ to extract a good set F' of pairs as sketched above (see Lemma 4.6.6). Combination subroutine $\mathcal{A}$ returns one of solutions $(F_2, \{\mathsf{rad}_2(i)\}_{i\in\mu(F_2)})$ or $(F', \{\mathsf{rad}(i)\}_{i\in\mu(F')})$ whichever has a smaller cost.

Theorem 4.6.8. *Combination subroutine $\mathcal{A}$ returns a feasible solution $\big(F, \{\mathsf{rad}(i)\}\big)$ with $cost(F) \leq (6.1821+3\epsilon)(OPT+z_1) + O(R^*)$ where $\big\{(i,\mathsf{rad}(i))\big\}_{i\in\mu(F)} \subseteq \mathcal{L}'$ is a set of non-intersecting pairs.*

Proof. We return (F_2, rad_2) if $cost(F_2) \leq cost(F')$, and $\big(F', \{\mathsf{rad}(i)\}_{i\in\mu(F')}\big)$ otherwise. Using Lemma 4.6.6, Claims 4.6.5 and 4.6.7 on the cost of $(F', \{\mathsf{rad}(i)\}_{i\in\mu(F')}$, we obtain that the cost of the solution returned is at most

$$\begin{aligned}\min\big\{C_2, 2bC_2 + (1+b)C_1\big\} + 30R^* &\leq 2.0607\big(aC_1 + bC_2\big) + 30R^* \\ &\leq 2.0607\Big((3+\epsilon)OPT + 4R^* + 3z_1\Big) + 30R^* \leq (6.1821+3\epsilon)(OPT+z_1) + 39R^*.\end{aligned}$$

where the first inequality follows from Claim 4.5.10 and the second inequality follows from Claim 4.6.7. □

4.6.2 Subroutine $\mathcal{B}\big((F_1, f_1, \mathsf{OUT}_1, \mathsf{rad}_1, \alpha^1, \gamma^1), (F_2, f_2, \mathsf{OUT}_2, \mathsf{rad}_2, \alpha^2, \gamma^2)\big)$

Subroutine $\mathcal{A}$ in the previous section yields a low-cost solution only if $z_1 = O(OPT)$. We complement subroutine $\mathcal{A}$ by now describing a procedure that returns a good solution when z_1 is

large. We assume in this section that $z_1 > (1+\epsilon)OPT$. Then $|F_1 \setminus f_1| \leq k'$ (otherwise $z \leq OPT$ by Theorem 4.6.3), so $|F_1 \setminus f_1| \leq k' < |F_1|$, which means that $k_1 = k' + 1$ and $f_1 \in F_1$.

First, we perform a preprocessing step on F_1 to take care of some simple cases. If there exists $(i, r) \in F_1 \setminus f_1$ such that $|\mathsf{uncov}\big(F_1 \setminus \{f_1, (i, r)\} \cup (i, r + 12R^*)\big)| \leq m$, then set $F = F_1 \setminus \{f_1, (i, r)\} \cup (i, r+12R^*)$. We have $cost(F) = cost(F_1 \setminus f_1) + 12R^* \leq 3 \cdot OPT + 15R^*$ (using Theorem 4.6.3). If there exist pairs $(i, r), (i', r') \in F_1$ such that $c(i, i') \leq 12R^*$, take r'' to be the minimum $\rho \geq r$ such that $B(i', r') \subseteq B(i, \rho)$ and set $F = F_1 \setminus \{(i, r), (i', r')\} \cup (i, r'')$. We have $cost(F) \leq cost(F_1 \setminus f_1) + 13R^* \leq 3 \cdot OPT + 16R^*$. In both cases, we return $\big(F, \{\mathsf{rad}_1(i)\}_{i \in \mu(F)}\big)$ (see Figure 4.6).

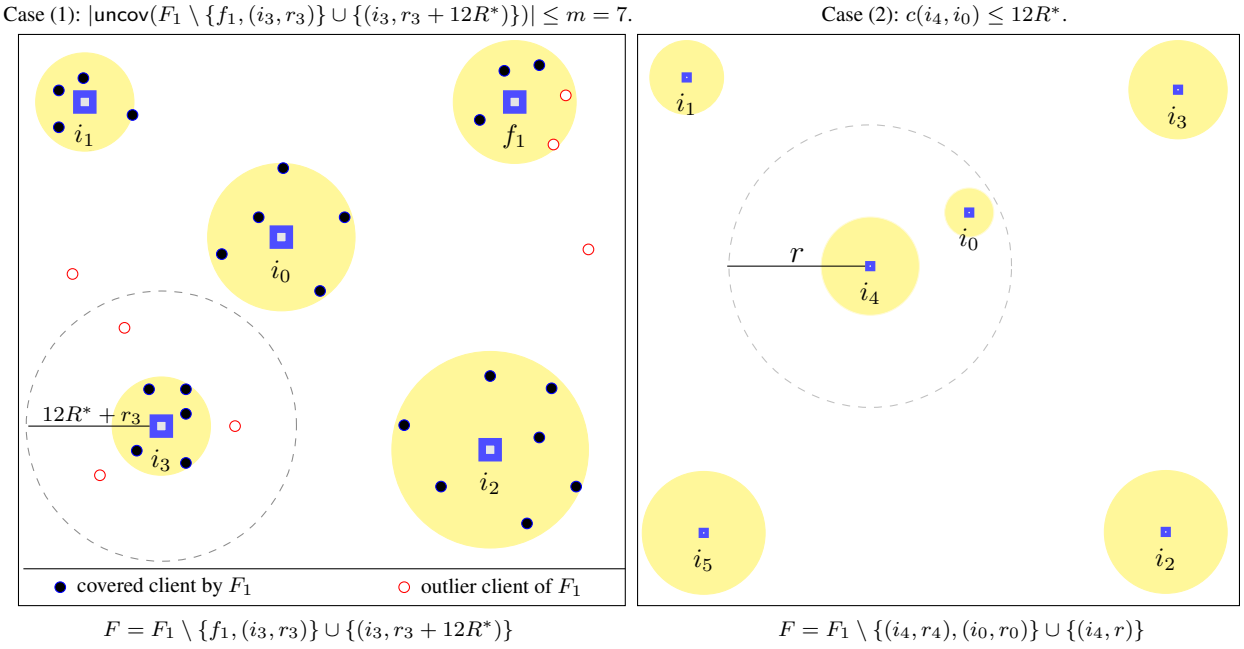


Figure 4.6: Example of the two cases resolved in the preprocessing step.

We assume in the sequel that neither of the above apply So all pairs in F_1 are well-separated. Note that, in particular, the last ball f_1 is far from $F_1 \setminus f_1$ and this means that $\alpha_j^1 = \gamma^1$ for all $j \in B_{f_1} \cap \mathcal{D}'$. We require the following continuity lemma, which generalizes Lemma 4.5.11; we defer the proof to the end of the chapter.

Lemma 4.6.9. *Let* $(F_p, \ldots, \alpha^p, \gamma^p) = \mathsf{PDAlg}^\circ(\mathcal{D}', \mathcal{L}', z_p)$ *for* $p = 1, 2$, *where* $0 \leq z_2 - z_1 \leq \delta_z$. *Then,* $\|\alpha_j^1 - \alpha_j^2\|_\infty \leq 2^n \delta_z$ *and* $|\gamma^1 - \gamma^2| \leq 2^n \delta_z$. *Thus, if* (4.2) *is tight for some* $(i, r) \in \mathcal{L}'$ *in one execution, then* $\sum_{j \in B(i,r) \cap \mathcal{D}'} \alpha_j^p \geq r + z_p - 2^n \delta_z$ *for* $p = 1, 2$.

Let $AT = \{(i,r) \in \mathcal{L}' : \sum_{j \in B(i,r) \cap \mathcal{D}'} \alpha_j^1 \geq r + z_1 - 2^n\delta_z\}$ and $AD = \{j \in \mathcal{D}' : \alpha_j^1 \geq \gamma^1 - 2^n\delta_z\}$. By Lemma 4.5.11, AT includes the tight pairs of $\mathsf{PDAlg}^{\circ}(\mathcal{F}, \mathcal{D}', c, \mathcal{L}', z_p)$ for both $p = 1, 2$, and $\mathsf{OUT}_1 \cup \mathsf{OUT}_2 \subseteq AD$. Since the tight pairs T_2 used for building solution F_2 are almost tight in $(\alpha^1, \gamma^1, z_1)$, we swap them in and swap out pairs from F_1 one by one while maintaining a feasible solution. Either at some point, we will be able to remove f, which will give us a solution of size k', or we will obtain a bound on $cost(F_2)$. The following lemma is our main tool for bounding the cost of the solution returned.

Lemma 4.6.10. *Let $F \subseteq \mathcal{L}'$, and let $T_F = \{(i, r_i')\}_{i \in \mu(F)}$ where $r_i' \leq r$ for each $(i,r) \in F$. Suppose $T_F \subseteq AT$ and pairs in T_F are non-intersecting. If $|F| \geq k'$ and $|AD \setminus \bigcup_{(i,r) \in F} B(i,r))| \geq m$ then $cost(T_F) \leq (1+\epsilon)OPT$. Moreover, if $|F| > k'$ then $z_1 \leq (1+\epsilon)OPT$.*

Proof. Let OUT_F be a subset of exactly m of clients from $AD \setminus \bigcup_{(i,r) \in F} B(i,r)$. Since the pairs in T_F are non-intersecting and almost tight, $\sum_{i \in \mu(F)}(r_i' + z) \leq \sum_{j \in \mathcal{D}' \setminus \mathsf{OUT}_F}(\alpha_j^1 + 2^n\delta_z)$, so

$$\sum_{i \in \mu(F)} (r_i' + z) \leq \sum_{j \in \mathcal{D}'} (\alpha_j^1 + 2^n\delta_z) - m(\gamma^1 - 2^n\delta_z) \leq \sum_{j \in \mathcal{D}'} \alpha_j^1 - m\gamma^1 + (m + |\mathcal{D}'|)2^n\delta_z \leq (1+\epsilon)OPT + k'z_1$$

where the last inequality follows since $(\alpha^1, \gamma^1, z_1)$ is a feasible solution to (D_3). So $cost(T_F) \leq (1+\epsilon)OPT$ if $|T_F| = |F| \geq k'$, and $z_1 \leq (1+\epsilon)OPT$ if $|F| > k'$. □

Define a mapping $\psi : F_2 \to F_1 \setminus f_1$ as follows. Note that any $(i,r) \in F_2$ may intersect with at most one F_1-pair: if it intersects $(i', r'), (i'', r'') \in F_1$, then we have $c(i', i'') \leq 12R^*$ which contradicts the fact that we assume that we are not in one of the simple cases. First, for each $(i,r) \in F_2$ that intersects with some $(i', r') \in F_1$, we set $\psi(i,r) = (i', r')$. Let $M \subseteq F_2$ be the F_2-pairs mapped by ψ this way. For every $(i,r) \in F_2 \setminus M$, we arbitrarily match (i,r) with a *distinct* $(i', r') \in F_1 \setminus \psi(M)$. We claim that ψ is, in fact, a one-one function.

Lemma 4.6.11. *Every $(i,r) \in F_1 \setminus f_1$ intersects with at most one F_2-pair.*

Proof. Suppose two pairs $(i_1, r_1), (i_2, r_2) \in F_2$ intersect with a common pair $(i,r) \in F_1 \setminus f_1$. Let $T_{1,I}$ be the tight pairs corresponding to $F_1 \setminus f_1$ obtained from the (pruning phase of) $\mathsf{PDAlg}^{\circ}(\mathcal{D}', \mathcal{L}', z_1)$. Let $(i, \mathsf{rad}_1(i)) \in T_{1,I}$ be the tight pair corresponding to (i,r). Let $(i_1, \mathsf{rad}_2(i_1)), (i_2, \mathsf{rad}_2(i_2))$ be the tight pairs corresponding to $(i_1, r_1), (i_2, r_2)$ obtained from $\mathsf{PDAlg}^{\circ}(\mathcal{D}', \mathcal{L}', z_2)$. We will show that either $z_1 \leq OPT$, or that $|\mathsf{uncov}(F_1 \setminus \{f_1, (i,r)\} \cup \{(i, r + 12R^*)\})| \leq m$, both of which lead to a contradiction.

Define $F' = F_1 \setminus \{f_1, (i,r)\} \cup \{(i_1, r_1), (i_2, r_2)\}$ (so $|F'| = k+1$), and define $T_{F'} = T_{1,I} \setminus \{(i, \mathsf{rad}_1(i))\} \cup \{(i_1, \mathsf{rad}_2(i_1)), (i_2, \mathsf{rad}_2(i_2))\}$ (See Figure 4.7). Since $(i_1, \mathsf{rad}_2(i_1))$ and

$(i_2, \mathsf{rad}_2(i_2))$ are non-intersecting and they do not intersect with any pair in $T_{1,I} \setminus (i, \mathsf{rad}_1(i))$, the pairs in $T_{F'}$ are non-intersecting. Also, $T_{F'} \subseteq AT$. If $|AD \setminus \bigcup_{(i',r') \in F'} B(i', r')| \geq m$, then $z \leq OPT$ by Lemma 4.6.10. Otherwise, note that every client $j \in B(i_1, r_1) \cup B(i_2, r_2)$ is at distance at most $r + 2\max(r_1, r_2) \leq r + 6R^*$ from i. So setting $F'' = F_1 \setminus \{f_1, (i, r)\} \cup \{(i, r + 12R^*)\}$, we have $\mathsf{uncov}(F'') \subseteq \mathsf{uncov}(F') \subseteq AD$, and so $|\mathsf{uncov}(F'')| \leq |AD \setminus \bigcup_{(i',r') \in F'} B(i', r')| \leq m$.

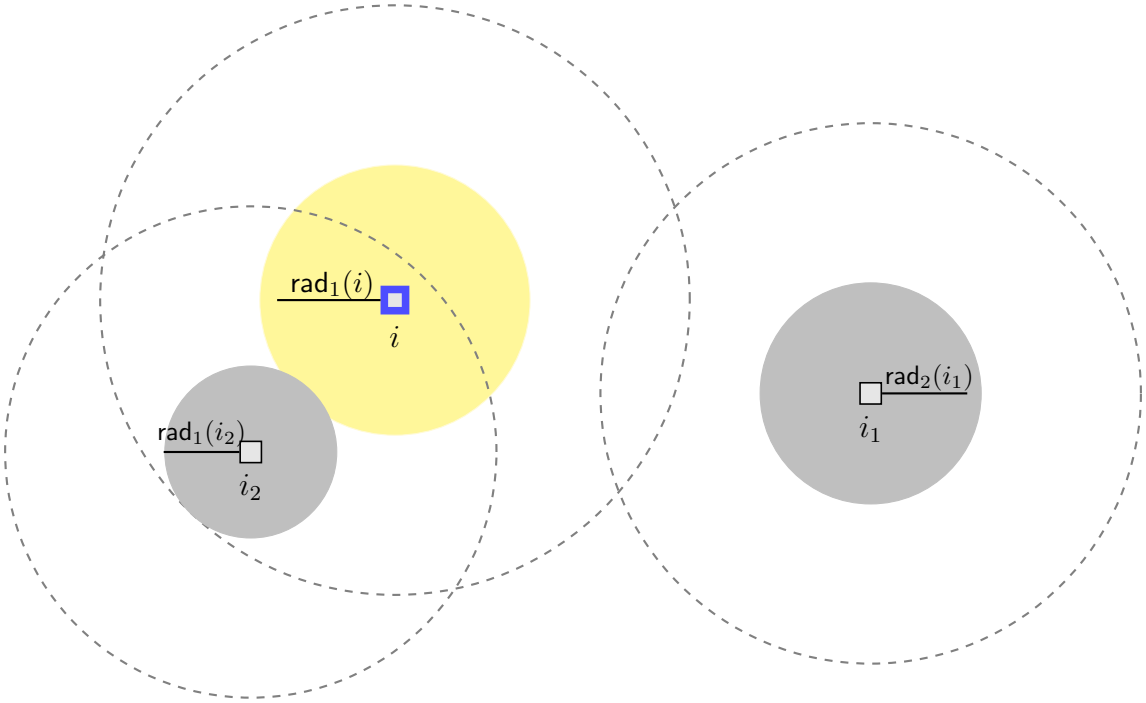


Figure 4.7: Depiction of the case that a pair (i, r) in F_1 intersects with two pairs (i_1, r_1) and (i_2, r_2) in F_2.

□

Let F_2' be the pairs $(i, r) \in F_2$ such that if $(i', r') = \psi(i, r)$, then $r' < r$. Let $P = F_2' \cap M$ and $Q = F_2' \setminus M$. For every $(i', r') \in \psi(Q)$ and $j \in B(i', r')$, we have $j \in \mathsf{uncov}(F_2) \subseteq AD$ (else (i', r') would lie in $\psi(M)$). Starting with $F' = F_1 \setminus f_1$, we iterate over $(i, r) \in F_2'$ and do the following. Let $(i', r') = \psi(i, r)$. If $(i, r) \in P$, we update $F' \leftarrow F' \setminus (i', r') \cup (i, r + 2r')$ (so $B(i, r + 2r') \supseteq B(i', r')$), else we update $F' \leftarrow F' \setminus (i', r') \cup (i, r)$. Let $T_{F'} = \{(i, \mathsf{rad}_1(i))\}_{(i,r) \in F' \cap F_1} \cup \{(i, \mathsf{rad}_2(i))\}_{(i,r) \in F' \setminus F_1}$. Note that $|F'| = k'$ and $\mathsf{uncov}(F') \subseteq AD$ at all times. Also, since (i, r) intersects only (i', r'), which we remove when (i, r) is added, we maintain that $T_{F'}$ is a collection of non-intersecting pairs and a subset of $AT \subseteq \mathcal{L}'$. This process continues until

$|\mathsf{uncov}(F')| \leq m$, or when all pairs of F_2' are swapped in (see Figure 4.8). In the former case, we argue that $cost(F')$ is small and return $\big(F', \{\mathsf{rad}_1(i)\}_{(i,r)\in F'\cap F_1} \cup \{\mathsf{rad}_2(i)\}_{(i,r)\in F'\setminus F_1}\big)$. In the latter case, we show that $cost(F_2')$, and hence $cost(F_2)$ is small, and return (F_2, rad_2).

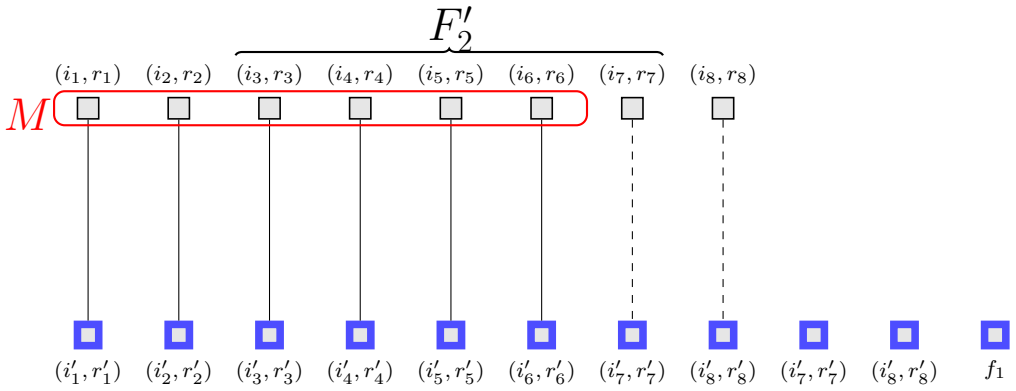


Figure 4.8: Combination step $\mathcal{B}$: The vertices on top represent F_2 pairs and the vertices on the bottom represent F_1 pairs. An edge between $(i,r) \in F_2$ and $(i',r') \in F_1 \setminus f_1$ indicates that $(i',r') = \psi(i,r)$: If (i,r) and (i',r') intersect then the edge is solid otherwise it is dotted. In this example, $P = \{(i_l, r_l) : 3 \leq l \leq 6\}$ and $Q = \{(i_7, r_7)\}$.

Lemma 4.6.12. *(i) If the algorithm stops with* $|\mathsf{uncov}(F')| \leq m$*, then* $cost(F') \leq (9+3\epsilon)OPT + 18R^*$.

(ii) If case (i) does not apply, then $cost(F_2) \leq (3+3\epsilon)OPT + 9R^*$.

(iii) The pairs corresponding to the radii returned are non-intersecting and form a subset of $\mathcal{L}'$.

Proof. Part (iii) follows readily from the algorithm description and the discussion above. Consider part (i). Let $(i,r) \in F_2'$ be the last pair scanned by the algorithm before it terminates, and $(i',r') = \psi(i,r)$. Let F'' be the set F' just before the last iteration. So $F'' = F' \setminus (i, r+2r') \cup (i',r')$ if $(i,r) \in P$, and $F'' = F' \setminus (i,r) \cup (i',r')$ if $(i,r) \in Q$. Note that $r + 2r' \leq 9R^*$. Since $\mathsf{uncov}(F'') \subseteq AD$ and $|\mathsf{uncov}(F'')| > m$, by Lemma 4.6.10, we have $cost(T_{F''}) \leq (1+\epsilon)OPT$. For all $(i,r) \in F_1$, we have $r \leq 3\mathsf{rad}_1(i)$ (since $f_1 \in F_1$). For all but at most one $(i,r) \in F_2$, we have $r \leq 3\mathsf{rad}_2(i)$ and for the one possible exception, we have $r \leq 3R^*$. Therefore,

$$\begin{aligned} cost(F') &\leq cost(F'' \cap F_1) + cost(F'' \setminus F_1) + 9R^* \leq 3 \cdot cost(T_{F''}) + 3R^* + 2 \cdot cost(F_1 \setminus F'') + 9R^* \\ &\leq 3(1+\epsilon)OPT + 3R^* + 2(3 \cdot OPT + 3R^*) + 9R^* = \big(9+3\epsilon\big)OPT + 18R^*. \end{aligned}$$

For part (ii), the same argument as above shows that $cost(T_{F'}) \leq (1+\epsilon)OPT$ and $cost(F_2') + cost\big(F_1 \setminus (f_1 \cup \psi(F_2'))\big) \leq 3 \cdot cost(T_{F'}) + 3R^*$. Now

$$\begin{aligned} cost(F_2) = cost(F_2') + cost(F_2 \setminus F_2') &\leq cost(F_2') + cost\big(\psi(F_2 \setminus F_2')\big) \\ &\leq cost(F_2') + cost\big(F_1 \setminus (f_1 \cup \psi(F_2'))\big) \\ &\leq 3(1+\epsilon) \cdot OPT + 3R^* \end{aligned}$$

where the first inequality follows by the definition of F_2'.

□

Theorem 4.6.13. *If $z > (1+\epsilon)OPT$ then, combination subroutine $\mathcal{B}$ returns a feasible solution $\big(F, \{\mathsf{rad}(i)\}\big)$ with $cost(F) \leq (9+3\epsilon)OPT + O(R^*)$ where $\big\{(i, \mathsf{rad}(i))\big\}_{i \in \mu(F)} \subseteq \mathcal{L}'$ is a set of non-intersecting pairs.*

Proof. If $z_1 > (1+\epsilon)OPT$, then $|F_1 \setminus f_1| \leq k'$ (otherwise $z \leq OPT$ by Theorem 4.6.3), so $|F_1 \setminus f_1| \leq k' < |F_1|$, which means that $k_1 = k' + 1$ and $f_1 \in F_1$. If either of the simple cases apply, then the cost of the returned solution is at most $cost(F_1) + O(R^*)$ which is at most $cost(F_1 \setminus f_1) + r_f + O(R^*) \leq 3OPT + O(R^*)$ by Theorem 4.6.3. In both these cases $\big\{(i, \mathsf{rad}(i))\big\}_{i \in \mu(F)}$ is a set of non-intersecting pairs as this set is a subset of $\big\{(i, \mathsf{rad}(i))\big\}_{i \in \mu(F_1)}$ which is a subset of $\mathcal{L}'$. If the combination step does not return a solution in the preprocessing step then it either returns $(F', \{\mathsf{rad}(i)\}_{i \in \mu(F')})$ or $(F_2, \{\mathsf{rad}(i)\}_{i \in \mu(F_2)})$ whichever has a smaller cost. Using Lemma 4.6.12, $cost(F) \leq (9+3\epsilon)OPT + O(R^*)$ and satisfies the condition in the Theorem. □

4.6.3 Analysis of k-BSAlg° Algorithm and LBkSRO Algorithm

The k-BSAlg° algorithm, which uses the subroutine $\mathcal{A}$ and $\mathcal{B}$ discussed in Sections 4.6.1 and 4.6.2 respectively, is summarized below.

Algorithm k-**BSAlg°**$(\mathcal{D}', \mathcal{L}', k')$. Output: $F \subseteq \mathcal{L}$ with $|F| \leq k'$, a radius $\mathsf{rad}(i)$ for all $i \in \mu(F)$.

C1. **Binary search.** Let $(F_1, \mathsf{rad}_1, \ldots) = \mathsf{PDAlg°}(\mathcal{D}', \mathcal{L}', 0)$. If $|F_1| \leq k'$ pairs, return (F_1, rad_1). Else perform binary-search in the range $[0, 2nk'c_{\max}]$ to find z_1, z_2 with $0 \leq z_2 - z_1 \leq \delta_z = \frac{\epsilon OPT}{3n2^n}$ such that letting $(F_p, f_p, \mathsf{OUT}_p, \mathsf{rad}_p, \alpha^p, \gamma^p) = \mathsf{PDAlg}(\mathcal{D}', \mathcal{L}', z_p)$ for $p = 1, 2$, we have $|F_2| \leq k' < |F_1|$.

C2. Let $\big(F_{\mathcal{A}}, \{\mathsf{rad}_{\mathcal{A}}(i)\}_{i \in \mu(F_{\mathcal{A}})}\big) = \mathcal{A}\big((F_1, \mathsf{rad}_1), (F_2, \mathsf{rad}_2)\big)$ (Section 4.6.1). If $|F_1 \setminus f_1| > k'$, return $(F_{\mathcal{A}}, \mathsf{rad}_{\mathcal{A}})$.

C3. Preporocessing in subroutine $\mathcal{B}$: If $\exists (i,r) \in F_1 \setminus f_1$ such that $|\mathsf{uncov}\big(F_1 \setminus \{f_1,(i,r)\} \cup (i, r + 12R^*)\big)| \leq m$, then set $F = F_1 \setminus \{f_1,(i,r)\} \cup (i, r + 12R^*)$. If $\exists (i,r),(i',r') \in F_1$ such that $c(i,i') \leq 12R^*$, let $r'' \geq r$ be the minimum $\rho \geq r$ such that $B(i',r') \subseteq B(i,\rho)$; set $F = F_1 \setminus \{(i,r),(i',r')\} \cup (i,r'')$. In either of the above apply, return $\big(F, \{\mathsf{rad}_1(i)\}_{i \in \mu(F)}\big)$.

C4. Let $\big(F_{\mathcal{B}}, \{\mathsf{rad}_{\mathcal{B}}(i)\}_{i \in \mu(F_{\mathcal{B}})}\big)$ be the output of subroutine $\mathcal{B}$ if no solution return in C3 (Section 4.6.2).

C5. If $cost(F_{\mathcal{A}}) \leq cost(F_{\mathcal{B}})$, return $(F_{\mathcal{A}}, \mathsf{rad}_{\mathcal{A}})$, else return $(F_{\mathcal{B}}, \mathsf{rad}_{\mathcal{B}})$.

Theorem 4.6.14. *k-$\mathsf{BSAlg}^{\circ}(\mathcal{D}', \mathcal{L}', k')$ returns a solution (F, rad) with $cost(F) \leq \big(12.365 + O(\epsilon)\big) \cdot OPT + O(R^*)$ where $\big\{(i, \mathsf{rad}(i))\big\}_{i \in \mu(F)} \subseteq \mathcal{L}'$ comprises non-intersecting pairs.*

Proof. This follows essentially from Theorem 4.6.8 and Theorem 4.6.13. When $z_1 \leq (1+\epsilon) \cdot OPT$, Theorem 4.6.8 yields the above bound on $cost(F_{\mathcal{A}})$. Otherwise, Theorem 4.6.13 yields the bound $(9+3\epsilon)OPT + O(R^*)$ on the $cost(F_{\mathcal{B}})$ or the solution returned in C_3 which is dominated by the term in the theorem. □

We now have all the ingredients needed for proving our main theorem for the LBkSRO problem.

Proof of Theorem 4.6.1. Again when $F^O = \{(i_1, r_1), \ldots (i_t, r_t)\}$ in Algorithm 2 corresponds to the t facilities in an optimal solution with largest radii, we obtain the desired approximation bound. In this case, if $t = k$, then F^O is an optimal solution, otherwise, we have $t \geq \frac{1}{\epsilon}$ and $R^* \leq \frac{O^*}{t} \leq \epsilon O^*$ and $OPT \leq O^* - \sum_{p=1}^{t} r_p$. Using Theorem 4.6.14, we get the result in the theorem. □

4.7 Proof of Lemma 4.6.9 (continuity lemma)

Let us abbreviate $\mathsf{PDAlg}(\mathcal{F}, \mathcal{D}', c, \mathcal{L}', z)$ to $\mathsf{PDAlg}(z)$ and let x^- denote a quantity infinitesimally smaller than x. Consider the dual-ascent phase of PDAlg for z_1 and z_2. Sort clients with respect to their $\alpha_j^0 = \min(\alpha_j^1, \alpha_j^2)$ value, i.e., the earliest time they become tight in one of the two executions. Let this ordering be $\alpha_1^0 \leq \alpha_2^0 \leq \cdots \leq \alpha_n^0$. We prove by induction that $|\alpha_j^1 - \alpha_j^2| \leq 2^{j-1}\delta_z$.

For the base case, assume without loss of generality that $\alpha_j^0 = \alpha_j^1$ for $j = 1$, and let (i,r) be the tight pair that caused j to become inactive in $\mathsf{PDAlg}(z_1)$. Consider time point $t = \alpha_1^0$ in the two executions. By definition all clients are active at time t^- in $\mathsf{PDAlg}(z_2)$. So the contribution $\sum_{j \in B(i,r) \cap \mathcal{D}'} \alpha_j$ of clients to the LHS of (4.2) at time t^- is at least as much as their contribution

in $\mathsf{PDAlg}(z_1)$ at time t^-. Therefore, we can increase α_1 by at most δ_z beyond time t in $\mathsf{PDAlg}(z_2)$ as $z_2 - z_1 = \delta_z$.

Suppose we have shown that for all clients $j = 1, 2, \cdots, \ell - 1$ (where $\ell \geq 2$), $|\alpha_j^1 - \alpha_j^2| \leq 2^{j-1}\delta_z$. Now consider client ℓ and let (i, r) be the tight pair that makes ℓ inactive at time α_ℓ^0 in $\mathsf{PDAlg}(z_p)$, where $p \in \{1, 2\}$. Consider time point $t = \alpha_\ell^0$ in both executions. By definition, all clients $j > \ell$ are still active at time t^- in both executions $\mathsf{PDAlg}(z_1)$ and $\mathsf{PDAlg}(z_2)$. (They might become inactive at time t but can not become inactive earlier.) The contribution $\sum_{j \in B(i,r) \cap \mathcal{D}'} \alpha_j$ of clients to the LHS of (4.2) in the execution other than p at time t^- is at least their contribution in $\mathsf{PDAlg}(z_p)$ at time t^- minus $\sum_{j=1}^{\ell-1} 2^{j-1}\delta_z$. The values of z in the two executions differs by at most δ_z, so in the execution other than p, α_ℓ can grow beyond t by at most $(1 + \sum_{j=1}^{\ell-1} 2^{j-1})\delta_z \leq 2^\ell \delta_z$.

Now if we consider a tight pair (i, r) in one of the execution, the value of RHS and LHS of $\sum_{j \in B(i,r)} \alpha_j \leq r + z$ for the other execution can differ by at most $(1 + \sum_{j=1}^{n} 2^{j-1})\delta_z \leq 2^n \delta_z$.

Now consider the case where $m > 0$. Note that in this case, we can assume that we have the execution for $m = 0$, pick the first time at which there are at most m active clients, i.e., time γ in PDAlg°, and set $\alpha_j = \gamma$ for every active client at this time point. Let $\gamma^0 = \min(\gamma^1, \gamma^2)$, suppose $\gamma^0 = \gamma_p$, where $p \in \{1, 2\}$. Note that by time $\gamma^0 + 2^n \delta_z$, all pairs that are tight in the p-th execution by time γ^0 are also tight in the other execution. So the number of active clients after this time point is at most m. Therefore $|\gamma^1 - \gamma^2| \leq 2^n \delta_z$.

4.8 Equivalence of lower-bounded k-supplier with outliers and lower-bounded k-center with outliers

Let $\mathsf{LB}k\mathsf{CentO}$ denote the special case of $\mathsf{LB}k\mathsf{SupO}$ where $\mathcal{F} = \mathcal{D}$. In this section, we show that if there exists an α-approximation for $\mathsf{LB}k\mathsf{CentO}$, then there exists an α-approximation for $\mathsf{LB}k\mathsf{SupO}$. Let $\mathcal{I} = (\mathcal{F}, \mathcal{D}, c, L, k, m)$ be an instance of $\mathsf{LB}k\mathsf{SupO}$ with $N = |\mathcal{F}|+1$ and $|\mathcal{D}| = n$. Define an instance $\mathcal{I}' = (\mathcal{D}', c', L', k', m')$ as follows: let $k' = k$, let $\mathcal{D}' = (\mathcal{D} \times \{1, 2, \cdots, N\}) \cup \mathcal{F}$ ($|\mathcal{D}'| = nN + N - 1$), let $c'((j, p), i) = c(j, i)$ for each $j \in \mathcal{D}, p \in [N], i \in \mathcal{F}$ and let $c'(q, q')$ be defined as symmetric closure of the fixed distances for $q, q' \in \mathcal{D}'$, let $L'_i = L_i$ for $i \in \mathcal{F}$ and $L'_{(j,p)} = N(n + 1)$, and let $m' = N \cdot m + (N - 1)$. Clearly $\mathcal{I}'$ can be constructed from $\mathcal{I}$ in polynomial time. Note that lower bounds for (j, p), $j \in \mathcal{D}, p \in [N]$ is set so that it is smaller than the number of clients, i.e., $L'_{(j,p)} < |\mathcal{D}'|$, so (j, p) cannot be opened as a center.

Let $OPT(\mathcal{I}')$ denote the value of optimal solution of $\mathcal{I}'$ and $OPT(\mathcal{I})$ denote the value of optimal solution of $\mathcal{I}$. We claim that $OPT(\mathcal{I}') \leq OPT(\mathcal{I})$. Let (F^*, σ^*) denote an optimal solution of $\mathcal{I}$. Let solution $(\hat{F}, \hat{\sigma})$ for $\mathcal{I}$ be constructed as follows: let $\hat{F} = F^*$, for each $p \in [N]$,

define $\hat{\sigma}(q) = i$ for $q = (j, p)$ if $\sigma^*(j) = i$, and $\hat{\sigma}(q) = \mathsf{out}$ otherwise. Note that since there are at most m outliers in solution (F^*, σ^*) then there are at most $N*m+|\mathcal{F}| = N*m+(N-1)$ outliers in $(\hat{F}, \hat{\sigma})$. Clearly the radius of opened centers is the same as before, so $OPT(\mathcal{I}') \leq OPT(\mathcal{I})$.

Now suppose there exists an α-approximation algorithm $\mathcal{A}$ for LBkCentO problem. We use this algorithm to obtain an α-approximation solution for LBkSupO instance $\mathcal{I}$. First, we use Algorithm $\mathcal{A}$ on LBkCentO instance $\mathcal{I}'$. Let (F', σ') be an output of Algorithm $\mathcal{A}$ with maximum radius r. Then we use Algorithm 3 described below to find a solution (F, σ) with radius r. Since $r \leq \alpha. \cdot OPT(\mathcal{I}')$, $r \leq \alpha OPT(\mathcal{I})$ as well and we have an α-approximation algorithm.

Algorithm 3 Algorithm for constructing feasible assignment LBkSupO solution (F, σ)

Input: LBkCentO instance $\mathcal{I}$, Instance LBkCentO solution (F', σ') found by Algorithm $\mathcal{A}(\mathcal{I}')$.

1: $F \leftarrow F'$.
2: Construct network $\mathcal{N} = (V, E)$ where $V = \{s, t\} \cup \mathcal{D} \cup F$ and $E = \{si : i \in F\} \cup \{ij : i \in F, j \in \mathcal{D}, c(i, j) \leq r\} \cup \{jt : j \in \mathcal{D}\}$.
3: $u_{ij} = 1$ for each $ij \in E, i \in F, j \in \mathcal{D}$.
4: $l_{si} = L_i$ for each $si \in E, i \in F$.
5: $u_{jt} = 1$ for each $jt \in E, j \in \mathcal{D}$.
6: $f \leftarrow \max - flow(\mathcal{N})$ respecting lower bounds l and upper bounds u on edges.
7: **if** value of f is $\geq n - m$ **then**,
8: $\quad \sigma(j) = i$ if $f_{jt} = 1$ and $f_{ij} = 1$ for $i \in F$.
9: $\quad \sigma(j) = \mathsf{out}$ if $f_{jt} = 0$.
10: $\quad$ **return** (F, σ).

Lemma 4.8.1. *Let (F', σ') be a feasible solution of* LBkCentO *instance $\mathcal{I}'$ with maximum radius at most r, then solution (F, σ) is a feasible solution to* LBkSupO *instance $\mathcal{I}$ with maximum radius at most r where σ is an output of Algorithm 3.*

Proof. Consider any set $S \subseteq F'$. There are at least $\sum_{i \in S} N \cdot L_i$ clients in $\mathcal{D}'$ assigned to S by σ'. Since there are $N - 1$ facilities in $\mathcal{D}'$, there are at least $\frac{\sum_{i \in S} N \cdot L_i - (N-1)}{N}$ clients at distance at most r from S. Note that the smallest number greater than $\frac{\sum_{i \in S} N \cdot L_i - (N-1)}{N}$ is $\sum_{i \in S} L_i$, so there are at least $\sum_{i \in S} L_i$ clients in neighbor set of any S in $F = F'$. Since each client at distance at most r from F can let 1 unit of flow through from F, there exists a flow f satisfying lower bounds and upper bounds on the edges.

It remains to show that value of f is at least $n - m$. If there is an incoming edge to a client j in $\mathcal{N}$, then a flow of 1 can be sent through j. So we want to bound the number of clients with no incoming edge in $\mathcal{N}$. If some copy of client j is assigned to some facility in solution (F', σ')

then j is at distance at most r of some facility in $F = F'$. Since there are at most $Nm + (N-1)$ outliers in (F', σ'), there are at most $\frac{Nm+(N-1)}{N}$ clients with no incoming edge in $\mathcal{N}$. Therefore there are at most m clients with no incoming edge as the largest integer smaller than $\frac{Nm+(N-1)}{N}$ is m. So there are at most m outliers. □

Chapter 5

Conclusions

In this thesis, we consider some sophisticated facility-location problems that better abstract some real-world settings than the basic FL problems like UFL, CFL, k-median. We develop techniques for tackling these problems by leveraging our understanding of basic FL problems, and our techniques yield the first and/or the best approximation guarantees for these problems. Our results open many avenues for future work, some of which are presented below.

Mobile facility location problem (MFL). An immediate question is whether the approximation factor can be improved. As mentioned in Chapter 2, MFL generalizes k-median problem and our $(3+\epsilon)$-approximation ratio matches the best approximation ratio for k-median based on local-search technique. Shortly after our result, Li and Svensson [64] followed by Byrka et al. [16], presented LP-based approaches yielding $(2.732+\epsilon)$ and $(2.675+\epsilon)$ approximation ratios, respectively. One interesting question is whether these improved LP-based methods for k-median can be utilized to obtain analogous improvements for MFL.

Our local-search based approach fails to deliver a good solution in the case that facilities and clients move in different metrics (see Section 2.8). All algorithms with $O(1)$ approximation ratio for this case are based on linear programming [34, 79]. A natural question is whether a combinatorial algorithm or local-search based algorithm with constant factor approximation for this case can be developed.

More generally, MFL (even with different client-assignment and facility-movement metrics) is a special case of the *matroid median* problem [60, 79]. A captivating open question is whether a combinatorial approximation algorithm can be developed for matroid median.

Minimum load k-facility location (MLkFL). This problem is rather poorly understood and other than our result for the line-metrics, star-metrics and tree-metrics, no approximation except the trivial $O(k)$-approximation, obtained by returning a k-median solution, is known. The most prominent open question is to find a true (non-bicriteria) approximation algorithm for the general case.

Our algorithm for line metrics is quite involved and an interesting question is whether a simple algorithm with a constant factor approximation can be obtained. One step in this direction would be to understand whether there exists a near-optimal solution without any crossings, i.e., a solution where client-assignments form a laminar family and the cost of the solution is within a constant factor of optimal cost.

In [4], we present a QPTAS for tree-metrics and a poly-time algorithm with constant factor approximation when the tree defining the metrics is a star. It is open to obtain a PTAS for tree metrics, or even a constant-factor polytime approximation. It would also be interesting to obtain an $O(1)$-approximation for Euclidean metrics, and we conjecture that such an algorithm exists.

Lower-bounded clustering. The main open question is whether our techniques, which are guided by the Lagrangian-relaxation paradigm but circumvent the difficulty posed by not having a Lagrangian-multiplier-preserving algorithm, can be applied to other problems, such as, most notably, k-median with outliers. Our approach uses different insights about the solutions obtained by the primal-dual algorithm. These insights should be helpful in devising an approximation algorithm for k-median with outlier as well. The only true-approximation result known for k-median problem with outliers is by Chen [23] who obtained a large constant-factor approximation for this problem (the constant is not calculated in [23] and its value is at least in the hundreds).

A closely related problem to the lower-bounded clustering problems that we consider, is lower-bounded facility location. Recall that in this problem, the cost of the solution consists of the sum of the facility cost and client assignment costs. The only results known for this problem are in the setting of uniform lower-bounds and the current state-of-the-art is 83 by [6]. Even for the case that the facility opening costs are zero, no approximation is known for non-uniform lower-bounds.

In terms of lower-bounds for the approximability of the min-sum-of-radii problem, it is only known that the problem is *NP*-hard; its complexity is not yet settled even for shortest-path metrics of unweighted graphs with only a quasipolytime (exact) algorithm known [36]. One fundamental question is to understand the hardness of this problem more precisely, either by proving stronger lower-bounds on the approximability or devising a polynomial time approximation scheme.

References

[1] Ankit Aggarwal, Louis Anand, Manisha Bansal, Naveen Garg, Neelima Gupta, Shubham Gupta, and Surabhi Jain. A 3-approximation for facility location with uniform capacities. In *Proceedings of the 14th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 149–162, 2010.

[2] Gagan Aggarwal, Tomás Feder, Krishnaram Kenthapadi, Samir Khuller, Rina Panigrahy, Dilys Thomas, and An Zhu. Achieving anonymity via clustering. *ACM Transactions on Algorithms (TALG)*, 6(3):49:1–49:19, 2010.

[3] Gagan Aggarwal, Tomás Feder, Krishnaram Kenthapadi, Rajeev Motwani, Rina Panigrahy, Dilys Thomas, and An Zhu. Approximation algorithms for k-anonymity. *Journal of Privacy Technology*, 6(3), 2005.

[4] Sara Ahmadian, Babak Behsaz, Zachary Friggstad, Amin Jorati, Mohammad R. Salavatipour, and Chaitanya Swamy. Approximation algorithms for minimum-load k-facility location. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, volume 28, pages 17–33. 2014.

[5] Sara Ahmadian, Zachary Friggstad, and Chaitanya Swamy. Local-search based approximation algorithms for mobile facility location problems. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1607–1621, 2013.

[6] Sara Ahmadian and Chaitanya Swamy. Improved approximation guarantees for lower-bounded facility location. In *International Workshop on Approximation and Online Algorithms (WAOA)*, pages 257–271, 2012.

[7] Sara Ahmadian and Chaitanya Swamy. Approximation algorithms for clustering problems with lower bounds and outliers. In *Proceedings of the 18th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, volume 55, pages 69:1–69:15, 2016.

[8] Hyung-Chan An, Aditya Bhaskara, Chandra Chekuri, Shalmoli Gupta, Vivek Madan, and Ola Svensson. Centrality of trees for capacitated k-center. *Mathematical Programming*, 154(1-2):29–53, 2015.

[9] Hyung-Chan An, Mohit Singh, and Ola Svensson. LP-based algorithms for capacitated facility location. In *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 256–265, 2014.

[10] Esther M. Arkin, Refael Hassin, and Asaf Levin. Approximations for minimum and min-max vehicle routing problems. *Journal of Algorithms*, 59(1):1–18, 2006.

[11] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristic for k-median and facility location problems. volume 33, pages 544–562, 2004.

[12] Michel L. Balinski and Philip Wolfe. On benders decomposition and a plant location problem. *ARO-27, Mathematica*, page 45, 1963.

[13] Manisha Bansal, Naveen Garg, and Neelima Gupta. A 5-approximation for capacitated facility location. In *Proceedings of the 20th Annual European Symposium on Algorithms (ESA)*, pages 133–144. 2012.

[14] Babak Behsaz and Mohammad R. Salavatipour. On minimum sum of radii and diameters clustering. *Algorithmica*, 73(1):143–165, 2015.

[15] Jarosław Byrka. An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 29–43. 2007.

[16] Jarosław Byrka, Thomas Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for k-median, and positive correlation in budgeted optimization. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 737–756, 2015.

[17] Vasilis Capoyleas, Günter Rote, and Gerhard Woeginger. Geometric clusterings. *Journal of Algorithms*, 12(2):341–356, 1991.

[18] Moses Charikar and Sudipto Guha. Improved combinatorial algorithms for facility location problems. *SIAM Journal on Computing*, 34(4):803–824, 2005.

[19] Moses Charikar, Sudipto Guha, David B. Shmoys, and Éva Tardos. A constant-factor approximation algorithm for the k-median problem. *Journal of Computer and System Sciences*, 65(1):129–149, 2002.

[20] Moses Charikar, Samir Khuller, David M. Mount, and Giri Narasimhan. Algorithms for facility location problems with outliers. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete algorithms (SODA)*, pages 642–651, 2001.

[21] Moses Charikar and Shi Li. A dependent LP-rounding approach for the k-median problem. In *Automata, Languages, and Programming*, pages 194–205. 2012.

[22] Moses Charikar and Rina Panigrahy. Clustering to minimize the sum of cluster diameters. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 1–10, 2001.

[23] Ke Chen. A constant-factor approximation algorithm for k-median clustering with outliers. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 826–835, 2008.

[24] Fabián A. Chudak and David B. Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. *SIAM Journal on Computing*, 33(1):1–25, 2003.

[25] Fabián A. Chudak and David P. Williamson. Improved approximation algorithms for capacitated facility location problems. In *Mathematical Programming*, volume 2, pages 207–222. 2005.

[26] Marek Cygan, MohammadTaghi Hajiaghayi, and Samir Khuller. LP rounding for k-centers with non-uniform hard capacities. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 273–282, 2012.

[27] Marek Cygan and Tomasz Kociumaka. Constant-factor approximation for capacitated k-center with outliers. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 25, 2014.

[28] Erik D. Demaine, MohammadTaghi Hajiaghayi, Hamid Mahini, Amin Sayedi-Roshkhar, Shayan Oveisgharan, and Morteza Zadimoghaddam. Minimizing movement. *ACM Transactions on Algorithms (TALG)*, 5(3):30:1–30:30, 2009.

[29] Nikhil Devanur, Naveen Garg, Rohit Khandekar, Vinayaka Pandit, Amin Saberi, and Vijay V. Vazirani. Price of anarchy, locality gap, and a network service provider game. In *Internet and Network Economics*, pages 1046–1055. 2005.

[30] Srinivas R. Doddi, Madhav V. Marathe, SS Ravi, David S. Taylor, and Peter Widmayer. Approximation algorithms for clustering to minimize the sum of diameters. In *Proceedings of the 11th Scandinavian Workshop on Algorithm Theory (SWAT)*, pages 237–250. 2000.

[31] Alina Ene, Sariel Har-Peled, and Benjamin Raichel. Fast clustering with lower-bounds: No customer too far, no shop too small. *arXiv preprint arXiv:1304.7318*, 2013.

[32] Guy Even, Naveen Garg, Jochen Könemann, R. Ravi, and Amitabh Sinha. Covering graphs using trees and stars. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 24–35, 2003.

[33] Greg N. Frederickson, Matthew S. Hecht, and Chul E. Kim. Approximation algorithms for some routing problems. *SIAM Journal on Computing*, 7:178, 1978.

[34] Zachary Friggstad and Mohammad R. Salavatipour. Minimizing movement in mobile facility location problems. *ACM Transactions on Algorithms (TALG)*, 7(3):28:1–28:22, 2011.

[35] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of* NP*-completeness*. 1979.

[36] Matt Gibson, Gaurav Kanade, Eric Krohn, Imran A. Pirwani, and Kasturi Varadarajan. On metric clustering to minimize the sum of radii. *Algorithmica*, 57(3):484–498, 2010.

[37] Matt Gibson, Gaurav Kanade, Eric Krohn, Imran A. Pirwani, and Kasturi Varadarajan. On clustering to minimize the sum of radii. *SIAM Journal on Computing*, 41(1):47–60, 2012.

[38] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[39] Inge L. Gørtz and Viswanath Nagarajan. Locating depots for capacitated vehicle routing. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 230–241. 2011.

[40] Sudipto Guha and Samir Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms*, 31(1):228–248, 1999.

[41] Sudipto Guha, Adam Meyerson, and Kamesh Munagala. Hierarchical placement and network design problems. In *Proceedings of 41st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 603–612, 2000.

[42] Anupam Gupta and Kanat Tangwongsan. Simpler analyses of local search algorithms for facility location. *arXiv preprint arXiv:0809.2554*, 2008.

[43] MohammadTaghi Hajiaghayi, Rohit Khandekar, and Guy Kortsarz. Local search algorithms for the red-blue median problem. *Algorithmica*, 63(4):795–814, 2012.

[44] Russell D. Halper. On the routing and location of mobile facilities. 2010. Ph.D. thesis, University of Maryland, College Park, MD.

[45] Dorit S. Hochbaum. Heuristics for the fixed cost median problem. *Mathematical Programming*, 22(1):148–162, 1982.

[46] Dorit S. Hochbaum. When are *NP*-hard location problems easy? *Annals of Operations Research*, 1(3):201–214, 1984.

[47] Dorit S. Hochbaum and David B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985.

[48] Dorit S. Hochbaum and David B. Shmoys. A unified approach to approximation algorithms for bottleneck problems. *Journal of the ACM (JACM)*, 33(3):533–550, 1986.

[49] Dorit S. Hochbaum and David B. Shmoys. A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach. *SIAM Journal on Computing*, 17(3):539–551, 1988.

[50] Wen-Lian Hsu and George L. Nemhauser. Easy and hhard bottleneck location problems. *Discrete Applied Mathematics*, 1(3):209–215, 1979.

[51] Anil K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[52] Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *Journal of the ACM (JACM)*, 50(6):795–824, 2003.

[53] Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 731–740, 2002.

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

[55] David R. Karger and Maria Minkoff. Building steiner trees with incomplete global knowledge. In *Proceedings of 41st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 613–623, 2000.

[56] Mohammad R. Khani and Mohammad R. Salavatipour. Improved approximation algorithms for the min-max tree cover and bounded tree cover problems. *Algorithmica*, 69:443–460, 2014.

[57] Samir Khuller and Yoram J. Sussmann. The capacitated k-center problem. *SIAM Journal on Discrete Mathematics*, 13(3):403–418, 2000.

[58] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. Heidelberg: Springer-Verlag, 2008.

[59] Madhukar R. Korupolu, C Greg Plaxton, and Rajmohan Rajaraman. Analysis of a local search heuristic for facility location problems. *Journal of Algorithms*, 37(1):146–188, 2000.

[60] Ravishankar Krishnaswamy, Amit Kumar, Viswanath Nagarajan, Yogish Sabharwal, and Barna Saha. The matroid median problem. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1117–1130, 2011.

[61] Alfred A. Kuehn and Michael J. Hamburger. A heuristic program for locating warehouses. *Management Science*, 9(4):643–666, 1963.

[62] Jan K. Lenstra, David B. Shmoys, and Éva Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46(1-3):259–271, 1990.

[63] Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Information and Computation*, 222:45–58, 2013.

[64] Shi Li and Ola Svensson. Approximating k-median via pseudo-approximation. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC)*, pages 901–910, 2013.

[65] Andrew Lim, Fan Wang, and Zhou Xu. A transportation problem with minimum quantity commitment. *Transportation Science*, 40(1):117–129, 2006.

[66] Jyh-Han Lin and Jeffrey S. Vitter. Approximation algorithms for geometric median problems. *Information Processing Letters*, 44(5):245–249, 1992.

[67] Mohammad Mahdian and Martin Pál. Universal facility location. In *Proceedings of the 11th Annual European Symposium on Algorithms (ESA)*, pages 409–421. 2003.

[68] Alan S. Manne. Plant location under economies-of-scale-decentralization and computation. *Management Science*, 11(2):213–235, 1964.

[69] Pitu B. Mirchandani and Richard L. Francis, editors. *Discrete Location Theory*. Jown Wiley and Sons, 1990.

[70] Hiroshi Nagamochi and Kohei Okada. Approximating the min-max rooted-tree cover in a tree. *Information Processing Letters*, 104(5):173–178, 2007.

[71] Martin Pál, Éva Tardos, and Tom Wexler. Facility location with non-uniform hard capacities. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 329–338, 2001.

[72] R. Ravi. Workshop on flexible network design, 2012. `http://fnd2012.mimuw.edu.pl/qa/index.php,qa=4&qa_1=approximating-star-cover-problems`.

[73] Pierangela Samarati. Protecting respondents identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.

[74] David B. Shmoys. The design and analysis of approximation algorithms: Facility location as a case study. In S. Hosten, J. Lee, and R. Thomas, editors, *Trends in Optimization, AMS Proceedings of Symposia in Applied Mathematics 61*, pages 85–97. 2004.

[75] David B. Shmoys and Éva Tardos. An Approximation Algorithm for the Generalized Assignment Problem. *Mathematical Programming*, 62(3):461–474, 1993.

[76] David B. Shmoys, Éva Tardos, and Karen Aardal. Approximation algorithms for facility location problems. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC)*, pages 265–274, 1997.

[77] John F. Stollsteime. A working model for plant numbers and locations. *Journal of Farm Economics*, 45(3):631–645, 1963.

[78] Zoya Svitkina and Éva Tardos. Facility location with hierarchical facility costs. *ACM Transactions on Algorithms (TALG)*, 6(2):37:1–37:22, 2010.

[79] Chaitanya Swamy. Improved approximation algorithms for matroid and knapsack median problems and applications. In *ACM Transactions on Algorithms (TALG)*, volume 12, pages 49:1–49:22, 2016.

[80] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge university press, 2011.