APPROXIMATION ALGORITHMS FOR CLUSTERING PROBLEMS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Chaitanya Swamy

May 2004

APPROXIMATION ALGORITHMS FOR CLUSTERING PROBLEMS

Chaitanya Swamy, Ph.D.
Cornell University 2004

Clustering is a ubiquitous problem that arises in many applications in different fields such as data mining, image processing, machine learning, and bioinformatics. Clustering problems have been extensively studied as optimization problems with various objective functions in the Operations Research and Computer Science literature. We focus on a class of objective functions more commonly referred to as *facility location problems*. These problems arise in a wide range of applications such as, plant or warehouse location problems, cache placement problems, and network design problems where the costs obey economies of scale.

In the simplest of these problems, the uncapacitated facility location (UFL) problem, we want to open facilities at some subset of a given set of locations and assign each client in a given set $\mathcal{D}$ to an open facility so as to minimize the sum of the facility opening costs and client assignment costs. This is a very well-studied problem; however it fails to address many of the requirements of real applications. In this thesis we consider various problems that build upon UFL and capture additional issues that arise in applications such as, uncertainties in the data, clients with different service needs, and facilities with interconnectivity requirements. By focusing initially on facility location problems in these new models, we develop new algorithmic techniques that will find application in a wide range of settings.

We consider a widely used paradigm in stochastic programming to model settings where the underlying data, for example, the locations or demands of the clients, may be uncertain: the 2-stage with recourse model that involves making some initial decisions, observing additional information, and then augmenting the initial decisions, if necessary, by taking recourse actions. We present a randomized polynomial time

algorithm that solves a large class of 2-stage stochastic linear programs (LPs) to near-optimality with high probability. We exploit this tool to devise the first approximation algorithms for various 2-stage stochastic integer problems such as the stochastic versions of the set cover, vertex cover, and facility location problems, when the underlying random data is only given as a "black box" and no restrictions are placed on the cost structure.

We introduce the facility location with service installation costs problem to model applications involving clients with different service requirements. We abstract such settings by insisting that a client may only be assigned to a facility if the service requested by it has been installed at the facility (incurring a service installation cost). The connected facility location problem captures settings where the open facilities want to communicate with each other or with a central authority; we model this by requiring that the open facilities be interconnected by a Steiner tree. We give intuitive and efficient algorithms for both these problems. We use these algorithms to obtain approximation algorithms for the k-median variants of these problems, where in addition to all of the constraints of the problem, a bound of k is imposed on the number of facilities that may be opened.

Biographical Sketch

Chaitanya Swamy was born on January 4, 1978, in Delhi, India. He received a B. Tech. in Computer Science in May 1999 from the Indian Institute of Technology, Delhi, where he realized that sitting and thinking about problems suited him just fine, and joined Cornell University in August 1999 with the intent of doing research in theoretical computer science. Five years hence, he expects to receive a Ph.D. in Computer Science from Cornell University in May 2004, and along the way he obtained an M.S. in Computer Science from Cornell in May 2003.

Acknowledgements

First and foremost, I thank my advisor David Shmoys. I am deeply indebted to David for his generous and invaluable advice on various matters, be it writing papers, giving talks, preparing slides, interviewing, looking for jobs, or other non-research matters. David's advice over the years has benefited me immensely, and will always stand me in good stead. It has been a pleasure working together and almost all the work in this thesis is joint work with him. More personally, his constant encouragement and belief in me were vital in the advancement of my graduate career. Thank You.

I thank Éva Tardos for several helpful discussions on both technical and non-technical matters. Éva has always shown an interest in my work, and like David, she has played a significant role in shaping my views on research.

I thank Mike Todd for serving on my committee and for his careful reading of this thesis. Mike's detailed comments and suggestions helped to improve the presentation and clarify the arguments in several contexts. I have benefited greatly from Mike's keen technical insights, in particular, on convex optimization, and I am grateful to him for enlightening me about the topic by patiently answering my numerous questions, which played a crucial role in the development of the results on stochastic optimization detailed in Part II of this thesis.

I thank Ramin Zabih for serving on my committee, and David Williamson for agreeing to act as a proxy for Ramin for my defense. Although David has been at Cornell only during my last semester here, he was always available whenever I needed to give a practice talk and his feedback has been very helpful.

Life at Cornell would have been less interesting without the discussions of the

Theory Group. In particular, I thank Aaron Archer, Ara Hayrapetyan, Amit Kumar, Retsef Levi, Martin Pál, and Tom Wexler for various interesting and useful discussions.

Going back a bit further in time, I thank Naveen Garg, my undergraduate advisor at the Indian Institute of Technology, Delhi, who got me excited about algorithms and research in theory.

Finally and most importantly, I thank my parents and my sister for their love, and constant support and encouragement. I am grateful to my parents for proofreading portions of this thesis, and to my mother for putting up with my long hours of work away from home in the days leading up to the thesis submission deadline and having to spend time alone, despite having traveled all the way from India to spend some time with me.

This work was supported in part by NSF grant CCR-9912422.

Table of Contents

List of Tables

List of Figures

Part I

Overview and Preliminaries

Chapter 1

Introduction

Clustering is a ubiquitous problem that arises in many applications in different fields such as data mining, image processing, machine learning, and bioinformatics. At a high level, a clustering problem seeks to group members of the data set so that, under some definition of similarity, similar members are grouped together and dissimilar members are not grouped together. The widespread use of clustering as a fundamental data analysis tool stems from the fact that, grouping the data into manageable chunks allows one to reason more effectively about the data; for example, one might be able to infer trends and patterns from the data, make reasonable predictions about unseen data, or simply manipulate the underlying data more efficiently. Clustering problems have been extensively studied as optimization problems under various objective functions in the Operations Research and Computer Science literature. Many of these optimization problems turn out to be computationally intractable, so a reasonable approach is to settle for approximation algorithms, that is, algorithms that run in polynomial time and always deliver provably near-optimal feasible solutions. A *ρ-approximation algorithm* is a polynomial-time algorithm that always returns a feasible solution with objective function value within a factor of ρ of the optimum; ρ is called the *approximation ratio* or *performance guarantee* of the algorithm.

In this thesis we focus on a class of clustering problems more commonly referred to as *facility location problems*, that arose originally in the context of warehouse location problems with the objective of locating warehouses and clustering the client locations (points) around these warehouses (centers). In the simplest of these problems, the *uncapacitated facility location* (UFL) problem, we are given a set of locations $\mathcal{F}$ at which facilities may be opened, and a set of clients $\mathcal{D}$ that need to be serviced by facilities. We want to open facilities at a subset of the locations in $\mathcal{F}$ and assign each client in $\mathcal{D}$ to an open facility. Opening a facility incurs a location-dependent *facility opening cost*, and assigning a client to an open facility incurs a *client assignment cost* proportional to the distance between the facility and client locations, and the aim is to minimize the sum of the facility opening costs and the client assignment costs. Unless otherwise stated, we will always assume that the distances form a *metric* (this is also sometimes referred to as metric UFL), that is, they are symmetric and satisfy the triangle inequality.

The uncapacitated facility location problem is one of the most widely studied problems in the Operations Research literature, and, in the past few years, has been a subject of active research in the Computer Science literature as well. This problem is rich in terms of both applications, and the algorithmic techniques it showcases. Shmoys, Tardos & Aardal [71] gave the first constant-factor approximation algorithm for this problem based on rounding a linear programming (LP) relaxation of this problem, and a variety of approaches have been subsequently used to obtain very effective approximation algorithms, such as the primal-dual method, local search heuristics, and greedy algorithms. UFL finds use in a gamut of applications ranging from classical applications such as modeling plant or warehouse location problems, to more recent applications such as modeling data management problems dealing with the placement of caches in a network, and its use as an important subroutine to solve so-called buy-at-bulk network design problems that involve agglomeration of traffic in the presence of costs that obey economies of scale, and various clustering problems such as the classical k-median problem and hierarchical clustering problems.

However, in many respects, UFL remains a "toy" problem that is a rather simple abstraction of real applications capturing only the very basic ingredients of the problem. (In fact, the simplicity of this model is highlighted by the fact that it is also referred to as the *simple* warehouse location problem in the Operations Research literature.) The study of the uncapacitated facility location problem has served as a useful first step towards attacking real problems, in that it has led to the development of a wide variety of techniques, and now there is a need to build upon these and devise new methods, or enhance existing techniques, to attack algorithmic problems in models that better abstract real problems.

Motivated by this goal, in this thesis, we consider various models that build upon UFL by incorporating additional issues that arise in applications such as, uncertainties in the data, clients with different service needs, and facilities with interconnectivity requirements. Our contributions are twofold: we devise techniques to tackle facility location problems that arise in these new models that should also prove useful in attacking problems outside of the domain of facility location problems, and in doing so, we obtain insights on why some techniques work well on some problems and others do not.

1.1 A Motivating Example

Consider the following caching/data management application, that we will use to motivate many of the models and problems investigated in this thesis. We are given a distributed network where the nodes may represent servers, workstations, individual PCs. Some of these nodes have processes running on them that periodically issue requests for data items that may be stored on other nodes of the network, and to satisfy such a request we have to fetch the data item from such a remote node incurring a certain latency of access. Further, the nodes in the network have some storage space, that may vary across different nodes, that could be used to set up a cache and store some of these data items so as to reduce the latency of data item requests. Setting up a cache incurs a certain fixed cost, for example, because one may need to expunge some data currently residing in the main memory to create the cache. The goal is to decide where to set up caches and how to assign the data requests to the caches, so as to minimize the total cost of setting up the caches and the latency of accesses.

The above description of the problem fits nicely in the mold of the uncapacitated facility location problem, and a first-cut approach might therefore be to model this problem as a UFL instance where the facilities are caches, the clients correspond to data requests, and the assignment cost of a client is the latency of access of the data request. While the simplicity of this model is appealing, it fails to capture many of the aspects of the above problem, some more glaring than others.

1.2 Models Considered in this Thesis

Facility Location with Services. In the caching application, each data request is for a *specific* data item, and should therefore be assigned only to a cache that contains that data item; however the UFL model ignores this data-specificity and assumes that a data request can be assigned to any cache. Converting the UFL solution into a feasible solution to our problem, by storing in a cache all the data items corresponding to the requests that are assigned to it, might produce very cost-ineffective sub-optimal solutions since there could be costs associated with storing data items in a cache (in addition to the cache setup cost), e.g., the latency to fetch data items initially from their remote locations, and the UFL model completely ignores these costs. In particular, such a solution might require one to store *every* data item in *each* cache, decidedly an overkill. We abstract settings such as the above caching application, by introducing the *facility location with service installation costs* problem, where we have a set of *services* (data items) in addition to clients and facilities. Each client (data request) requires a specific service, and to satisfy a client, we must assign it to an open facility (cache) at which that service has been installed by paying a certain *service installation cost*, that is, in the above setting, a cache may satisfy a data request only if it contains the requested data item, and there is a cost incurred in storing a given data item.

Stochastic Facility Location Problems. In UFL, one assumes that the client demands are known precisely in advance. Often in real applications, there are uncertainties in the data because the data corresponds to information that is revealed only in the future, or it is very difficult to model the data very precisely, or there may be some inherent fluctuation in the data due to noise, etc. One approach to dealing with uncertainty that has been extensively studied under the paradigm of stochastic optimization, is to assume that the uncertainty can be modeled by a probability distribution. In the case of facility location, this means that instead of exact information about the client demands, one is only given distributional information about the demand, which could have been obtained through market surveys, customer profiles, in the context of data management/caching applications from traces of data item requests, or through any other possible means. Based on this (inexact) information one has to decide which facilities to open; later, once we learn the actual demands, one still has the opportunity to open additional facilities to handle extra demand if necessary, but opening a facility at the "last minute" would typically incur a higher cost as compared to the initial opening cost.

For example, in the caching application above, the network designer may decide on an initial placement of items in caches by carefully taking into account the available storage, the amount of extra storage space that has to be freed to set up caches to store data items, and balancing the cost of doing so against the estimated demand for data items. But later it may turn out that there is an unexpected increase in the demand for some data items, and with the current placement, these data item requests incur a large latency of access because the data items are stored in remote caches. In an effort to improve performance, the designer may therefore choose to set up a more "local" cache and/or store these data items in an existing local cache. This would incur some overhead, since one may have to create some storage space by relocating the data items and the associated data requests that are currently using this storage, which results in an increased cost to set up a cache and/or store items in a cache.

These considerations motivate the 2-stage stochastic versions of UFL and the facility location with service installation costs problem, that lie in the genre of 2-stage stochastic optimization with recourse problems: first, one must commit on some initial (first-stage) decisions given only distributional information about (some of) the data, and then once the actual data is realized (according to the distribution), further (recourse) actions can be taken (second-stage), and the goal is to minimize the total cost of the first-stage decisions and the expected cost of the second-stage decisions. We study this general class of problems in Part II, and use the algorithm developed therein for solving a fractional relaxation of these problems, to design algorithms for some 2-stage stochastic facility location problems in Chapter 4 and Section 5.6 of Part III.

Facilities with Communication Requirements. The above model forms a good abstraction of the caching problem if there are only read-requests. Write-requests add a degree of complication, because in order to maintain consistency of data, one has to ensure that a write-request on an item updates all copies of the item. Thus, to handle write-requests, it is necessary for the caches that contain the same data item to be able to communicate with each other or with a common central authority. One way to model this, is to insist that the caches be interconnected via a Steiner tree, which could serve as a multicast tree used to update all copies of a data object upon a write on that object (this model was actually proposed by [48]). This motivates the *connected facility location* problem, where we capture settings in which the open facilities want to communicate with each other or with a central authority, by insisting that the open facilities be interconnected by a Steiner tree. This problem arises in diverse settings. It deals with the design of a two-layered solution where the demand points are first clustered around hubs and the hubs are then interconnected; this kind of clustering problem is used as a building block in the so-called *buy-at-bulk network design* problems, where one wants to design a multi-layered solution and the costs in the different layers obey economies of scale, so one seeks to agglomerate data at each layer before moving to the next layer.

***k*-Median Problems.** In many settings there may be a bound imposed on the number of facilities that may be opened, in place of, or in addition to, the facility costs. For example, the facility opening cost might consist of a long-term running cost and a short-term opening cost, and we may want to minimize the long-term running costs of the facilities and the client assignment costs, subject to a budget constraint on the short-term costs. If the short-term costs of the facilities are comparable to each other, then this translates to a cardinality bound on the number of facilities that may be opened. This gives rise to the k-median version of the facility location problem, where in addition to all the constraints of the original facility location problem, there is an extra constraint that limits the number of facilities that may be opened to at most k. k-median problems also arise as natural clustering problems. For example, the k-median version of UFL with no facility opening costs and where one may open a facility at any location is the classical k-median clustering problem: given a set of points (clients) in a metric space, we want to choose k of these as medians (facilities) and assign each point to a median so as to minimize the sum of the distances from each point to its assigned median. Viewed from this perspective, the k-median version of the facility location with service installation costs problem captures a clustering objective where points representing data objects with multiple attributes can be assigned to multiple clusters. We are given points in a metric space with each point being described by a set of attributes (services); we have to choose k points as medians, allot each median a set of attributes, and assign each attribute of each point to a median to which that attribute is allotted; the quality of the clustering is measured by the total number of allotted attributes and the sum of the distances from each point-attribute to its assigned median.

1.3 A Primer on Linear Programming and Approximation Algorithms

In order to describe our results in a more meaningful way, we give some background on linear programming and its use in the design and analysis of approximation algorithms.

Over the past several years there has been a tremendous advancement in our understanding of general principles for the design and analysis of approximation algorithms for *NP*-hard problems. One technique that has proved to be quite successful in the design of approximation algorithms is that of expressing a relaxation of the discrete optimization problem as a linear program (LP), and either explicitly solving the relaxation and converting the optimal LP solution to a feasible solution to the original problem, or using the relaxation only implicitly to guide the design and analysis of the algorithm. In many cases, the discrete optimization problem can be formulated as a polynomial size *integer program* with a linear objective function and linear constraints (other than the constraints that force the variables to take on integer values), and a simple way of obtaining such a relaxation is to drop the integrality constraints and replace them by weaker constraints, such as just non-negativity constraints. Dropping the integrality constraints this way gives a *linear programming relaxation* for the original problem, and since one has only relaxed the constraints, the optimal value of this linear program provides a lower bound (in case of a minimization problem) on the optimal value of the integer program which is equal to the value of the optimal solution to the original problem.

Algorithms that explicitly solve the LP relaxation are often called LP-rounding algorithms because they are based on "rounding" the (possibly) fractional LP solution to an integer solution. The best known examples of algorithms where the LP relaxation is used only implicitly are algorithms based on the so-called primal-dual schema, that are often called primal-dual algorithms. These algorithms exploit the fact that any *primal* minimization linear program, has a corresponding *dual* maximization linear program such that the value of any feasible solution to the dual problem is a *lower bound* on the value of the optimal primal LP solution, a fact known as *weak duality*, and hence on the value of the optimal solution to the original problem. The basic mechanism of the primal-dual method involves constructing simultaneously an (integer) primal solution and a feasible dual solution, and then arguing that the cost of the primal solution constructed is within some factor ρ of the value of the dual solution constructed; consequently, one obtains a ρ-approximation algorithm. Thus, in a primal-dual algorithm, the LP is not explicitly solved but is only used as an aid in algorithm design and analysis.

1.4 Our Contributions

1.4.1 Stochastic Optimization

Solving 2-stage stochastic linear programs. Stochastic optimization problems attempt to model uncertainty in the data by assuming that (part of) the input is specified in terms of a probability distribution, rather than by deterministic data given in advance. In Part II, we study an important subclass of stochastic optimization problems, namely *2-stage stochastic problems with recourse*: given only a probability distribution about (some of) the data, one has to commit to some first-stage decisions, but then once the actual input is realized according to this distribution, one can extend the solution in the second stage by taking some *recourse actions*, where these recourse costs might be different (and typically greater) than the original ones. Two-stage stochastic problems with recourse are often computationally quite difficult, both from a practical perspective, and from the point of view of complexity theory. This is mainly due to the fact that the distribution might assign a non-zero probability to an exponential number of scenarios, leading to a considerable increase in the problem complexity, a phenomenon often phrased as "the curse of dimensionality."

In Chapter 3, we give an algorithm to solve a large class of 2-stage stochastic *linear programs* to near-optimality. We present a randomized algorithm that, with high probability, returns a solution of objective function value at most $(1+\epsilon)$ times the optimum, where ϵ can be set arbitrarily close to 0, in time that is polynomial in the input size and $\frac{1}{\epsilon}$. Thus, we obtain a *fully polynomial randomized approximation scheme* (FPRAS) for a rich class of 2-stage stochastic linear programs. Two-stage stochastic linear programs have been extensively studied in the Operations Research literature, in particular, in the stochastic programming literature; to the best of our knowledge however, this is the first result that demonstrates the polynomial-time solvability of (a class of) 2-stage stochastic LPs. The algorithm works for both discrete and continuous distributions, and functions in the so-called "black-box" model, where one is given only a black box that one can use to draw independent samples from the underlying probability distribution. Thus, the algorithm we present does not require any assumptions about the probability distribution (or the cost structure of the input).

Our result should be viewed as indicative of the fact that, contrary to conventional wisdom, an exponential number of scenarios is *not* an insurmountable impediment to the design of efficient algorithms for these problems. (For example, in [12], this commonly held view was used as a motivation for considering so-called "robust" versions of deterministic optimization problems, as opposed to their stochastic versions.) Whereas a naive LP formulation of a 2-stage problem might involve an exponential number of both variables and constraints due to the exponential number of scenarios to consider, we show that one can still approximately solve this linear program by working with an equivalent (but compact) convex programming formulation whose variables are just the first-stage decision variables, and show that the ellipsoid algorithm can be adapted to yield such a scheme[1]. In doing so, a significant difficulty that we must overcome is that even evaluating the objective function of this convex program may be $\#P$-hard, in general.

We believe that our study of 2-stage stochastic linear programs is an important

[1]Thus when we say "solving the 2-stage stochastic linear program," we mean to say that we are solving the equivalent convex program. Throughout this thesis, we use the phrase with this intended meaning.

step that could lead to both better computational procedures to solve 2-stage stochastic LPs, as well as algorithms with provable guarantees for more general stochastic optimization models such as multi-stage problems or chance-constrained problems (the book by Birge & Louveaux [13] contains a discussion of these and some other alternate models).

Approximation algorithms for 2-stage stochastic problems. Many applications involving uncertain data can be abstracted by the 2-stage recourse model, such as for example, the problem of installing facilities to serve a set of clients, that is, the 2-stage stochastic UFL problem; given only a probability distribution on the set of clients that need to be served, one must decide where to open facilities initially, but then later, once the actual input is realized according to the distribution, one may choose to open additional facilities incurring recourse costs.

The ability to solve 2-stage stochastic linear programs to near-optimality provides us with a powerful and versatile tool that we exploit to design approximation algorithms for 2-stage stochastic integer optimization problems in a manner analogous to the way in which linear programming has been (very successfully) exploited to design and analyze approximation algorithms for deterministic integer optimization problems. A 2-stage integer optimization problem can be formulated as a 2-stage stochastic integer program (IP), and by relaxing the integrality constraints one obtains a 2-stage stochastic linear program. Using our algorithm to solve 2-stage stochastic linear programs, one can obtain a near-optimal solution to the 2-stage stochastic LP; the next step is to round this solution to get an integer solution without blowing up the objective function value by much.

We sketch a general framework for rounding this (near-) optimal solution, that proceeds by "decoupling" stage I and the different stage II scenarios, which then allows one to lift existing algorithms and guarantees for the deterministic optimization problem to the stochastic setting and thereby obtain an approximation algorithm for the 2-stage stochastic optimization problem. Thus in some sense, we give a way to "reduce" the stochastic optimization problem to its deterministic counterpart, that allows one to use the machinery developed for the deterministic problem to attack the stochastic problem. We illustrate this methodology by applying it to the 2-stage stochastic set-cover and uncapacitated facility location problems in Chapter 4. The rounding procedure for the stochastic UFL problem forms the basis for rounding the other 2-stage facility location problems that we consider in Section 4.4 and Section 5.6.

The results in Chapters 3 and 4 were obtained in joint work with David Shmoys [69].

1.4.2 Deterministic Facility Location Problems

In Chapter 5, we consider the facility location with service installation costs problem that abstracts the caching application mentioned earlier. We give a constant-factor approximation algorithm for this problem based on the primal-dual schema, under a certain assumption on the service installation costs: we assume that the locations in $\mathcal{F}$ can be sorted, so that if i comes before i' in this ordering, then for *every* service, the cost of installing the service at i is at most the cost if installing that service at i'. This assumption is general enough to capture settings where the service installation cost may depend only on the location, or only on the service. In the latter special case, we also give an algorithm based on rounding an LP formulation of the problem that attains a better approximation ratio. The results in this chapter, except for the work in Section 5.6 where we look at the 2-stage stochastic version of the problem, were jointly obtained with David Shmoys and Retsef Levi [70].

Chapter 6 focuses on the connected facility location problem which abstracts settings where the open facilities want to communicate with each other or with a common central authority. We model this by requiring that the open facilities be interconnected via a Steiner tree, and the cost incurred is the sum of the facility opening costs, client assignment costs, and the length of the tree scaled by an input parameter $M \geq 1$. The parameter M represents the higher cost of interconnecting the facilities, e.g., connecting the core nodes in a telecommunication network via high bandwidth links. A special case of this problem, called the *rent-or-buy* problem arises if facilities can be opened at any location and there are no facility opening costs. We give primal-dual algorithms for these problems that attain approximation ratios of 8.55 for connected facility location, and 4.55 for the rent-or-buy problem. Our algorithms integrate the primal-dual approaches for the facility location and Steiner tree problems, yet are simple and intuitive, and easy to analyze; the previous best guarantees were obtained by rounding the optimal solution to an LP relaxation, and were not combinatorial. The results here are from [75] and represent joint work with Amit Kumar.

Finally, in Chapter 7 we consider the k-median clustering variants of the above problems, where in addition to facility costs, we impose a bound of k on the number of facilities that may be opened. We show that the primal-dual algorithms developed for the facility location with service installation costs, and the connected facility location problems are versatile, and can be used to obtain approximation algorithms for the k-median variants of the respective problems as well.

1.5 Related Work

In this section we describe some previous work on the uncapacitated facility location problem and stochastic optimization, to place our contributions in context. We confine ourselves to a high level overview; a more detailed review of work related to the contents of a chapter appears at the beginning of each chapter.

1.5.1 Uncapacitated Facility Location

The uncapacitated facility location (UFL) problem is one of the few problems that showcases all the well-known algorithmic techniques in approximation algorithms design, such as LP rounding, primal-dual algorithms, local search heuristics and greedy methods. It is also a shining example of the power and versatility of linear programming based methods in the design of approximation algorithms. There is a great deal of literature on UFL that considers the problem from perspectives such as probabilistic analysis of average case performance, polyhedral characterizations, empirical investigation of heuristics, which we do not mention here; see [58] for an extensive survey of work on UFL. We limit ourselves to a review of the literature that deals with the design of approximation algorithms for UFL.

The non-metric case of UFL, that is, where the distances need not be symmetric and need not satisfy the triangle inequality, is known to be as hard as the set-cover problem. Combined with the results of Raz & Safra [64] and Feige [25], this shows that for some constant $c < 1$, non-metric UFL cannot be approximated to within a factor better than $c \ln |\mathcal{D}|$ unless P=NP [64], or to a factor better than $\ln |\mathcal{D}|$ unless $NP \subseteq DTIME[n^{O(\log \log n)}]$ [25]. Complementing this, Hochbaum [38] gave an $O(\ln |\mathcal{D}|)$-approximation algorithm, and Lin & Vitter [54] gave another $O(\ln |\mathcal{D}|)$-approximation algorithm based on LP rounding using their filtering technique.

Much of the work on approximating UFL has therefore concentrated on the metric version of UFL. The metric version of UFL is also *NP*-hard. Shmoys, Tardos & Aardal [71] gave the first constant-factor approximation algorithm for this problem. They gave a 3.16-approximation algorithm for UFL; in contrast, Guha & Khuller [29] showed that one cannot get a constant-factor better than 1.463 in polynomial time unless P=NP. The work of [71] led to a flurry of activity on UFL in the Computer Science literature, that has resulted in the development of a variety of techniques for approximating UFL. We build upon some of these techniques in several ways to tackle the algorithmic problems that arise in the enhanced models that we consider. The techniques that have been developed can be broadly classified into three categories.

LP-rounding algorithms. The 3.16-approximation algorithm of [71] is based on rounding a classical LP relaxation of this problem due to Balinski [9] using an elegant filtering technique due to Lin & Vitter [54]. After an improvement due to Guha & Khuller [29], Chudak & Shmoys [18] further strengthened the LP rounding approach, and improved the ratio to $\left(1 + \frac{2}{e}\right)$ by exploiting the structure in the *optimal* primal and dual solutions resulting from complementary slackness. Their algorithm combines randomized rounding [59] and the decomposition technique of [71] to get a variant that might be called *clustered randomized rounding*. More recently Sviridenko [73] gave a 1.58-approximation algorithm by combining some of the above ideas with the so-called *pipage rounding* technique [1].

Primal-dual algorithms and greedy dual-fitting based algorithms. Jain & Vazirani [41] gave an elegant primal-dual 3-approximation algorithm for UFL, where the LP is used only in the analysis, and also showed that this primal-dual framework can be extended to devise algorithms for some other variants of UFL. Very recently, Jain, Mahdian, Markakis, Saberi & Vazirani [40] gave a set-cover style greedy algorithm for UFL, that can also be viewed as a primal-dual algorithm, with a performance guarantee of 1.61. Their algorithm constructs simultaneously a primal solution and a dual solution, and the analysis uses the dual LP to lower bound the cost of the optimal solution, so in these respects the algorithm and its analysis fall into the primal-dual schema. An aspect in which it differs from the usual primal-dual setup, is that the algorithm does not construct a feasible dual solution, but instead ensures that the cost of the primal solution is equal to the value of the constructed dual. The performance guarantee hinges on showing that the infeasibility in the dual is bounded by a small factor $\rho \geq 1$, that is, showing that the dual variables can be scaled down by a factor ρ to obtain a feasible dual solution, and thus proving that the algorithm is a ρ-approximation algorithm. This style of algorithm design and analysis is sometimes called the *dual fitting* approach.

The current best algorithm for UFL is a 1.52-approximation algorithm due to Mahdian, Ye & Zhang [56], that combines the Jain et al. algorithm with a greedy local improvement step due to Guha & Khuller [29].

Local search heuristics. Another technique that has been successfully used for UFL is *local search*. Korupolu, Plaxton & Rajaraman [47] gave a simple local search procedure and showed that the cost of any locally optimal solution is within a factor of 5 of the globally optimal solution. Subsequently, Charikar & Guha [15], and later

Arya, Garg, Khandekar, Meyerson, Munagala & Pandit [7] gave more sophisticated local search heuristics that improved the approximation ratio to 3.

1.5.2 Stochastic Optimization

The field of stochastic optimization, also called stochastic programming, has its roots way back in the 1950s in the work of Dantzig [21] and Beale [11], and has since grown into a tremendous field with applications in a variety of areas, such as logistics, transportation models, inventory planning, financial instruments, and network design. The 2-stage with recourse model is a well-studied paradigm in stochastic optimization, but relatively little is known about polynomial-time algorithms for 2-stage problems that deliver provably near-optimal solutions to the 2-stage stochastic integer or linear program. There is an abundance of literature that deals with computational aspects of solving 2-stage stochastic linear programs, and in some cases provides theoretical asymptotic guarantees. One approach taken is to sample a certain number of times from the distribution on scenarios, approximate the probability of a scenario by its frequency, and solve the 2-stage problem for this approximate distribution. While there are some results that prove asymptotic convergence to the optimal solution in the limit *as the number of samples goes to infinity*, fewer results are known about the rate of convergence to a near-optimal solution and the sample size required to obtain a near-optimal solution. The work that seems to come closest in this respect is a paper by Kleywegt, Shapiro & Homem-De-Mello [46] (see also Shapiro [67]), which proves a bound on the sample size that depends on the variance of a certain quantity (calculated using the scenario distribution) that might not vary polynomially with the input size. To the best of our knowledge our result showing the polynomial time convergence of our algorithm to a near-optimal solution is new, and gives the first FPRAS for a large class of 2-stage stochastic linear programs.

The area of worst-case performance analysis of approximation algorithms for 2-stage stochastic integer programming problems is an even less explored area. The first such approximation result for discrete 2-stage problems with recourse appears to be due to Dye, Stougie & Tomasgard [22], who consider a resource provisioning problem in the setting where the uncertainty in the data is limited to a polynomial number of scenarios. Subsequently, a sequence of three papers in the Computer Science literature — Ravi & Sinha [61]; Immorlica, Karger, Minkoff & Mirrokni [39]; and Gupta, Pál, Ravi & Sinha [35] — have considered 2-stage stochastic versions of some other (integer) combinatorial optimization problems. All three papers consider models where there are restrictions imposed, either limiting the class of probability distributions, or on the cost structure of the two stages.

Thus, our work leads to the first approximation algorithms for various discrete 2-stage stochastic problems such as the stochastic versions of the set cover, vertex cover, and facility location problems (and some variants), and the other problems considered in Shmoys & Swamy [69]. Our framework of first writing the LP relaxation of the 2-stage stochastic problem, then solving this LP using the algorithm for 2-stage stochastic linear programming, and then rounding the near-optimal solution obtained using known algorithms for the deterministic problem, appears to be quite general and should find applications in devising approximation algorithms for various other 2-stage optimization problems as well.

1.6 Guide for Reading this Thesis

Chapter 2 is devoted to uncapacitated facility location and is a prerequisite for all of Part III of this Thesis. It describes three algorithms for uncapacitated facility location; in subsequent chapters in Part III, we either use these algorithms directly as black boxes, or incorporate ideas from these algorithms to design efficient solutions to our problems. We encourage even the reader who is familiar with the uncapacitated facility location problem to skim through this chapter to become familiar with some basic notation and conventions that are followed in the rest of the Thesis. Part II consists only of Chapter 3 and can be read independently of Chapter 2. The chapters in Part III are mostly independent of each other with the exception of Section 5.6 that uses ideas from the rounding algorithm for stochastic uncapacitated facility location described in Chapter 4, and Chapter 7 where we devise algorithms for the k-median versions of the facility location problems considered earlier by building upon the primal-dual algorithms developed in Chapters 2, 5 and 6. Finally, Chapter 4 and Section 5.6 use the main result of Chapter 3 (Theorem 3.5.4), but are otherwise independent of Chapter 3.

Chapter 2

Uncapacitated Facility Location

In this chapter we focus on the classical uncapacitated facility location (UFL) problem and describe three algorithms for this problem — the primal-dual algorithm due to Jain & Vazirani [41], an algorithm of Chudak & Shmoys [18] based on LP rounding, and a hybrid LP rounding algorithm that combines the filtering-based algorithm of Shmoys, Tardos & Aardal [71] and the Chudak-Shmoys rounding procedure. These algorithms introduce several ideas that we build on and adapt in subsequent chapters in Part III to devise algorithms for various facility location problems that arise in enhanced models and generalize UFL. This chapter therefore serves as a prerequisite for all the chapters in Part III of this thesis.

These algorithms rely on a natural linear programming formulation of UFL and the Jain-Vazirani (JV) and Chudak-Shmoys (CS) algorithms also use the dual of this linear program. The JV algorithm uses these linear programs only in the analysis and not in the specification of the algorithm, whereas the CS algorithm is based on first solving the linear program, and then rounding the optimal solution to get a near-optimal integer solution. We first describe the JV algorithm in Section 2.2, and then in Section 2.3 the LP rounding algorithm given by Chudak & Shmoys. In Section 2.4, we describe the hybrid algorithm. This algorithm has two useful properties. First, the algorithm rounds *any primal fractional solution* to an integer solution incurring a constant factor blowup in the cost, and second, the rounding

Table 2.1: Dependence between sections of this chapter and portions of the thesis.

Section	**Prerequisite for**
2.1	Part III
2.2	Section 5.4, Chapter 6, Section 7.2
2.3	Sections 2.4, 5.5
2.4	Chapter 4, Section 5.6

procedure *does not depend on the actual client demands*. We exploit these properties in Chapter 4 and Section 5.6 to devise algorithms for some stochastic facility location problems. Table 2.1 shows the dependence between sections of this chapter and different portions of the thesis.

2.1 Problem Definition and an LP Relaxation

In the uncapacitated facility location problem, given a set of candidate locations $\mathcal{F}$ at which *facilities* may be opened and a set of *clients* $\mathcal{D}$ that need to be assigned to facilities, we want to open facilities at some subset of the locations in $\mathcal{F}$ and assign each client to an open facility. Opening a facility at location i incurs a *facility opening cost* of f_i, and assigning a client at location j to a facility at location i incurs an *assignment cost* equal to the distance c_{ij}, and the goal is to minimize the total facility opening and client assignment costs. In certain settings a client at location j may have some *demand* or *weight* d_j, and the cost of assigning it to a facility at location i is given by the weighted distance $d_j c_{ij}$. Throughout this thesis, we consider the setting where the distances c_{ij} form a *metric*, that is, they are symmetric and satisfy the triangle inequality. In the sequel we will use the term “facility i” (respectively “client j”) to denote a facility at location i (respectively client at location j). We use the terms assignment cost and service cost, and the terms client and demand (which refers to a client j along with its weight d_j) interchangeably.

We can formulate UFL as a mathematical program involving two types of decision variables, x_{ij} and y_i, where i indexes the facilities in $\mathcal{F}$ and j indexes the clients in $\mathcal{D}$. Variable y_i indicates if facility i is open, and x_{ij} indicates if client j is assigned to facility i.

$$\begin{aligned}
\min \quad & \sum_i f_i y_i + \sum_j d_j \sum_i c_{ij} x_{ij} && && \text{(UFL-P)} \\
\text{s.t.} \quad & \sum_i x_{ij} \geq 1 && \text{for all } j, && (1) \\
& x_{ij} \leq y_i && \text{for all } i, j, && (2) \\
& x_{ij}, y_i \geq 0 && \text{for all } i, j.
\end{aligned}$$

Constraints (1) state that each client has to be assigned to a facility, and (2) ensure that if a client is assigned to facility i, then facility i is open. If we require that the x_{ij} and y_i variables take on $\{0, 1\}$ values, then we get an exact formulation of UFL. Dropping the integrality constraints gives us a linear program which one can solve efficiently to get an optimal fractional solution. The dual of the above linear program is the following maximization problem.

$$\begin{aligned}
\max \quad & \sum_j \alpha_j && && \text{(UFL-D)} \\
\text{s.t.} \quad & \alpha_j \leq d_j c_{ij} + \beta_{ij} && \text{for all } i, j, && (3) \\
& \sum_j \beta_{ij} \leq f_i && \text{for all } i, && (4) \\
& \alpha_j, \beta_{ij} \geq 0 && \text{for all } i, j.
\end{aligned}$$

The dual can be motivated as follows. The dual problem gives a way of obtaining a lower bound on the value of (UFL-P). We interpret α_j as the amount that j is willing to spend to get itself assigned to a facility. Suppose first that there are no facility opening costs, that is, $f_i = 0$ for each i. Then each client simply gets assigned to the facility closest to it, and we obtain a lower bound of $\sum_j d_j \min_i c_{ij}$. This bound is also trivially valid even if the facility costs are non-negative, but we can get a much better lower bound. Suppose that each facility i decides to divide up its cost f_i among the different clients charging an amount β_{ij} from client j to "use" facility

i. So to get assigned to i, client j must now pay both the assignment cost $d_j c_{ij}$ and this additional cost β_{ij}, or a net amount of $d_j c_{ij} + \beta_{ij}$. With this cost-sharing scheme, the facility costs are now again effectively zero, and so each client j pays an amount $\alpha_j = \min_i(d_j c_{ij} + \beta_{ij})$, as encoded by constraint (3). Since any such cost-sharing scheme gives a lower bound, to obtain the best lower bound, we want to maximize over all valid cost sharings, which is precisely the dual maximization program.

For simplicity, we will assume from now on that each client has unit demand, i.e., $d_j = 1$ for all j. We can reduce the arbitrary demand case to the unit demand case as follows. Assuming rational demands d_j (irrational demands can be approximated to an arbitrary precision by rational demands), first we rescale so that each d_j is an integer (the facility cost f_i also gets rescaled). Now for every client j with integer demand d_j, we create d_j clients co-located at location j, each having unit demand. Any solution to the modified instance yields a solution to the original instance with the same cost (ignoring the scaling factor) and vice versa. However, this reduction makes the algorithm run in only pseudo-polynomial time. Nevertheless, we can simulate this reduction by always treating the d_j co-located copies identically. This requires only cosmetic changes to the algorithms and the analyses presented.

2.1.1 Complementary Slackness

Let (x, y) and (α, β) be the optimal primal and dual solutions, respectively. Using linear programming theory, we get, as a useful consequence of optimality, that these solutions satisfy the following *complementary slackness* conditions.

Primal Slackness Conditions

$$y_i > 0 \implies \sum_j \beta_{ij} = f_i$$

$$x_{ij} > 0 \implies \alpha_j = c_{ij} + \beta_{ij}$$

Dual Slackness Conditions

$$\alpha_j > 0 \implies \sum_i x_{ij} = 1$$

$$\beta_{ij} > 0 \implies x_{ij} = y_i$$

Our earlier interpretation of the dual program gives some insight about why these conditions should hold at optimality. For example, the primal slackness conditions state that (i) if a facility i is open at all, then the payments it receives from the clients should be able to recover its facility opening cost, i.e., $\sum_{ij} \beta_{ij} = f_i$, and (ii) if a client j is using a facility i, then it has to pay the full (net) amount charged for using facility i, i.e., the payment α_j should be $c_{ij} + \beta_{ij}$. The complementary slackness conditions can be used to show that (x, y) and (α, β) have the same value. This fact is true for any pair of primal and dual linear programs, and is known as *strong duality* in linear programming theory.

Strong Duality *For any pair of primal and dual linear programs, the optimal primal value is equal to the optimal dual value*[1]*.*

As a corollary we obtain the following.

Weak Duality *The value of any feasible solution to the dual of a minimization linear program is a lower bound on the optimal value of the minimization program.*

2.2 The Jain-Vazirani Algorithm

Jain & Vazirani [41] gave an elegant algorithm for UFL based on the primal-dual schema. The basic mechanism involves constructing simultaneously a feasible dual solution and an (integer) primal solution. In almost all applications of the schema to date, this is done by a dual ascent algorithm and so all dual variables are only increased throughout the execution of the algorithm. The analysis shows that the cost of the primal solution so obtained is within a factor ρ of the value of the dual solution. Consequently, by weak duality, we get that the algorithm is a ρ-approximation algorithm.

The JV algorithm for UFL works in two steps. Step 1 is a dual ascent process where we simultaneously build a dual solution and a primal feasible solution. Recall that each client j has a dual variable α_j that can be interpreted intuitively as the amount that j is willing to pay, and the dual problem seeks to maximize the total payment from all clients. The dual ascent process increases each dual variable α_j

[1]Here we are assuming that the primal and dual programs are feasible.

uniformly. All other variables react to this change trying to maintain feasibility or complementary slackness. Once α_j becomes equal to c_{ij} for some facility i, we start increasing β_{ij} and start paying toward the facility opening cost of i. When the total contribution to i from the various clients equals f_i, we declare i to be *tentatively open*, and freeze all the (unfrozen) clients that have already reached i (i.e., $\alpha_j \geq c_{ij}$) or reach i at a later point, that is, we do not raise their dual variables any further. Intuitively, we think of these clients as being served by this tentatively open facility (but this facility might get closed in Step 2). The process ends when all clients are frozen, with every client being assigned to a tentatively open facility. At this point a client could be contributing toward multiple tentatively open facilities and the primal solution might be very expensive. In Step 2 we remedy this by carefully picking a subset of the tentatively open facilities to open, and thus extract a near-optimal primal solution. The analysis shows that if the facility i that caused a client j to freeze in Step 1 is not opened, then there is a "nearby" open facility i' to which j can be assigned.

We now describe the algorithm more precisely. There is a notion of time, t. We start with $t = 0$ and all dual variables set to 0. As time increases we raise each dual variable α_j at unit rate until one of the following events happens (if several events happen simultaneously, consider them in any order):

1. At some time t, for some client j and facility i, we have $\alpha_j = t$ equal to c_{ij}, so (3) holds with equality. If facility i is not tentatively open, we start increasing β_{ij} at the same rate as α_j, so we maintain that $\beta_{ij} = \alpha_j - c_{ij}$ and ensure that constraint (3) is satisfied (with equality). If i is tentatively open, we freeze demand j.

2. A facility gets paid for, that is, $\sum_j \beta_{ij} = f_i$. We tentatively open facility i and freeze all (unfrozen) clients j that have reached i, that is, for which $\alpha_j \geq c_{ij}$.

We only raise the α_j and β_{ij} for unfrozen clients j. Frozen demands do not participate in any events. We continue this process until all demands become frozen. Let (α, β)

denote the final dual solution obtained by the above process.

Now we decide which of the tentatively open facilities to open. We say that two facilities i and i' are *dependent* if there is some client j such that both $\beta_{ij}, \beta_{i'j} > 0$. Call a set of facilities *independent* if no two facilities in the set are dependent. We select a maximal independent subset of tentatively open facilities and open these. Now we simply assign each client to the nearest open facility. Note that we do not specify how to pick the independent set — the analysis will show that *any maximal independent set* suffices. We use this property in Chapter 5.

Analysis. Let OPT denote the common optimal value of (UFL-P) and (UFL-D). Let F' be the set of open facilities, and D' be the set of demands that pay for the facilities in F'; that is, $D' = \{j : \exists i \in F' \text{ s.t. } \beta_{ij} > 0\}$. Observe that by our independent set construction, for each demand $j \in D'$, there is exactly one facility i in F' such that $\beta_{ij} > 0$. Define t_i to be the time at which facility i was declared tentatively open. Let $i(j)$ denote the facility to which client j is assigned. We will show that the cost of the solution returned is at most $3\sum_j \alpha_j \leq 3 \cdot OPT$. In fact, we will prove a stronger guarantee that shows that $3\sum_{i\in F'} f_i + \sum_j c_{i(j)j} \leq 3\sum_j \alpha_j$. This stronger guarantee will be useful when we consider k-median problems in Chapter 7. The following fact is evident from the construction of the algorithm.

Fact 2.2.1 *If $\beta_{ij} > 0$, then $\alpha_j \leq t_i$ and $c_{ij} = \alpha_j - \beta_{ij}$.*

Theorem 2.2.2 *The solution returned by the JV algorithm satisfies*

$$3\sum_{i\in F'} f_i + \sum_j c_{i(j)j} \leq 3\sum_j \alpha_j \leq 3 \cdot OPT.$$

Proof : For each facility in F', we have $f_i = \sum_j \beta_{ij} = \sum_{j\in D'} \beta_{ij}$, since i was tentatively opened. Summing over all i in F', we get that the facility opening cost is bounded by $\sum_{i\in F'}\sum_{j\in D'} \beta_{ij} = \sum_{j\in D'} \beta_{o(j)j}$ where $o(j), j \in D'$, denotes the unique facility in F' for which $\beta_{ij} > 0$.

We bound the service cost of a client j by showing that there always is an open facility at most a certain distance away, implying that the nearest open facility can only be closer. If $j \in D'$, then $o(j)$ is open and $c_{o(j)j} = \alpha_j - \beta_{o(j)j}$ by Fact 2.2.1. Now consider $j \notin D'$. Let i be the facility that caused j to freeze. So we have $t_i \leq \alpha_j$. If $i \in F'$, then it is open and $c_{ij} \leq \alpha_j$. Otherwise since we pick a maximal independent set, i must be dependent with some facility $i' \in F'$ via some client k. So we have $\beta_{ik}, \beta_{i'k} > 0$. Then, i' is open, and repeatedly applying Fact 2.2.1 we get that $c_{ik}, c_{i'k} < \alpha_k$ and $\alpha_k \leq \min(t_i, t_{i'}) \leq \alpha_j$. Hence, $c_{i'j} < 3\alpha_j$.

The service cost $c_{i(j)j}$ is therefore at most $\alpha_j - \beta_{o(j)j}$ (which is non-negative) if $j \in D'$ and at most $3\alpha_j$ otherwise. Now we can bound $3\sum_{i \in F'} f_i + \sum_j c_{i(j)j}$ by

$$\sum_{j \in D'} \left(3\beta_{o(j)j} + \alpha_j - \beta_{o(j)j}\right) + \sum_{j \notin D'} 3\alpha_j \leq 3\sum_j \alpha_j \leq 3 \cdot OPT.$$

■

Remark 2.2.3 With demands d_j, the only change to the algorithm and its analysis is that we replace α_j and β_{ij} by α_j/d_j and β_{ij}/d_j respectively, where raising $\{\alpha_j, \beta_{ij}\}/d_j$ at unit rate means that we increase $\{\alpha_j, \beta_{ij}\}$ at rate d_j.

2.3 The Chudak-Shmoys Algorithm

We now describe the LP rounding algorithm of Chudak & Shmoys [18] which achieves an approximation ratio of $\left(1 + \frac{2}{e}\right)$. Let (x, y) and (α, β) be the optimal primal and dual solutions. Recall that OPT is the common optimal value. Conceptually, one can view the CS algorithm as a combination of the decomposition technique due to Shmoys, Tardos & Aardal [71] (STA) and the randomized rounding technique of Raghavan & Thompson [59]. Before delving into the CS algorithm, we briefly discuss a few key ideas from the STA algorithm. Shmoys, Tardos & Aardal defined the following notion of a g-close solution: a solution $(\hat{x}, \hat{y})$ is g-close if for every j, $\hat{x}_{ij} > 0 \implies c_{ij} \leq g_j$. A central component of the STA algorithm is a procedure to transform any g-close fractional solution $(\hat{x}, \hat{y})$ to a $3g$-close *integer* solution $(\tilde{x}, \tilde{y})$ with facility cost $\sum_i f_i \tilde{y}_i$ at most $\sum_i f_i \hat{y}_i$. Thus the total cost of this integer solution is at most, $\sum_i f_i \hat{y}_i + \sum_j 3g_j$.

Chudak & Shmoys build upon this in two ways. First, they observed that the optimal solution (x, y) is α-close due to complementary slackness. So running the STA rounding procedure on (x, y) produces a solution of cost at most $\sum_i f_i y_i + \sum_j 3\alpha_j \leq 4 \cdot OPT$ (by strong duality), giving a 4-approximation algorithm. Second, and more significantly, they use randomization to select which facilities to open, and by doing so, obtain a randomized procedure that shows that any g-close fractional solution $(\hat{x}, \hat{y})$ can be transformed to an integer solution $(\tilde{x}, \tilde{y})$ with *expected* facility cost at most $\sum_i f_i \hat{y}_i$, and expected assignment cost at most $\sum_{j,i} c_{ij}\hat{x}_{ij} + \frac{2}{e}\sum_j g_j$. Running this procedure on the optimal solution (x, y) (which is α-close), we get an integer solution of total expected cost at most $\left(1 + \frac{2}{e}\right) \cdot OPT$.

For convenience, we now work with the optimal solution (x, y) which is α-close, although the algorithm and analysis apply more generally to any g-close fractional solution. Assume for now that for every i and j, if $x_{ij} > 0$ then $x_{ij} = y_i$. Let $F_j = \{i : x_{ij} > 0\}$. The CS algorithm proceeds by first dividing the fractionally open facilities into disjoint clusters. Each cluster is centered around some client j, and consists of the facilities in F_j; so the cluster contains a fractional facility weight equal to 1, that is, $\sum_{i \in F_j} y_i = 1$. We create these clusters in such a way that each non-cluster-center client is "near" some cluster center, that acts as a representative of the client. We will open each facility i with probability y_i, and so the expected facility opening cost is bounded by $\sum_i f_i y_i$. Each non-cluster facility i is opened independently with probability y_i. The facilities within a cluster are opened in a dependent fashion so that exactly one facility is opened from the cluster. The facility opened in a cluster serves as a *backup facility* for all clients that have this cluster center as their representative. We assign each client to the nearest open facility. To analyze the total assignment cost incurred, we consider a suboptimal way of assigning a client j to an open facility, and bound the cost incurred under this assignment. We assign j to the nearest facility open among the facilities in F_j, if some such facility is open, and otherwise to its backup facility. Due to randomization, one can argue that the latter event happens with a small probability and thereby show that the service cost incurred is at most $\sum_{j,i} c_{ij}x_{ij} + \frac{2}{e}\sum_j \alpha_j$.

The algorithm details are as follows. First, we ensure that the fractional solution (x, y) has the property that for every i, j, $x_{ij} > 0 \implies x_{ij} = y_i$. This property is called *completeness* in [18]. If (x, y) is not complete, we will obtain an equivalent instance, and a complete solution $(\hat{x}, \hat{y})$ for this instance with the same cost as (x, y). Let i be a facility such that for some j, $0 < x_{ij} < y_i$. Let $0 < v_1 < v_2 < \cdots < v_k$ be the distinct non-zero values in $\{x_{ij}\}_{j\in\mathcal{D}}$. Note that $y_i = v_k$ otherwise we could decrease y_i to get a solution of lower cost. Let $v_0 = 0$. We replace facility i by k "clones" $i_1, \ldots, i_k$ and set $\hat{y}_{i_l} = v_l - v_{l-1}$. For any j, if $x_{ij} = v_l > 0$, we set $\hat{x}_{i_m j} = \hat{y}_{i_m}$ for $m \le l$, and $\hat{x}_{i_m j} = 0$ for $m > l$. Clearly, $(\hat{x}, \hat{y})$ is a feasible complete solution for the new instance with the same cost as (x, y). Furthermore, any solution to the new instance gives a solution to the original instance of no greater cost. To avoid cumbersome notation, we therefore simply assume that (x, y) is a complete solution. Let $\bar{C}_j = \sum_i c_{ij}x_{ij}$ be the cost incurred by the LP solution to assign client j.

A1. Let $\mathcal{S}$ be a list of clients ordered by increasing $\bar{C}_j + \alpha_j$ value. We repeatedly do the following: pick the first client in $\mathcal{S}$, that is, the client with smallest $\bar{C}_j + \alpha_j$ value among all clients in $\mathcal{S}$, and form a cluster around it consisting of all the facilities in F_j. We then remove from $\mathcal{S}$ every client (including j) that is served by some facility in F_j, make j the representative of each such client, and continue with the remaining list of clients until $\mathcal{S}$ becomes empty. Let D be the set of cluster centers, and $\sigma(k) \in D$ denote the representative of a client k. We call the facilities within clusters, *central facilities* and the facilities outside clusters, *non-central facilities*.

A2. Within each cluster F_j, $j \in D$, we open exactly one facility, choosing facility i with probability y_i. This facility is called the backup facility for every client k with $\sigma(k) = j$.

A3. Each non-central facility is opened independently with probability y_i.

A4. We assign each client to the nearest open facility.

2.3.1 Analysis

To bound the assignment cost, consider the following way of assigning a client j to an open facility: assign j to the nearest open facility in F_j; if no facility in F_j is opened, assign j to its backup facility (opened from $F_{\sigma(j)}$). Let X_j be the random variable denoting the assignment cost of j under this scheme. Let Z_j be the event that no facility in F_j is open.

Lemma 2.3.1 *For any client $j \in D$, we have* $\mathrm{E}\big[X_j\big] = \bar{C}_j$.

Proof : Recall that $x_{ij} > 0$ implies that $x_{ij} = y_i$. If $j \in D$, then *exactly* one of the facilities in F_j is open, facility i being open with probability y_i. So $\mathrm{E}\big[X_j\big] = \sum_i c_{ij} x_{ij} = \bar{C}_j$. ■

Lemma 2.3.2 *For any client $j \notin D$,* $\mathrm{E}\big[X_j | Z_j\big] \leq 2\alpha_j + \bar{C}_j$.

Proof : Let $k = \sigma(j) \in D$. Let $A = F_j \cap F_k \neq \emptyset$. For any facility $i \in A$ we have $c_{ij} \leq \alpha_j$ and for any facility $i \in F_k$, we have $c_{ik} \leq \alpha_k$ due to complementary slackness. Event Z_j implies that j is assigned to its backup facility in F_k, so conditioned on Z_j, X_j is equal to c_{Ij} where I is the (random) backup facility opened from F_k. If there is some facility $i \in A$ such that $c_{ik} \leq \bar{C}_k$, then we have a deterministic bound of $X_j \leq \alpha_j + \bar{C}_k + \alpha_k$. Otherwise, since the unconditional expectation $\mathrm{E}\big[X_k\big]$ is at most $\bar{C}_k$ (Lemma 2.3.1), by conditioning on Z_j, we are only removing weight from facilities that have a larger c_{ik} value than the average. So the conditional expectation $\mathrm{E}\big[X_k | Z_j\big] \leq \bar{C}_k$ and it follows that $\mathrm{E}\big[X_j | Z_j\big] \leq c_{jk} + \mathrm{E}\big[X_k | Z_j\big] \leq \alpha_j + \alpha_k + \bar{C}_k$. So in either case we have that $\mathrm{E}\big[X_j | Z_j\big] \leq \alpha_j + \alpha_k + \bar{C}_k \leq 2\alpha_j + \bar{C}_j$, where the last inequality follows since we consider clients in increasing order of $\alpha_j + \bar{C}_j$ and k was picked before j. ■

We will need the following lemma to bound the assignment cost.

Lemma 2.3.3 *Let $d_1 \leq d_2 \leq \cdots \leq d_m$ and $0 \leq p_n \leq 1$ for $n = 1, \ldots, m$. Then,*

$$p_1d_1 + (1-p_1)p_2d_2 + \cdots + (1-p_1)(1-p_2)\dots(1-p_{m-1})p_md_m \\ \leq \frac{\sum_{n\leq m} p_nd_n}{\sum_{n\leq m} p_n}\Big(1 - \prod_{n\leq m}(1-p_n)\Big).$$

Proof : The proof here is from [73] (see [18] for an alternate proof) and uses the Chebyshev Integral Inequality (see [36]) which states the following. Let g, h be functions from the interval $[a, b)$ to $\mathbb{R}_+$ where g is monotonically non-increasing and h is monotonically non-decreasing. Then,

$$\int_a^b g(x)h(x)dx \leq \frac{\left(\int_a^b g(x)dx\right)\left(\int_a^b h(x)dx\right)}{b-a}.$$

Now take $g(x)$ and $h(x)$ to be functions defined on the interval $\big[0, P = \sum_{n\leq m} p_n\big)$ with $g(x) = \Pi_{n=1}^{i-1}(1-p_n)$ and $h(x) = d_i$ over the interval $\big[\sum_{n=1}^{i-1} p_n, \sum_{n=1}^{i} p_n\big)$ for $i = 1, \dots, m$. This gives,

$$p_1d_1 + (1-p_1)p_2d_2 + \cdots + (1-p_1)(1-p_2)\dots(1-p_{m-1})p_md_m \\ = \int_0^P g(x)h(x)dx \ \leq \ \frac{\left(\int_0^P g(x)dx\right)\left(\int_0^P h(x)dx\right)}{P} \ = \ \frac{\sum_{n\leq m} p_nd_n}{\sum_{n\leq m} p_n}\Big(1 - \prod_{n\leq m}(1-p_n)\Big).$$

■

Lemma 2.3.4 *For any client j, we have* $\mathrm{E}\big[X_j\big] \leq \bar{C}_j + \frac{2}{e}\alpha_j$.

Proof : This is true for a client $j \in D$ by Lemma 2.3.1. Consider $j \notin D$. Recall that if $x_{ij} > 0$ then $x_{ij} = y_i$. For every non-central facility $i \in F_j$, let E_i be the event that i is opened. Let $p_i = \Pr[E_i] = y_i$ and $d_i = c_{ij}$. For every cluster center $k \in D'$ such that $S_k = F_j \cap F_k \neq \emptyset$, let E_k denote the event that a facility in S_k is open. Let $p_k = \Pr[E_k] = \sum_{i\in S_k} y_i$ and define $d_k = \big(\sum_{i\in S_k} c_{ij}x_{ij}\big)/\big(\sum_{i\in S_k} x_{ij}\big)$ which is the expected distance between j and the facility opened from S_k conditioned on event E_k. Let the events be ordered so that $d_1 \leq d_2 \leq \cdots \leq d_m$, where m is the total number of events. We will bound $\mathrm{E}\big[X_j\big]$ by considering a suboptimal way of assigning j to an open facility in F_j (if one exists). Instead of assigning j to the nearest open facility in F_j, we will assign it to the open "facility" with smallest d_i, where when we say that a "facility" of type $S_k, k \in D$ is open, we mean that some facility $i \in S_k$ is open, and assigning j to facility S_k means that we assign it to the open facility in S_k. Observe that the events E_i are all *independent* and $Z_j = \bigcap_{n=1}^{m} \bar{E}_n$. Therefore, $p = \Pr[Z_j] = \prod_{n=1}^{m}(1-p_n) \le e^{-\sum_n p_n} = e^{-1}$. So,

$$\begin{aligned}
\mathrm{E}\big[X_j\big] &\le\ p_1 d_1 + (1-p_1)p_2 d_2 + \cdots + (1-p_1)\ldots(1-p_{m-1})p_m d_m + p \cdot \mathrm{E}\big[X_j|Z_j\big] \\
&\le\ \frac{\sum_{n\le m} p_n d_n}{\sum_{n\le m} p_n}(1-p) + p\cdot(2\alpha_j + \bar{C}_j) \qquad\qquad \text{(Lemma 2.3.2, Lemma 2.3.3)} \\
&=\ (1-p)\bar{C}_j + p\cdot(2\alpha_j + \bar{C}_j)\ \le\ \bar{C}_j + \tfrac{2}{e}\cdot\alpha_j.
\end{aligned}$$

■

Remark 2.3.5 The bounds stated in Lemma 2.3.2 and 2.3.4 hold for any g-close solution with α_j replaced by g_j, if we modify the criterion for selecting a cluster center in step A1 so that the new cluster center chosen is the client with the smallest $\bar{C}_j + g_j$ value among all clients in $\mathcal{S}$.

Theorem 2.3.6 *The expected cost of the solution returned is at most* $\big(1+\frac{2}{e}\big)\cdot OPT$.

Proof : The expected facility cost incurred is at most $\sum_i f_i y_i$, since each facility is opened with probability y_i. By Lemma 2.3.4, the total service cost is at most $\sum_{j,i} c_{ij}x_{ij} + \frac{2}{e}\sum_j \alpha_j$. Adding the two, we see that the total cost incurred is at most $\big(1+\frac{2}{e}\big)\cdot OPT$, since $\sum_j \alpha_j = OPT$. ■

Remark 2.3.7 Again, as in the JV algorithm, to handle arbitrary demands d_j, we just need to replace α_j with α_j/d_j. Of course, the assignment cost incurred is now given by $\sum_j d_j X_j$.

2.4 The Primal Rounding Algorithm

We now describe an algorithm that takes as input *any* feasible fractional solution (x, y) and returns an integer solution $(\tilde{x}, \tilde{y})$ of cost at most a constant factor times the cost of (x, y). A useful property of this rounding algorithm is that neither the algorithm nor the performance guarantee depend on the actual client demands d_j. This will prove useful in Chapter 4 and Section 5.6 when we design approximation algorithms for the stochastic versions of some facility location problems.

Observe that neither of the two algorithms described earlier have this "demand-obliviousness" property. In the JV algorithm, to handle arbitrary demands we raise the variable α_j at rate d_j. In general, since the demands appear in the dual constraints, it seems necessary that in a primal-dual algorithm one would need to know the actual demands in order to construct a good feasible dual solution. The performance guarantee of the CS algorithm, since it uses complementary slackness, relies critically on the fact that we have an optimal solution to start with, and getting an optimal fractional solution requires knowledge of the demands. The STA algorithm [71] does have this property, and by incorporating ideas from both the STA algorithm and the CS algorithm we design an algorithm with a better approximation guarantee.

The algorithm is as follows. Let $F_j = \{i : x_{ij} > 0\}$. Note that the fractional assignment x_{ij} depends only on the y_i values, that is, the fractionally open facilities, and not on the demand d_j. We may assume that (x, y) is a complete solution, i.e., $x_{ij} = 0$ or $x_{ij} = y_i$ for every i and j, if necessary by cloning facilities as in the CS algorithm. Let $\bar{C}_j = \sum_i c_{ij}x_{ij}$ be the cost incurred in the fractional solution to assign one unit of j's demand. Let $0 < \gamma < 1$ be a parameter and $r = \frac{1}{\gamma}$. The algorithm essentially involves getting a g-close solution where g_j is bounded in terms of $\bar{C}_j$, and running the CS algorithm on this solution, but the analysis is different and more refined.

Sort the facilities in F_j by increasing c_{ij} value. Let i' be the first facility in this ordering such that the x_{ij} weight of facilities in F_j up to and including i' (in this ordering) is at least γ, that is, $\sum_{i:i=i', \text{ or comes before } i'} x_{ij} \geq \gamma$. Define $R_j(\gamma) = c_{i'j}$ and $\bar{C}_j(\gamma)$ as the x_{ij}-weighted average distance to the set of facilities in F_j considered in sorted order, that gathers an x_{ij} weight of *exactly* γ. So the last facility i' may be included partially and we have $\bar{C}_j(\gamma) = \big(\sum_{i<i'} c_{ij}x_{ij} + c_{i'j}(\gamma - \sum_{i<i'} x_{ij})\big)/\gamma$. Note that $\bar{C}_j(1) = \bar{C}_j$. Let $N_j \subseteq F_j$ be the facilities up to and including i' in the sorted order.

To simplify the description we assume that each $y_i \leq \gamma$ and for any j, $\sum_{i\in N_j} y_i$ is *exactly* γ. If some $y_i > \gamma$, then we can create at most $\lceil 1/\gamma \rceil$ clones of i and set $y_{i_l} \leq \gamma$ for each clone i_l so that $\sum_{\text{clones } i_l} y_{i_l} = y_i$ (setting the variables $x_{i_l j}$ accordingly). Similarly, if $\sum_{i\in N_j} y_i > \gamma$, we can split i' (the last facility in N_j) into two copies i'_1 and i'_2 and set $y_{i'_2} = \sum_{i\in N_j} y_i - \gamma, y_{i'_1} = y_{i'} - y_{i'_2}$ (and the other variables accordingly). We include only i'_1 in N_j thus ensuring that $\sum_{i\in N_j} y_i = \gamma$. The cost of the fractional solution remains unchanged by these transformations and a solution to the modified instance translates in the obvious way to a solution to the original instance of no greater cost. Hence, $\bar{C}_j(\gamma) = \big(\sum_{i\in N_j} c_{ij}x_{ij}\big)/\gamma$.

Now consider the fractional solution $(\hat{x}, \hat{y})$, where $\hat{x}_{ij} = x_{ij}/\gamma$ for $i \in N_j$ and 0 otherwise, and $\hat{y}_i = y_i/\gamma$. We run the CS algorithm on this instance using the following modified rule for selecting a cluster center in step A1: we now pick the client j with smallest $R_j(\gamma) = \max_{i:\hat{x}_{ij}>0} c_{ij}$ value to be a cluster center.

2.4.1 Analysis

Note that $(\hat{x}, \hat{y})$ is a $R_j(\gamma)$-close solution, and $\sum_i c_{ij}\hat{x}_{ij} = \bar{C}_j(\gamma)$. We prove a tighter bound on the assignment cost than that implied by Lemma 2.3.4 (when applied to the instance $(\hat{x}, \hat{y})$ with the modified cluster-selection criterion). Consider assigning j to the nearest open facility in F_j, and if no such facility is open, to its backup facility in $N_{\sigma(j)}$. Let X_j be the random variable denoting the distance between j and the facility to which it is assigned under this scheme, and Z_j be the event that no facility in F_j is open. Recall that $r = \frac{1}{\gamma}$.

Lemma 2.4.1 *For any client j, we have,*

$$\mathrm{E}\big[X_j\big] \leq \Big(1 + e^{-r} \cdot \frac{2+\gamma}{1-\gamma}\Big)\bar{C}_j.$$

Proof : The proof is as in Lemma 2.3.4. We can decompose event Z_j into the intersection of various independent events E_i and for each of these events we define $p_i = \Pr[E_i]$ and d_i as in Lemma 2.3.4. Let $k = \sigma(j)$ be the representative client j. Observe that X_j is deterministically bounded by $R_j(\gamma) + 2R_k(\gamma) \leq 3R_j(\gamma)$, since $N_j \cap N_k \neq \emptyset$ and some facility from N_k is always opened. Thus, we can trivially bound the conditional distance $\mathrm{E}\big[X_j|Z_j\big]$ by $3R_j(\gamma)$. Also $p = \Pr[Z_j]$ is at most $e^{-\sum_{i\in F_j} ry_i} = e^{-r}$, and $\big(\sum_n p_n d_n\big)/\big(\sum_n p_n\big) = \bar{C}_j$. So as in Lemma 2.3.4, we obtain that

$$\mathrm{E}\big[X_j\big] \leq (1-p)\bar{C}_j + p \cdot 3R_j(\gamma) \leq (1-p)\bar{C}_j + p \cdot \frac{3}{1-\gamma}\bar{C}_j,$$

where the last inequality follows since $R_j(\gamma) \leq \frac{\bar{C}_j}{1-\gamma}$. Since $p \leq e^{-r}$, $\mathrm{E}\big[X_j\big]$ is at most $\big(1 + e^{-r} \cdot \frac{2+\gamma}{1-\gamma}\big)\bar{C}_j$. ■

Theorem 2.4.2 *Consider any demands $d_j \geq 0$. Let $C = \sum_j d_j\bar{C}_j$ and $F = \sum_i f_i y_i$. For any parameter $\gamma, 0 \leq \gamma \leq 1$, the above algorithm produces an integer solution $(\tilde{x}, \tilde{y})$ with expected facility cost at most $r \cdot F$ and expected assignment cost $\mathrm{E}\big[\sum_{j,i} d_j c_{ij}\tilde{x}_{ij}\big]$ at most $\big(1 + e^{-r} \cdot \frac{2+\gamma}{1-\gamma}\big) \cdot C$, where $r = \frac{1}{\gamma}$. Thus with $\gamma = \frac{1}{1.858}$, we get a solution of total cost at most $1.858(F + C)$.*

Proof : Each facility i is opened with probability $r \cdot y_i$, so the facility cost is bounded by $r \cdot F$. The expected service cost is at most $\sum_j d_j\mathrm{E}\big[X_j\big]$ which is at most $\big(1 + e^{-r} \cdot \frac{2+\gamma}{1-\gamma}\big) \cdot C$ by Lemma 2.4.1. The theorem follows. ■

Remark 2.4.3 In particular with demands $d_j \in \{0, 1\}$, this shows that we can choose which facilities to open, without knowing the actual client set that we have to serve! The theorem shows that for *any* client set S, the total cost of the integer solution $\tilde{y}$ is within a constant factor of the cost of the fractional solution where clients in S are assigned (fractionally) to the fractionally open facilities in y.

Part II

Stochastic Linear Programming

Chapter 3

An Algorithm to Solve 2-Stage Stochastic Linear Programs

3.1 Introduction

Stochastic optimization problems have been studied since the work of Dantzig [21] and Beale [11] in the 1950s, and attempt to model uncertainty in the data by assuming that (part of) the input is specified in terms of a probability distribution, rather than by deterministic data given in advance. Since the work of Dantzig, stochastic optimization, also referred to as stochastic programming, has grown into a tremendous field with a vast literature including various textbooks [13, 42], surveys and collected papers [72, 79, 66] and repositories on the web [20, 78]. Stochastic optimization techniques and models have become an important paradigm in a wide range of application areas, including transportation models, logistics, financial instruments, and network design.

In this chapter we focus on an important and widely used model in stochastic programming: the *2-stage recourse model*, where one makes decisions in two steps. First, given only distributional information about (some of) the data, one commits on initial (first-stage) actions, and then once the actual data is realized, according to the distribution, further *recourse actions* can be taken, so that one can augment the earlier solution to satisfy the revealed requirements, if necessary. Typically the recourse actions entail making decisions in rapid reaction to the observed scenario, that is, at the "last minute," and are therefore costlier than decisions made ahead of time. Thus there is a natural trade-off between committing in advance without having precise information and paying a low cost, and waiting to observe the scenario that materializes and then making decisions having complete information but paying a higher price, and this reflects the need for careful planning in deciding the initial actions, that is, the first-stage decisions. The class of 2-stage recourse problems finds a wide variety of applications, and has been extensively studied in the stochastic optimization literature. For example, much of the textbook of Birge & Louveaux [13] is devoted to applications and algorithms for this class of problems.

Consider, as an illustrative example, the following inventory management problem that might form one cog of a supply-chain logistics problem, where one has to decide on inventory levels given only estimates or likelihood information about the demand that one has to satisfy. Here the uncertainty in the demand might be due to a combination of several factors such as market forces, inflation, state of the economy, uncertainty propagating from other parts of the supply-chain. Having decided initially on certain inventory levels, one gets to observe the actual demand requirement, and then one has the opportunity to adjust the inventory levels accordingly, at the expense of a recourse cost. If there is excess demand, then one can raise inventory levels to satisfy the extra demand, but since the extra demand has to be satisfied in a timely manner (not to lose out on possible sales and profits) this requires (re-) positioning the inventory at a short notice, that is, at a smaller turnaround time, and typically incurs a higher cost. Conversely, if there is less demand, then one may either choose to store the inventory and incur a certain holding cost, or in a supply chain one might be able to sell the surplus inventory at a lower price to the previous supplier(s), and incur a loss. The inventory management problem fits nicely in the 2-stage recourse model. The aim here is to obtain a good set of initial (first-stage) decisions, that is, come up with an effective plan in the face of uncertain data.

As another example, picture a network designer who has to allocate bandwidth on network links without knowing the exact traffic requirements of the end-users. He has the flexibility to install some bandwidth initially, and once the traffic requirements become known, or better understood, the recourse might consist of increasing capacity on some of the links. Here again, there is a trade-off between decisions made in the first stage or in the second stage. The network designer may be able to arrange cheap long-term contracts for capacity purchased ahead of time (without knowing exactly how much capacity to purchase), whereas he may need to purchase capacity at the last minute in a more expensive "spot market."

To capture the above problems, we formalize the 2-stage recourse model as follows: we are given a probability distribution on input instances A, and we construct a solution in two stages. In the first-stage, we may choose some elements to construct an anticipatory part of the solution, x, and incur a cost $c(x)$, then the problem instance, or scenario, say A, is revealed, and in the second stage, we may augment the first stage decisions by choosing some more elements y_A (if necessary) incurring a certain cost $f(x, y_A)$. The goal is to decide which elements to choose in stage I, that is, the vector x, so as to minimize $c(x) + \mathrm{E}_A\big[f(x, y_A)\big]$, that is, the sum of the stage I cost and the expected stage II cost, where the expectation is taken over all possible scenarios according to the given probability distribution.

3.1.1 Summary of Results

In this chapter, we describe an algorithm to solve a rich class of 2-stage stochastic *linear programs* to near-optimality. We present an algorithm that returns a solution of objective function value within $(1 + \epsilon)$ of the optimum, where ϵ can be set arbitrarily close to 0, in running time that is polynomial in the input size, $\frac{1}{\epsilon}$, and in the *ratio* of the second stage and first stage costs, thus obtaining a *fully polynomial time randomized approximation scheme* (FPRAS) for a large class of 2-stage stochastic linear programs. The algorithm works for both discrete and continuous distributions, and does not require any assumptions about the probability distribution (or the cost structure of the input); all we require is a "black box" that one can use to draw independent samples from the distribution. This result is based upon formulating the stochastic linear program, which in general has both an exponential number of variables and an exponential number of constraints, as an equivalent, compact convex program, and then adapting the ellipsoid method to return a near-optimal solution. In doing so, a significant difficulty that we must overcome is that even evaluating the objective function of this convex program at a given point may be quite difficult and $\#P$-hard [24] in general, due to the exponential number of scenarios that one may have to consider.

The ability to solve stochastic linear programs to near-optimality provides one with a powerful and versatile tool to design approximation algorithms for 2-stage stochastic integer optimization problems, in much the same way that linear programming has proved to be immensely useful in the design and analysis of approximation algorithms for deterministic integer optimization problems. Given a 2-stage stochastic integer problem one can consider a linear relaxation of the problem obtained by dropping the integrality constraints. We will show in subsequent chapters that one can convert a fractional near-optimal solution to this 2-stage stochastic linear program to an integer solution using approximation algorithms for the deterministic version of the problem at the expense of a small constant-factor blowup in the objective function value, and thus obtain an approximation algorithm for the 2-stage stochastic integer problem. We exploit this tool in Chapter 4 and Section 5.6 to devise the first approximation algorithms for the stochastic uncapacitated facility location (SUFL) problem and some of its variants, without placing any restrictions on the underlying probability distribution or the cost structure of the input.

This chapter is structured as follows. In Section 3.2 we consider a stochastic generalization of the deterministic set cover problem which is used to illustrate the algorithm and its analysis in Section 3.4. In Section 3.5 we generalize the arguments to show that the algorithm can be used to solve a larger class of stochastic linear programs. The running time of our algorithm depends on the ratio of the stage II and stage I costs; in Section 3.6, we show that this dependence is unavoidable in the black-box model.

3.1.2 Related Work

Two-stage stochastic programs, both linear and integer programs, have been extensively studied in the Operations Research literature, but not much is known about algorithms that deliver solutions that are provably good approximations to the optimum stochastic linear or integer program objective value in polynomial time. We first discuss work relating to 2-stage stochastic linear programming, and then briefly review work related to approximation results for 2-stage stochastic integer programs, which is discussed in more detail in Section 4.1.2 in the context of the stochastic uncapacitated facility location problem.

Two-stage stochastic linear programs are often computationally quite difficult, both from the point of view of complexity theory, and from a practical perspective, mainly due to the fact that there may be an exponential number of scenarios, and the problem complexity increases considerably as the number of scenarios increases, a phenomenon often phrased as "the curse of dimensionality." Dyer & Stougie [24] showed that even evaluating the objective function, which includes the expectation over stage II scenarios, at a given point may be $\#P$-hard.

There is a large body of literature that deals with computational aspects of solving 2-stage stochastic linear programs, and providing theoretical asymptotic guarantees where possible, and we only sample a few key ideas here. One approach used is to enumerate all possible scenarios and express the problem as a (huge) linear program (LP), and then exploit the fact that this LP has a great deal of structure to devise specialized heuristics for solving the LP; for example, the L-shaped structure of the coefficient matrix has given rise to the so called L-shaped method (see for example, [13]). However, even this additional structure might not be enough to offset the computational burden of having to deal with an exponential number of scenarios. An alternative approach taken is to sample a certain number of times from the distribution on scenarios, approximate the probability of a scenario by its frequency, and solve the 2-stage problem for this approximate distribution (that has support of size at most the number of samples). There are several variants of this basic method, depending on whether the approximate probability distribution is used to approximate the function value or some other quantity such as optimality cuts or gradients, the type of sampling procedure used, whether sampling is performed only initially or repeatedly, in the latter case whether the number of samples is kept fixed or varied. The textbook by Birge & Louveaux [13] gives an account of these methods as well as the non-sampling-based methods mentioned earlier. While there are some results that prove asymptotic convergence to the optimal solution in the limit *as the number of samples goes to infinity*, and it has been reported that some of these algorithms converge quickly [55, 77], fewer results are known about the rate of convergence to a near-optimal solution and the sample size required to obtain a near-optimal solution (with high probability). The work that seems to most closely deal with the issue of bounding the number of required samples, is a paper by Kleywegt, Shapiro & Homem-De-Mello [46] (see also Shapiro [67]). They give a bound that is polynomial in the dimension, but depends on the variance of a certain quantity (calculated using the scenario distribution) that might not vary polynomially with the input size.

The algorithm we present incorporates some of the above ideas and builds upon them. We show that a randomized polynomial algorithm that samples from the probability distribution (which is treated as a black box) can compute an approximate subgradient (appropriately defined), and that this approximate subgradient information can be leveraged within the framework of the ellipsoid algorithm to guarantee convergence to a $(1+\epsilon)$-optimal solution in polynomial running time. Thus, this gives a theoretical justification for the effectiveness of repeated sampling as a tool to tackle such problems and obtain good convergence results. The running time of our algorithm, and hence the number of samples used, is polynomial in the input size, $\frac{1}{\epsilon}$ and the maximum *ratio* λ between the stage II and stage I costs, but *does not depend on the underlying distribution*. We show that this dependence on λ is unavoidable, and that a performance guarantee of ρ requires $\Omega(\lambda/\rho)$ samples. The dependence on $\frac{1}{\epsilon}$ is also unavoidable in light of the $\#P$-hardness results. Also related is the work of Dyer, Kannan, and Stougie [23], who focus on computing an estimate for the objective function value at a given point (though for a maximization version), by sampling from the distribution sufficiently many times. But this yields a running time that is polynomial only in the *maximum value* attained by *any* scenario. Furthermore, their guarantee does not yield a fully polynomial approximation scheme for 2-stage linear programs. In contrast, we focus on approximating the subgradient at a given point, and show that the variation in the subgradient vector components is more controlled and depends only on the ratio λ; consequently our running time depends only on λ.

The first worst-case analysis of approximation algorithms for 2-stage stochastic integer programming problems with recourse appears to be due to Dye, Stougie & Tomasgard in 1999 [22], who consider a resource provisioning problem where there are only a polynomial number of scenarios and give a constant performance guarantee based on an LP rounding algorithm. One important issue left ambiguous in the description of the 2-stage recourse model above is the way in which the probability distribution is specified, and several approaches have recently been considered in papers that address related 2-stage stochastic integer optimization problems. Dye et al. [22], and Ravi and Sinha [61] assume that there are only a polynomial number of scenarios, i.e., choices for A that occur with positive probability; we will refer to this model as the *polynomial scenario* model. Independently, Immorlica, Karger, Minkoff, and Mirrokni [39] consider both this model, and the model where each element is assumed to occur with its own independent probability; for example, in the inventory management application, this means that the demand for each product is independently set and has no effect on the demands for other products. We denote this model the *independent-activation* model. By looking at distributions generated this way, Immorlica et al. enlarge the space of scenarios to be exponentially large. This is done with the rather severe restriction of assuming that the *costs in the two stages are proportional*, that is, there is a uniform inflation parameter λ that blows up the cost incurred in making *any* decision, in *any* stage II scenario, by a factor of λ compared to the cost incurred in making that decision in stage I. We call this the *proportional-costs* model. In the inventory example, this would imply that the cost to set up a warehouse in stage II (at the last minute) in scenario A is λ times the cost to set it up in stage I, and this parameter λ is the same for *every* warehouse, and *every scenario* A. Gupta, Pál, Ravi, and Sinha [35] also require this assumption, but give a more general way to specify the probability distribution, which we shall call the *black-box model*: they assume that the algorithm may make use of samples that are drawn according to the distribution of scenarios. Ravi and Sinha [61], and independently Immorlica et al. [39], and subsequently Gupta et al. [35] consider 2-stage stochastic versions of some combinatorial optimization problems, and give approximation algorithms in the models mentioned above which restrict either the probability distribution, or the cost structure (or both).

3.2 An Illustrative Problem

We shall initially focus on a stochastic generalization of the ordinary set cover problem to illustrate our technique for solving 2-stage stochastic linear programs. In Section 3.5, we show that the technique can be used to solve a rich class of 2-stage programs to near-optimality.

The deterministic weighted set cover problem (SC) is the following: given a universe U of elements $e_1, \ldots, e_n$ and a collection of subsets of U, $S_1, \ldots, S_m$ with set S_i having weight w_i, we want to choose a minimum weight collection of sets so that every element $e_j, j = 1, \ldots, n$, is included in some chosen set. The problem can be formulated as an integer program and the integrality constraints can be relaxed to yield the following linear program:

$$\min \quad \sum_S w_S x_S \quad \text{subject to} \quad \sum_{S: e \in S} x_S \geq 1 \quad \text{for all } e; \quad x_S \geq 0 \quad \text{for all } S. \qquad \text{(SC-P)}$$

In the two-stage stochastic generalization of the problem, abbreviated SSC, the elements to be covered are not known in advance. There is a probability distribution over scenarios, and each scenario specifies the actual set of elements $A \subseteq U$ to be covered. For our purposes, a scenario is just some subset of the elements $A \subseteq U$. We will assume without loss of generality that the set of all possible scenarios is the power set 2^U (including the empty set $\emptyset$), and use p_A to denote the probability of scenario A. Note that p_A could be 0, implying that scenario A never actually materializes. We point out that *the quantities p_A are never explicitly used by the algorithm.* We define them only for ease of exposition and to aid us in the analysis. Throughout we will use A to index the scenarios.

Each set S_i has two costs associated with it, an *a priori* weight w_i^{I}, and an *a posteriori* weight w_i^{II}. In the first stage, one selects some of these sets, incurring a cost of w_S^{I} for choosing set S, and then a scenario $A \subseteq U$ is drawn according to a specified distribution, and then additional sets may be selected incurring their second stage weights so as to ensure that A is contained in the union of the sets selected both in stage I and in stage II. The aim is to minimize the expected total cost of the solution, that is, the sum of the cost incurred in stage I and the expected stage II cost of a scenario, where the expectation is taken over all scenarios A.

The 2-stage problem can also be formulated as an integer program and the integrality constraints can be relaxed to yield a linear program. We have a variable x_S for each set S indicating whether set S is chosen in stage I, and variables $r_{A,S}$ for each set S and scenario A, indicating if set S is chosen in scenario A.

$$\begin{aligned}
\min \quad & \sum_S w_S^{\mathrm{I}} x_S + \sum_{A,S} p_A w_S^{\mathrm{II}} r_{A,S} && && \text{(SSC-P1)}\\
\text{s.t.} \quad & \sum_{S:e\in S} x_S + \sum_{S:e\in S} r_{A,S} \geq 1 && \text{for all } A, e \in A, && (1)\\
& x_S, r_{A,S} \geq 0 && \text{for all } A, S.
\end{aligned}$$

Constraint (1) says that in every scenario A, every element in that scenario has to be covered by a set chosen either in stage I or in stage II. An integer ($\{0,1\}$) solution corresponds exactly to a solution to our problem, and relaxing the integrality constraints gives a linear program. Notice however, that this LP has both an exponential number of variables and an exponential number of constraints, and in general, obtaining an optimal solution to (SSC-P1) in its present form seems difficult, since even writing out an optimal solution may take exponential space (and time). However, in order to round a fractional solution to (SSC-P1) to an integer solution, and thus determine which sets to pick in stage I, it turns out, as we will show later in Section 4.3.1, that one only needs to examine *only the stage I variables* x_S in the fractional solution. This motivates the following compact formulation, equivalent to (SSC-P1), where we only have the stage I variables x_S.

$$\min \quad \sum_S w_S^{\mathrm{I}} x_S + f(x) \quad \text{subject to} \quad 0 \leq x_S \leq 1 \text{ for all } S, \tag{SSC-P2}$$

$$\begin{aligned} \text{where } f(x) &= \sum_{A \subseteq U} p_A f_A(x), \\ \text{and} \quad f_A(x) &= \min \sum_S w_S^{\mathrm{II}} r_{A,S} \\ & \quad \text{s.t.} \sum_{S: e \in S} r_{A,S} \geq 1 - \sum_{S: e \in S} x_S \qquad \text{for all } e \in A, \\ & \qquad\quad r_{A,S} \geq 0 \qquad\qquad\qquad\qquad \text{for all } S. \end{aligned} \tag{2}$$

It is easy to see that any feasible solution to (SSC-P2) maps to a feasible solution (perhaps many solutions) to (SSC-P1) of the same value. Conversely, any solution to (SSC-P1) maps to a solution to (SSC-P2) of no greater value, and hence the the formulations (SSC-P2) and (SSC-P1) are equivalent, in the sense that they have the same optimal value.

Lemma 3.2.1 *The function $f(x)$ in* (SSC-P2) *is convex.*

Proof : It suffices to show that $f_A(x)$ is convex for each $A \subseteq U$. Consider any two points x_1, x_2 and let $x = \lambda x_1 + (1-\lambda)x_2$. Let $r_A^{(1)}$ and $r_A^{(2)}$ be the optimal solutions to the minimization problem for scenario A at x_1, x_2 respectively. Then $\lambda r_A^{(1)} + (1-\lambda) r_A^{(2)}$ is a *feasible solution* for the scenario A minimization problem at point x, and has value $\lambda f_A(x_1) + (1-\lambda) f_A(x_2)$. Therefore $f_A(x) \leq \lambda f_A(x_1) + (1-\lambda) f_A(x_2)$. ■

3.3 Solving the Convex Program: Algorithm Overview

We now leverage the fact that the objective function of (SSC-P2) is convex to adapt a technique from convex optimization, namely the ellipsoid method, and show that one can find a near-optimal solution to (SSC-P2) in polynomial time. In doing so, a significant difficulty that we need to overcome however, is the fact *that evaluating $f(x)$, and hence the objective function, may in general be #P-hard*. Section 3.5 generalizes the arguments to show that the algorithm can be applied to a more general class of 2-stage stochastic programs.

The ellipsoid method starts by containing the feasible region within a ball and generates a sequence of ellipsoids, each of successively smaller volume. In each iteration, one examines the center of the current ellipsoid and obtains a specific half-space defined by a hyperplane passing through the current ellipsoid center. If the current ellipsoid center is infeasible, then one uses a violated inequality as the hyperplane, otherwise, one uses an *objective function cut*, to eliminate (some or all) feasible points whose objective function value is no better than the current center, and thus make progress. A new ellipsoid is then generated by finding the minimum-volume ellipsoid containing the half-ellipsoid obtained by the intersection of the current one with this half-space. Continuing in this way, using the fact that the volume of the successive ellipsoids decreases by a significant factor, one can show that after a certain number of iterations, the feasible point generated with the best objective function value is a near-optimal solution.

The above description makes clear that the inability to evaluate $f(x)$ is an obstacle to applying the ellipsoid method in this case. Let $\mathcal{P} = \mathcal{P}_0$ denote the polytope $\{x \in \mathbb{R}^m : 0 \leq x_S \leq 1 \text{ for all } S\}$, and let $h(x)$ be the (convex) objective function $w^{\mathrm{I}} \cdot x + f(x)$. If one finds that the current iterate x_i is feasible, that one could add the constraint $h(x) \leq h(x_i)$ while maintaining the convexity of the feasible region[1]. But then, in subsequent iterations, one would need to check if the current iterate is

[1]Note that if $h(x)$ were a linear function of x, i.e., $h(x) = c \cdot x$, then this is precisely an objective function cut, $c \cdot x \leq c \cdot x_i$.

feasible, and generate a separating hyperplane if not. Without the ability to evaluate (or even estimate) the objective function value, we cannot even decide whether the current point is feasible (or even almost-feasible), and finding a separating hyperplane appears to pose a formidable difficulty. An alternative possibility is to use cuts generated by a *subgradient*, which essentially plays the role of the gradient when the function is not differentiable.

Definition 3.3.1 *Let $g : \mathbb{R}^m \mapsto \mathbb{R}$ be a function. We say that d is a subgradient of g at the point u if the inequality $g(v) - g(u) \geq d \cdot (v - u)$ holds for every $v \in \mathbb{R}^m$.*

Note that the subgradient at a given point need not be unique. It is known (see [14]) that if a function is convex then it has a subgradient at every point. If d_i is the subgradient at point x_i, one can add the *subgradient cut* $d_i \cdot (x - x_i) \leq 0$ and proceed with the (smaller) polytope $\mathcal{P}_{i+1} = \mathcal{P}_i \cap \{x : d_i \cdot (x - x_i) \leq 0\}$. Unfortunately, even computing the subgradient at a point x seems hard to do in polynomial time for the objective functions that arise in stochastic programs. To circumvent this obstacle, we define the following notion of an approximate subgradient which is crucial to the working of our algorithm.

Definition 3.3.2 *We say that $\hat{d}$ is a $(\omega, \mathcal{D})$-approximate subgradient (or simply a $(\omega, \mathcal{D})$-subgradient) of a function $g : \mathbb{R}^m \mapsto \mathbb{R}$ at the point $u \in \mathcal{D}$ if for every $v \in \mathcal{D}$, we have $g(v) - g(u) \geq \hat{d} \cdot (v - u) - \omega g(u)$.*

We will only use $(\omega, \mathcal{P})$-approximate subgradients in the algorithm, which we abbreviate and denote as ω-subgradients from now on. We show that one can compute, with high probability, a ω-subgradient of $h(.)$ at any point x, by sampling from the black box on scenarios. Since we approximate the subgradient at x and *not* the function value $f(x)$, the running time of our algorithm does not depend on the maximum value of the function $f_A(x)$ over scenarios A and (feasible) points x, in contrast to [23]. Instead, as we show later, the variation in the subgradient vector components is more controlled and depends only the maximum ratio of the stage II and stage I costs, and consequently our running time depends only on this ratio.

At a feasible point x_i, we compute a ω-subgradient $\hat{d}_i$ and add the inequality $\hat{d}_i \cdot (x - x_i) \leq 0$ to chop off a region of $\mathcal{P}_i$ and get the polytope $\mathcal{P}_{i+1}$. Since we use an approximate subgradient to generate the cut, we might be discarding points from $\mathcal{P}_i$ with objective value better than the current function value $h(x_i)$. But for each point y in $\mathcal{P}_{i+1} \setminus \mathcal{P}_i$, we can show that $h(y) \geq (1-\omega)h(x_i)$, so the function value at a discarded point is not *much better off* than the current function value. Continuing this way we obtain a polynomial number of points $x_0, x_1, \ldots, x_k$ such that $x_i \in \mathcal{P}_i \subseteq \mathcal{P}_{i-1}$ for each i, and the volume of the ellipsoid centered at x_k containing $\mathcal{P}_k$ (and hence of $\mathcal{P}_k$) is "small" (we will make this precise later). Now if the function $h(.)$ has bounded variation on nearby points, then one can show that $\min_i h(x_i)$ is close to the optimal value $h(x^*)$ with high probability.

Yet another difficulty remains however. Since we cannot compute $h(x)$ we will not be able to determine the iterate x_i with the best objective function value. Nonetheless, here again we exploit approximate subgradient information to compute a point $\bar{x}$ in the convex hull of $x_0, \ldots, x_k$, at which the objective function function value is close to $\min_i h(x_i)$ (without, however, computing these values). At the heart of this procedure is a subroutine that given two points y_1, y_2, returns a point y on the line segment joining y_1 and y_2 such that $h(y)$ is close to $\min(h(y_1), h(y_2))$. We find such a point y by performing a bisection search, using the subgradient to infer which direction to move along the line segment joining y_1 and y_2. By repeatedly calling the above subroutine with $\bar{x}$ (initialized to x_0) and x_i for $i = 1, \ldots, k$ and each time updating $\bar{x}$ to the point returned by the subroutine, at the end we get a point $\bar{x}$ such that $h(\bar{x})$ is close to $\min_i h(x_i)$.

3.4 Algorithm Details

Let $OPT = \min\{h(x) : x \in \mathcal{P}\}$ denote the optimal solution value. We describe the algorithm for an arbitrary convex function $h(x)$ and an arbitrary (rational) polytope $\mathcal{P}$ (so the feasible region is bounded). We use $\|u\|$ to denote the ℓ_2 norm of u, i.e., $\left(\sum_{i=1}^m u_i^2\right)^{\frac{1}{2}}$. The following definition makes precise the notion of bounded variation.

Definition 3.4.1 (Lipschitz Condition) *Given a function $g : \mathbb{R}^m \mapsto \mathbb{R}$, we say that g has Lipschitz constant (at most) K if $|g(v)-g(u)| \leq K\|v-u\|$ for all $u, v \in \mathbb{R}^m$.*

Let the objective function $h : \mathbb{R}^m \mapsto \mathbb{R}$ have Lipschitz constant K. We assume that $x \geq \mathbf{0}$ is a defining inequality of $\mathcal{P}$. Let $B(\mathbf{0}, R) = \{x : \|x\| \leq R\}$ be a ball containing the polytope $\mathcal{P}$. Such an R is easy to obtain: in all the stochastic optimization problems we will consider, one can put a trivial upper bound $x \leq U$ (we also have $x \geq \mathbf{0}$) on the stage I decision vector x and can therefore set $R = \|U\|$ (it suffices to show that $x^* \leq U$ where x^* is an *optimal* solution, since one can then optimize over the polytope $\mathcal{P}' = \mathcal{P} \cap \{x : x \leq U\}$). Otherwise, we can set $R = 2^{4m^2L}$ where L is the maximum row size of the constraint matrix defining the polytope $\mathcal{P}$ (see [27], Lemmas 6.2.4, 6.2.5). For simplicity we assume that $\mathcal{P}$ is full-dimensional and therefore contains a ball of radius at least $r \geq 2^{-7m^3L}$ ([27], Lemma 6.2.6). Again, this is true for all the problems considered and one can get much better lower bounds on r. Moreover, one can always perturb the polytope slightly to make it full-dimensional, and hence this assumption is not really required. Set $V = \min(1, r)$.

For ease of understanding, we divide the algorithm description and its analysis into two parts. The bulk of the work is performed by procedure FindOpt. FindOpt takes two parameters γ and ϵ and returns a feasible solution $\bar{x}$ such that $h(\bar{x}) \leq OPT/(1-\gamma)+\epsilon$, where $\gamma \leq \frac{1}{2}$ without loss of generality, in time polynomial in the dimension m, and $\ln\left(\frac{KRm}{V\epsilon}\right)$ assuming that one can compute ω-subgradients for a sufficiently small ω. This is the main procedure that uses the ellipsoid method and the notion of ω-subgradients to get close to an optimal solution as discussed earlier. We describe this procedure in Section 3.4.1 and prove the above guarantee on the quality of the solution returned and the running time.

By our earlier discussion, one can always choose R and V so that $\ln\left(\frac{R}{V}\right)$ is polynomial in the input size. In Section 3.4.2 we show that for the stochastic set cover problem given by the formulation (SSC-P2), one can compute ω-subgradients (with a sufficiently high probability), and can set the parameter K, so that the entire procedure runs in polynomial time, and delivers a solution of cost at most $OPT/(1-\gamma)+\epsilon$

with high probability. To convert this to a purely multiplicative guarantee we use a procedure ConvOpt to bootstrap algorithm FindOpt. In procedure ConvOpt we first sample a certain number of times from the distribution on scenarios, and use the samples to determine with high probability that either, $x = \mathbf{0}$ is an optimal solution and return this solution, or obtain a lower bound on OPT and then call FindOpt, with an appropriate setting of the parameters γ and ϵ. Wrapping FindOpt within this initial sampling procedure allows us to assume that FindOpt executes only if OPT is "large," and therefore set γ and ϵ so that FindOpt returns a solution of cost at most $(1+\kappa)\cdot OPT$. We describe this procedure ConvOpt in Section 3.4.2, and prove that it returns a $(1+\kappa)$-optimal solution with high probability.

In Section 3.5 we generalize the arguments of Section 3.4.2 to show that ω-subgradients can be computed for a large class of 2-stage stochastic programs, and therefore, procedures FindOpt and ConvOpt can be used to find a $(1+\kappa)$-optimal solution.

3.4.1 The Generic Algorithm using ω-Subgradients

We now describe algorithm FindOpt. The algorithm without procedure FindMin is an adaptation of ellipsoid-based algorithms for convex optimization that have been studied in the mathematical programming literature. Procedure FindMin is responsible for finding a feasible point of cost comparable to $\min_i h(x_i)$ using ω-subgradients.

FindOpt(γ, ϵ)

[Returns a point $\bar{x}$ such that $h(\bar{x}) \le OPT/(1-\gamma)+\epsilon$. Assume $\gamma \le \frac{1}{2}$.]

O1. Set $k \leftarrow 0,\ y_0 \leftarrow \mathbf{0},\ N \leftarrow 2m^2 \ln\left(\frac{16KR^2}{V\epsilon}\right),\ n \leftarrow N \log\left(\frac{8NKR}{\epsilon}\right)$, and $\omega \leftarrow \gamma/2n$. Let $E_0 \leftarrow B(\mathbf{0}, R)$ and $\mathcal{P}_0 \leftarrow \mathcal{P}$.

O2. For $i = 0, \ldots, N$ do the following.

[We maintain the invariant that E_i is an ellipsoid centered at y_i containing the current polytope $\mathcal{P}_k$.]

a) If $y_i \in \mathcal{P}_k$, set $x_k \leftarrow y_i$. Let $\hat{d}_k$ be a ω-subgradient of $h(.)$ at x_k. Let H

denote the half space $\{x \in \mathbb{R}^m : \hat{d}_k \cdot (x - x_k) \leq 0\}$. Set $\mathcal{P}_{k+1} \leftarrow \mathcal{P}_k \cap H$ and $k \leftarrow k + 1$.

b) If $y_i \notin \mathcal{P}_k$, let $a \cdot x \leq b$ be a violated inequality, that is, $a \cdot y_i > b$, whereas $a \cdot x \leq b$ for all $x \in \mathcal{P}_k$. Let H be the half space $\{x \in \mathbb{R}^m : a \cdot (x - y_i) \leq 0\}$.

c) Set E_{i+1} to be the ellipsoid of minimum volume containing the half-ellipsoid $E_i \cap H$.

O3. Let $k \leftarrow k - 1$. We now have a collection of points $x_0, \ldots, x_k$ such that each $x_l \in \mathcal{P}_l \subseteq \mathcal{P}_{l-1}$. Return $\mathsf{FindMin}(\omega; x_0, \ldots, x_k)$.

Clearly we maintain the invariant stated in the algorithm. Before describing procedure FindMin we show that $\min_{i=0}^k h(x_i)$ is close to OPT. We will need the following well-known facts (see for example [27]).

Fact 3.4.2 *The volume of the ball $B(u, D) = \{x \in \mathbb{R}^m : \|x - u\| \leq D\}$ where $u \in \mathbb{R}^m, D \geq 0$ is $D^m V_m$ where V_m is the volume of the unit ball $B(\mathbf{0}, 1)$ in $\mathbb{R}^m$.*

Fact 3.4.3 *Let $E \subseteq \mathbb{R}^m$ be an ellipsoid and $H \subseteq \mathbb{R}^m$ be a half space passing through the center of E. Then there is a unique ellipsoid E' of minimum volume containing the half-ellipsoid $E \cap H$ and $\frac{\mathsf{vol} E'}{\mathsf{vol} E} \leq e^{-1/(2m)}$.*

Fact 3.4.4 *Let $T : \mathbb{R}^m \mapsto \mathbb{R}^m$ be an affine transformation with $T(x) = Qx + t$, where $\det Q \neq 0$. Then for any set $S \subseteq \mathbb{R}^m$ we have $\mathsf{vol}(T(S)) = |\det Q| \mathsf{vol}(S)$.*

Lemma 3.4.5 *The points $x_0, \ldots, x_k$ generated by FindOpt satisfy $\min_{i=0}^k h(x_i) \leq \left(OPT + \frac{\epsilon}{4}\right)/(1 - \omega)$.*

Proof : Let x^* be an optimal solution. If $\hat{d}_l \cdot (x^* - x_l) \geq 0$ for some l, then $h(x_l) \leq h(x^*)/(1 - \omega)$ since $\hat{d}_l$ is a ω-subgradient at x_l. Otherwise let $r = \frac{\epsilon}{8KR}$. Consider the *affine transformation* T defined by $T(x) = rI_m(x - x^*) + x^* = rx + (1 - r)x^*$ where I_m is the $m \times m$ identity matrix, and let $W = T(\mathcal{P})$, so W is a shrunken version of $\mathcal{P}$ "centered" around x^*. Observe the following facts: (1) $W \subseteq \mathcal{P}$ because $\mathcal{P}$ is convex, and any point $y = T(x) \in W$ is a convex combination of $x \in \mathcal{P}$ and $x^* \in \mathcal{P}$, so $y \in \mathcal{P}$; (2) $\mathsf{vol}(W) = r^m \mathsf{vol}(\mathcal{P}) \geq (rV)^m \mathsf{vol}(B(\mathbf{0},1))$ using Facts 3.4.4 and 3.4.3 and since $\mathcal{P}$ contains a ball of radius V by assumption; and (3) for any $y = Tx \in W$, $\|y - x^*\| = r\|x - x^*\| \leq \frac{\epsilon}{4K}$ since $x, x^* \in B(\mathbf{0}, R)$, so $h(y) \leq h(x^*) + \frac{\epsilon}{4}$ since $h(.)$ has Lipschitz constant K. Since $\frac{\mathsf{vol}(E_{i+1})}{\mathsf{vol}(E_i)} \leq e^{-1/(2m)}$ for every i, and the volume of the ball $E_0 = B(\mathbf{0}, R)$ is $R^m \mathsf{vol}\big(B(\mathbf{0},1)\big)$, plugging things together we obtain

$$\mathsf{vol}(\mathcal{P}_k) \leq \mathsf{vol}(E_N) \leq e^{-N/(2m)}\,\mathsf{vol}(E_0) = \left(\tfrac{rV}{2}\right)^m \mathsf{vol}(B(\mathbf{0},1)) < \mathsf{vol}(W),$$

so there must be a point $y \in W$ that lies on a boundary of $\mathcal{P}_k$ generated by a hyperplane $\hat{d}_l \cdot (x - x_l) = 0$. This implies that $h(x_l) \leq h(y)/(1-\omega) \leq \big(h(x^*) + \frac{\epsilon}{4}\big)/(1-\omega)$. ■

$\mathsf{FindMin}(\boldsymbol{\omega}; \boldsymbol{x_0}, \ldots, \boldsymbol{x_k})$

M1. Set $\rho \leftarrow \epsilon/4k$, $\bar{x} \leftarrow x_0$, $N' \leftarrow \log\big(\frac{8kKR}{\epsilon}\big)$.

M2. For $i = 1, \ldots, k$ do the following.

[We maintain the invariant that $h(\bar{x}) \leq \big(\min_{l=0}^{i-1} h(x_l) + (i-1)\rho\big)/(1-\omega)^{(i-1)N'}$.]

a) We use binary search to find y on the $\bar{x} - x_i$ line segment with value close to $\min(h(\bar{x}), h(x_i))$. Initialize $y_1 \leftarrow \bar{x}$, $y_2 \leftarrow x_i$.

b) For $j = 1, \ldots, N'$ do the following.

[We maintain that $h(y_1) \leq h(\bar{x})/(1-\omega)^{j-1}$, $h(y_2) \leq h(x_i)/(1-\omega)^{j-1}$.]

- Let $y \leftarrow \frac{y_1 + y_2}{2}$. Compute a ω-subgradient $\hat{d}$ of h at the point y. If $\hat{d} \cdot (y_1 - y_2) = 0$, then exit the loop. Otherwise exactly one of $\hat{d} \cdot (y_1 - y)$ and $\hat{d} \cdot (y_2 - y)$ is positive.
- If $\hat{d} \cdot (y_1 - y) > 0$, set $y_1 \leftarrow y$, else set $y_2 \leftarrow y$.

c) Set $\bar{x} \leftarrow y$.

M3. Return $\bar{x}$.

Lemma 3.4.6 *Procedure* $\mathsf{FindMin}$ *returns a point* $\bar{x}$ *such that* $h(\bar{x}) \leq \big(\min_{i=0}^{k} h(x_i) + \frac{\epsilon}{4}\big)/(1-\omega)^{kN'}$.

Proof : The proof follows from the invariant stated in step M2 with $i = k+1$, so we show the invariant. The invariant clearly holds when $i = 1$. Suppose the invariant holds at the beginning of iteration i. If we show that the inner "For j=..." loop returns a point y such that $h(y) \leq \min(h(\bar{x}), h(x_i))/(1-\omega)^{N'} + \rho$, then after we set $\bar{x} \leftarrow y$ in step M2c) at the end of iteration i, we get that $h(\bar{x}) \leq (\min_{l=0}^{i} h(x_l) + i\rho)/(1-\omega)^{iN'}$, so the invariant is satisfied at the beginning of iteration $i+1$.

To prove the claim about the inner loop, first notice that if at any point we have $\hat{d} \cdot (y_1 - y_2) = 0$, then since y_1, y_2 and y all lie on the $\bar{x} - x_i$ line segment, we also have $\hat{d} \cdot (\bar{x} - y) = \hat{d} \cdot (x_i - y) = 0$. This implies that $h(y) \leq \min(h(\bar{x}), h(x_i))/(1-\omega)$ and in this case the claim holds. So assume that this is not the case. We will show by induction that $h(y_1) \leq h(\bar{x})/(1-\omega)^{j-1}$ and $h(y_2) \leq h(x_i)/(1-\omega)^{j-1}$ at the start of the j^{th} iteration of the inner loop. This is true at the beginning of the inner loop when $j = 1$. Suppose that this is true for iterations $1, \ldots, j-1$. So we have, $h(y_1) \leq h(\bar{x})/(1-\omega)^{j-2}$ and $h(y_2) \leq h(x_i)/(1-\omega)^{j-2}$ at the start of the $(j-1)^{th}$ iteration. In iteration $j-1$, we set $y = \frac{y_1+y_2}{2}$ and either $\hat{d} \cdot (y_1 - y) > 0$ or $\hat{d} \cdot (y_2 - y) > 0$. In the former case, we have $h(y) \leq h(y_1)/(1-\omega) \leq h(\bar{x})/(1-\omega)^{j-1}$ and we update $y_1 \leftarrow y$; similarly, in the latter case we have $h(y) \leq h(x_i)/(1-\omega)^{j-1}$ and we update $y_2 \leftarrow y$. So in either case, at the beginning of the j^{th} iteration we maintain the invariant that $h(y_1) \leq h(\bar{x})/(1-\omega)^{j-1}$ and $h(y_2) \leq h(x_i)/(1-\omega)^{j-1}$, and by induction the invariant holds through all iterations. After iteration N' finishes, we have $\|y - y_1\|$, $\|y - y_2\|$ both at most $\frac{\|\bar{x} - x_i\|}{2^{N'}} \leq \rho/K$, since $\bar{x}$ and x_i both lie in $\mathcal{P} \subseteq B(\mathbf{0}, R)$ and hence $\|\bar{x} - x_i\| \leq 2R$, which implies that $h(y) \leq \min(h(y_1), h(y_2)) + \rho \leq \min(h(\bar{x}), h(x_i))/(1-\omega)^{N'} + \rho$. This proves the claim about the inner loop on j, and hence the lemma. ■

Theorem 3.4.7 *Algorithm* FindOpt *returns a feasible point $\bar{x}$ satisfying $h(\bar{x}) \leq OPT/(1-\gamma) + \epsilon$ in time $\operatorname{poly}\big(m, \ln(\frac{KRm}{V\epsilon})\big) \cdot T(\omega)$, where $T(\omega)$ denotes the time taken to compute an ω-subgradient and $\omega = \Theta\big(\gamma/\operatorname{poly}(m, \ln(\frac{KRm}{V\epsilon}))\big)$.*

Proof : By Lemmas 3.4.5 and 3.4.6, we get that $h(\bar{x}) \leq \big(OPT + \frac{\epsilon}{2}\big)/(1-\omega)^{kN'+1}$. Since $kN' \leq N \log\big(\frac{8NKR}{\epsilon}\big) = n$ and $\omega = \gamma/2n$, we have $(1-\omega)^{kN'+1} \geq (1-\omega)^{n+1} \geq$

$(1-\gamma) \geq \frac{1}{2}$ (since we assumed $\gamma \leq \frac{1}{2}$) which proves the performance guarantee, and shows that $\omega = \Theta\big(\gamma/\operatorname{poly}(m, \ln(\frac{KRm}{V\epsilon}))\big)$. The running time is $O\big((N+n)T(\omega)\big)$ which is $O\big(T(\omega) \cdot m^2 \ln^2(\frac{KRm}{V\epsilon})\big)$. ■

3.4.2 Computing ω-Subgradients and Fixing Parameters

We now focus on the stochastic set cover problem given by the formulation (SSC-P2) and show that the FindOpt can be used to obtain a $(1+\kappa)$-optimal solution. Define $\lambda = \max\big(1, \max_S \frac{w_S^{\mathrm{II}}}{w_S^{\mathrm{I}}}\big)$. The procedure for computing ω-subgradients and its analysis proceeds as follows. In Lemma 3.4.8 we show that to compute a ω-subgradient of $h(.)$ at any point x, it suffices to find a vector $\hat{d}$ that component-wise approximates a subgradient at x to within a certain additive accuracy. Next, in Lemma 3.4.9 we show that at any point x, there is a "nice" subgradient d with components $d_S \in [-w_S^{\mathrm{II}}, w_S^{\mathrm{I}}]$. This will give us a bound on the Lipschitz constant K, and will show that since the components d_S lie in a range bounded multiplicatively by λ, $\operatorname{poly}\big(m, \lambda, \frac{1}{\omega}\big)$ samples suffice to compute an estimate $\hat{d}$ that component-wise approximates the subgradient d to within the desired accuracy with high probability, and thus obtain a ω-subgradient with high probability (Corollary 3.4.12). Using this procedure to compute ω-subgradients in procedure FindOpt, and by setting a small enough error probability in the ω-subgradient computations, one obtains a point $\bar{x}$, such that, $h(\bar{x}) \leq OPT/(1-\gamma) + \epsilon$ with probability at least $1-\delta$ in time $\operatorname{poly}\big(\text{input size}, \lambda, \frac{1}{\gamma}, \ln(\frac{1}{\delta})\big)$. This is shown in Lemma 3.4.13.

Lastly, we describe procedure ConvOpt where we sample initially to determine a lower bound on OPT before calling FindOpt. We show (Lemma 3.4.14) that the sampling in ConvOpt (correctly) determines, with probability at least $1-\delta$, either that $x = \mathbf{0}$ is an optimal solution, or that $OPT \geq \varrho/\lambda$ for a suitable ϱ. Thus by setting γ and ϵ appropriately (in the call to FindOpt), we get that ConvOpt finds a solution $\bar{x}$ of most $(1+\kappa) \cdot OPT$ with probability at least $1-2\delta$ in time $\operatorname{poly}\big(\text{input size}, \lambda, \frac{1}{\kappa}, \ln(\frac{1}{\delta})\big)$. Theorem 3.4.15 puts together these various components and proves the above statement. Recall that the objective function is $h(x) = w^{\mathrm{I}} \cdot x + f(x)$ where $f(x) = \sum_{A \subseteq U} p_A f_A(x)$, and

$$f_A(x) = \min\Big\{\sum_S w^{\mathrm{II}}_S r_{A,S} : \sum_{S:e\in S} r_{A,S} \geq 1 - \sum_{S:e\in S} x_S \quad \forall e \in A; \quad r_{A,S} \geq 0 \quad \forall S\Big\}.$$

By taking the dual, we can write $f_A(x) = \max\big\{\sum_e (1 - \sum_{S:e\in S} x_S) z_{A,e} : z_A \in \mathcal{Q}_A\big\}$ where $\mathcal{Q}_A$ is the polytope

$$\Big\{z_e : \sum_{e\in S} z_e \leq w^{\mathrm{II}}_S \text{ for all } S; \quad z_e = 0 \text{ for all } e \notin A; \quad z_e \geq 0 \text{ for all } e\Big\}.$$

The dual should only have variables $z_{A,e}$ for $e \in A$, however it is convenient to write it in this equivalent form.

Lemma 3.4.8 *Let d be a subgradient of $h(.)$ at the point $x \in \mathcal{P}$, and suppose $\hat{d}$ is a vector such that $d_S - \omega w^{\mathrm{I}}_S \leq \hat{d}_S \leq d_S$ for all S. Then $\hat{d}$ is a ω-subgradient of $h(.)$ at x.*

Proof : Let y be any point in $\mathcal{P}$. Since the polytope $\mathcal{P}$ has $x \geq \mathbf{0}$ as a defining constraint, it follows that $x_S, y_S \geq 0$ for all S. We have $h(y) - h(x) \geq d \cdot (y - x) = \hat{d} \cdot (y-x) + (d - \hat{d}) \cdot (y - x)$, so we need to lower bound the second term by $-\omega h(x)$. Since $d_S - \hat{d}_S \geq 0$ and $y_S \geq 0$ for every S, $(d - \hat{d}) \cdot (y - x) \geq -(d - \hat{d}) \cdot x$. Now $d_S - \hat{d}_S \leq \omega w^{\mathrm{I}}_S$ and $x_S \geq 0$ for every S, so $-(d - \hat{d}) \cdot x \geq -\sum_S \omega w^{\mathrm{I}}_S x_S \geq -\omega h(x)$ (since $f(x) \geq 0$ always). So $(d - \hat{d}) \cdot (y - x) \geq -\omega h(x)$, completing the proof. ∎

Lemma 3.4.9 *Consider any point $x \in \mathbb{R}^m$, and let z^*_A be an optimal dual solution for the scenario A minimization problem with x as the stage I decision vector. The vector d with components $d_S = w^{\mathrm{I}}_S - \sum_A p_A \sum_{e\in S} z^*_{A,e}$ is a subgradient at x, and $\|d\| \leq \lambda \|w^{\mathrm{I}}\|$.*

Proof : Let y be any point in $\mathbb{R}^m$. We have to show that $h(y) - h(x) \geq d \cdot (y - x)$. We know that $f_A(x) = \sum_e (1 - \sum_{S:e\in S} x_S) z^*_{A,e}$ for every scenario A. Also, since $z^*_A \in \mathcal{Q}_A$, at point y,we have $f_A(y) \geq \sum_e (1 - \sum_{S:e\in S} y_S) z^*_{A,e}$ for every scenario A. So $h(y) \geq w^{\mathrm{I}}_S \cdot y + \sum_{A\subseteq U} p_A \big(\sum_e (1 - \sum_{S:e\in S} y_S) z^*_{A,e}\big)$. The last term can be rewritten as

$$\sum_{A\subseteq U,e} p_A z^*_{A,e} - \sum_{A\subseteq U} p_A \sum_e \sum_{S:e\in S} y_S z^*_{A,e} = \sum_{A\subseteq U,e} p_A z^*_{A,e} - \sum_{A\subseteq U} p_A \sum_S y_S \Big(\sum_{e\in S} z^*_{A,e}\Big),$$

therefore we get that $h(y) \geq \sum_S y_S\big(w^{\mathrm{I}}_S - \sum_{A\subseteq U} p_A \sum_{e\in S} z^*_{A,e}\big) + \sum_{A\subseteq U,e} p_A z^*_{A,e}$. We can express $h(x)$ in a similar way with an equality instead of the inequality, replacing y_S with x_S. Subtracting the two terms, we get that $h(y) - h(x) \geq \sum_S (y_S - x_S)d_S$ where $d_S = w^{\mathrm{I}}_S - \sum_{A\subseteq U} p_A \sum_{e\in S} z^*_{A,e}$. To bound $\|d\|$, since $z^*_{A,e} \geq 0$ for all A, e, we have $d_S \leq w^{\mathrm{I}}_S$. Also, observe that $w^{\mathrm{I}}_S - w^{\mathrm{II}}_S \leq d_S$, since $\sum_{e\in S} z^*_{A,e} \leq w^{\mathrm{II}}_S$ for every scenario A, and $\sum_{A\subseteq U} p_A = 1$, because some scenario has to materialize (recall that we include the empty set also as a scenario). Therefore $|d_S| \leq \lambda w^{\mathrm{I}}_S$, and hence $\|d\| \leq \lambda \|w^{\mathrm{I}}\|$. ∎

Claim 3.4.10 *Suppose $\|d(x)\| \leq K$ for every x, where $d(x)$ is a subgradient of $h(.)$ at point x. Then $h(.)$ has Lipschitz constant (at most) K.*

Proof : Consider any two points $u, v \in \mathbb{R}^m$ and let d, d' denote the subgradients at u, v respectively, with $\|d\|, \|d'\| \leq K$. We have

$$h(v) - h(u) \geq d\cdot(v-u) \geq -\|d\|\,\|v-u\| \geq -K\|v-u\|,$$

and similarly $h(u) - h(v) \geq -\|d'\|\,\|u-v\| \geq -K\|u-v\|$. ∎

Claim 3.4.10 and Lemma 3.4.9 show that we can set the Lipschitz constant K to $\lambda\|w^{\mathrm{I}}\|$. Note that $\ln K$ is polynomially bounded. Observe also, that although we have already argued that $\ln\big(\frac{R}{V}\big)$ is polynomially bounded, for the stochastic set cover problem, since the polytope $\mathcal{P}$ is just the unit cube, we can set $R = \sqrt{m}$ and $V = 0.5$, and get a much improved bound. We next state a sampling lemma that we will use to show that at any point x, we can compute a ω-subgradient of $h(.)$ with probability at least $1-\delta$, using $O\big(\frac{\lambda^2}{\omega^2}\ln(\frac{m}{\delta})\big)$ samples.

Lemma 3.4.11 *Let $X \in [-a, b]$ be a random variable, $a, b > 0$, computed by sampling from a probability distribution π. Let $\mu = \mathrm{E}\big[X\big]$ and $\alpha = \max(1, a/b)$. Then for any $c > 0$, by taking $\frac{100\alpha^2}{3c^2}\ln\big(\frac{1}{\delta}\big)$ independent samples from π, one can compute an estimate Y such that $\mu - 2c\cdot b \leq Y \leq \mu$ with probability at least $1-\delta$.*

Proof : Let $q = \max(a, b)$. The variance of X is $\sigma^2 = \mathrm{E}\big[X^2\big] - \mu^2 \leq q^2$. We divide the samples into $s_1 = \frac{20}{3}\ln\big(\frac{1}{\delta}\big)$ groups, each group containing $s_2 = 5\alpha^2/c^2$ samples. Let X_{ij} be the value of X computed from the j^{th} sample of group i, $i = 1, \ldots, s_1$, $j = 1, \ldots, s_2$. Let Y_i be the average of the X_{ij} values. We set $Y = \mathrm{median}(Y_1, \ldots, Y_{s_1}) - c \cdot b$. The variables X_{ij} are iid with mean μ and variance σ^2. So we have $\mathrm{E}\big[Y_i\big] = \mu$ and $\mathrm{Var}\big[\mathrm{Y_i}\big] = \sigma^2/s_2$. By Chebyshev's inequality, we get $\Pr[|Y_i - \mu| > c \cdot b] \leq \frac{\sigma^2}{s_2(cb)^2} \leq \frac{\alpha^2}{s_2 c^2} \leq \frac{1}{5}$. Let $Z_i = 1$ if $|Y_i - \mu| > c \cdot b$, and 0 otherwise, and $Z = \sum_{i=1}^{s_1} Z_i$. Then $\mathrm{E}\big[Z\big] \leq s_1/5$ and the variables Z_i are independent. If $Y > \mu$ or $Y < \mu - 2c \cdot b$, then at least $s_1/2$ variables Z_i must be set to 1. Therefore by Chernoff bounds we have $\Pr\big[Y \notin [\mu - 2c \cdot b, \mu]\big] \leq \exp(-\frac{3s_1}{20}) \leq \delta$. ■

Corollary 3.4.12 *At any point $x \in \mathcal{P}$, one can compute a ω-subgradient with probability at least $1 - \delta$ using at most $\frac{400\lambda^2}{3\omega^2}\ln\big(\frac{m}{\delta}\big)$ independent samples from the probability distribution on scenarios.*

Proof : The proof is an easy corollary of Lemmas 3.4.9, 3.4.8 and 3.4.11. We use the sampling process described in Lemma 3.4.11. Each time we sample and get a scenario A, we compute the quantities $X_S = w_S^{\mathrm{I}} - \sum_{e \in S} z_{A,e}^*$ where z_A^* is an optimal dual solution for scenario A, with x as the first-stage vector. The proof now follows from Lemma 3.4.9, Lemma 3.4.11 with error probability δ/m and $c = \omega/2$, and Lemma 3.4.8. Observe that if $d_S = \mathrm{E}\big[X_S\big]$ then $d_S = w_S^{\mathrm{I}} - \sum_A p_A \sum_{e \in S} z_{A,e}^*$, so the vector d with components d_S is a subgradient at x by Lemma 3.4.9. Since $X_S \in [-w_S^{\mathrm{II}}, w_S^{\mathrm{I}}]$ for each S, using Lemma 3.4.11 with error probability δ/m and $c = \omega/2$, we can estimate the expectation $\mathrm{E}\big[X_S\big] = d_S$ by $\hat{d}_S$ using the claimed number of samples, so that for each S individually, we have $d_S - \omega w_S^{\mathrm{I}} \leq \hat{d}_S \leq d_S$ with probability at least $1 - \delta/m$. So the error probability over all sets S is at most δ, that is, $\Pr[\forall S,\, d_S - \omega w_S^{\mathrm{I}} \leq \hat{d}_S \leq d_S] \geq 1 - \delta$. So if $\hat{d} = \{\hat{d}_S\}$ is the resulting vector of estimates then by Lemma 3.4.8 $\hat{d}$ is a ω-subgradient at x with probability at least $1 - \delta$. ■

We can plug in the time required to compute a ω-subgradient (with a sufficiently small error probability) in Theorem 3.4.7 to get the following.

Lemma 3.4.13 *Using the above procedure for computing ω-subgradients, algorithm* FindOpt *finds a feasible solution $\bar{x}$ such that $h(\bar{x}) \leq OPT/(1-\gamma)+\epsilon$ with probability at least $1-\delta$ in time* $\text{poly}\big(\textit{input size}, \frac{1}{\gamma}, \ln(\frac{1}{\epsilon}), \ln(\frac{1}{\delta})\big)$.

Proof : Theorem 3.4.7 gives the performance guarantee and accounts for the time taken excluding the time taken to compute ω-subgradients. We need to show that with high probability *every* vector the algorithm computes is a ω-subgradient for $\omega = \gamma/2n$ where $n = N\log\big(\frac{8NKR}{\epsilon}\big)$, $N = 2m^2\ln\big(\frac{16KR^2}{V\epsilon}\big)$. The total number of times we need to compute a ω-subgradient is at most $N+n$. Setting the error probability to $\delta/(N+n)$, and $\omega = \gamma/2n$ in Corollary 3.4.12, we get that $O\big(\frac{\lambda^2 n^2}{\gamma^2}\ln(\frac{m(N+n)}{\delta})\big)$ samples suffice to ensure that each individual vector computed is a ω-subgradient with probability at least $1-1/((N+n)\delta)$. So the overall error probability over all ω-subgradient computations is at most δ. The time taken is $O\big((N+n)(\text{time to compute a }\omega\text{-subgradient})\big)$ which is $O\big(n^3\lambda^2(\ln N + \ln(\frac{1}{\delta}))/\gamma^2\big)$, hence polynomial in the input size, $\frac{1}{\gamma}$, $\ln\big(\frac{1}{\epsilon}\big)$ and $\ln\big(\frac{1}{\delta}\big)$. ■

Now we describe procedure ConvOpt that bootstraps FindOpt, and summarize the entire algorithm below.

ConvOpt(κ, δ)

[Returns $\bar{x}$ such that $h(\bar{x}) \leq (1+\kappa)\cdot OPT$ with high probability. Assume $\delta \leq \frac{1}{2}$.]

C1. Sample $M = \lambda\ln\big(\frac{1}{\delta}\big)$ times from the distribution on scenarios. Let X denote the number of times that a non-null scenario occurs.

C2. If $X = 0$, return $x = \mathbf{0}$ as an optimal solution.

C3. Otherwise (with probability at least $1-\delta$), $OPT \geq \varrho/\lambda$, where $\varrho = \frac{\delta}{\ln(1/\delta)}$. Set $\epsilon = \kappa\varrho/(2\lambda)$, $\gamma = \kappa/3$. Return FindOpt (γ,ϵ).

We make the very mild assumption that in a non-null scenario, the cost of any solution is at least 1, that is, $w^{\mathrm{I}}\cdot x + f_A(x) \geq 1$, at any point x, for every scenario $A \neq \emptyset$. Note that with integer costs, this is simply saying that in any non-null scenario, we incur a non-zero total cost. (The constant 1 may be replaced by any constant c by adjusting the number of samples required by $\mathsf{ConvOpt}$ accordingly.)

Lemma 3.4.14 *By drawing $M = \lambda \ln\left(\frac{1}{\delta}\right)$ samples, with probability at least $1-\delta$, $\mathsf{ConvOpt}$ (correctly) determines that $OPT \geq \varrho/\lambda$ where $\varrho = \frac{\delta}{\ln(1/\delta)}$, or that $x = \mathbf{0}$ is an optimal solution.*

Proof : Note that $\varrho \leq 1$ since $\delta \leq \frac{1}{2}$. Since in every non-null scenario, we incur a cost of at least 1, $OPT \geq q$, where $q = \sum_{A \subseteq U, A \neq \emptyset} p_A$ is the probability of occurrence of a non-null scenario. Let $r = \Pr[X = 0] = (1-q)^M$. So $r \leq e^{-qM}$ and $r \geq 1 - qM$. If $q \geq \ln\left(\frac{1}{\delta}\right)/M$, then $\Pr[X = 0] \leq \delta$. So with probability at least $1-\delta$ we will say that $OPT \geq \varrho/\lambda$ which is true since $OPT \geq q \geq \frac{1}{\lambda}$. If $q \leq \delta/M$, then $\Pr[X = 0] \geq 1 - \delta$. We return $x = \mathbf{0}$ as an optimal solution with probability at least $1-\delta$ which is indeed an optimal solution, because $q \leq \frac{1}{\lambda}$ implies that it is always at least as good to defer to stage II since the expected stage II cost of a set S is at most $q \cdot w_S^{\mathrm{II}} \leq w_S^{\mathrm{I}}$. If $\delta/M < q < \ln\left(\frac{1}{\delta}\right)/M$, then we always return a correct answer since it is both true that $x = \mathbf{0}$ is an optimal solution, and that $OPT \geq q \geq \varrho/\lambda$. ■

Combining Lemma 3.4.13 and Lemma 3.4.14 gives the following theorem.

Theorem 3.4.15 *Procedure $\mathsf{ConvOpt}$ computes a feasible solution to* (SSC-P2) *of cost at most $(1+\kappa) \cdot OPT$ with probability at least $1 - 2\delta$ in time polynomial in the input size, $\frac{1}{\kappa}$, and $\ln\left(\frac{1}{\delta}\right)$.*

Proof : By Lemma 3.4.14, we know that if $\mathsf{ConvOpt}$ calls $\mathsf{FindOpt}$ then, with probability at least $1-\delta$, we have $OPT \geq \varrho/\lambda$ where $\varrho = \frac{\delta}{\ln(1/\delta)}$. The performance guarantee and the time bound now follow from Lemma 3.4.13. Since $\mathsf{FindOpt}$ may err with probability at most δ, the net error probability is at most 2δ. ■

3.5 A General Class of Solvable Stochastic Programs

We show that algorithm ConvOpt can be used to solve the following broad class of 2-stage stochastic programs.

$$\min \quad w^{\mathrm{I}} \cdot x + f(x) \quad \text{subject to} \quad x \geq \mathbf{0},\ x \in \mathcal{P} \subseteq \mathbb{R}^m, \tag{Stoc-P}$$

$$\text{where} \quad f(x) = \sum_{A \in \mathcal{A}} p_A f_A(x), \quad \text{and} \tag{3}$$

$$\begin{aligned} f_A(x) \ = \ \min \quad & w^A \cdot r_A + q^A \cdot s_A \\ \text{s.t.} \quad B^A s_A & \geq h^A & (4) \\ D^A s_A + T^A r_A & \geq j^A - T^A x & (5) \\ r_A, s_A & \geq 0,\ r_A \in \mathbb{R}^m,\ s_A \in \mathbb{R}^n. \end{aligned}$$

Here $\mathcal{A}$ denotes the set of all possible scenarios, and $\mathcal{P}$ is the feasible region polytope. We require that (a) $T^A \geq \mathbf{0}$ for every scenario A, and (b) at every feasible point $x \in \mathcal{P}$, $f(x) \geq 0$ and that the primal and dual problems corresponding to $f_A(x)$ are feasible. A sufficient condition for (b) is to insist that $0 \leq f_A(x) < +\infty$ at every point $x \in \mathcal{P}$ and scenario $A \in \mathcal{A}$.

Remark 3.5.1 We can relax condition (a) somewhat and solve a more general class of programs, for example, programs with upper bounds on the second-stage decisions r_A under certain conditions (where the matrix T^A has negative entries). Such upper bounds are useful in problems with capacity constraints, such as the stochastic multi-commodity flow problem considered in [69] for which the resulting 2-stage stochastic program can be solved using algorithm ConvOpt. The class of programs captured by formulation (Stoc-P) suffices for the applications considered in this thesis.

The essential property of this class of programs is that in any scenario A, the same matrix T^A acts upon both the recourse vector r_A and the stage I vector x, implying that the stage I actions and the stage II actions play the same role. All of the stochastic optimization problems we will consider can be expressed as convex programs in the above form, and as we argue below, one can therefore obtain a near-optimal fractional solution for each of these problems in polynomial time. Observe that this class of stochastic programs is rich enough to model stochastic problems where the stage II recourse costs *may depend on the scenario that materializes.* For example in the stochastic set cover problem, this means that we could have stage II costs w_S^A depending on the scenario A. To prevent an exponential blowup in the input, we consider an oracle model where an oracle supplied with scenario A reveals the scenario-dependent data $\left(w^A, q^A, h^A, j^A, B^A, D^A, T^A\right)$; procedure ConvOpt will only need to query this oracle a polynomial number of times.

Let $h(.)$ denote the objective function. First we state the basic fact that the objective function of (Stoc-P) is convex. The proof of this is very similar to the proof of Lemma 3.2.1.

Lemma 3.5.2 *The objective function of* (Stoc-P) *is convex.*

Define $\lambda = \max\left(1, \max_{A \in \mathcal{A}, S} \frac{w_S^A}{w_S^I}\right)$. We assume that *the algorithm knows the value of* λ. To extend the analysis in Section 3.4.2 and argue that one can compute a near-optimal solution using procedure ConvOpt, we need to show the following three things: (1) one can compute a ω-subgradient in polynomial time, (2) the Lipschitz constant K can be set so that $\ln K$ is polynomially bounded, and (3) one can detect with high probability that OPT is large. The third requirement is easily handled by Lemma 3.4.14. Under the mild assumption that every "non-null" scenario $A \in \mathcal{A}$ incurs a cost of at least 1, Lemma 3.4.14 holds, and shows that by sampling $\lambda \ln\left(\frac{1}{\delta}\right)$ times one can determine with high probability, that either $OPT \geq \frac{\delta}{\ln(1/\delta)\lambda}$, or that $x = \mathbf{0}$ is an optimal solution. So we may assume as before that if FindOpt gets called, then $OPT \geq \frac{\delta}{\ln(1/\delta)\lambda}$ with probability at least $1 - \delta$.

The fact that one can compute ω-subgradients efficiently, and the bound on the Lipschitz constant, will both follow from an argument along the same lines as that in Section 3.4.2. We show that at any point, there is a subgradient with a nice structure, which will give a bound on the Lipschitz constant, and show that by approximating this subgradient component-wise, one can obtain a ω-subgradient. The following lemma shows that there is a subgradient whose components lie in a range bounded multiplicatively by λ. Combined with Lemma 3.4.11, this will allow us to compute an ω-subgradient with high probability by repeated sampling.

Lemma 3.5.3 *Consider any point $x \in \mathbb{R}^m$, and let (u_A^*, z_A^*) be an optimal dual solution for scenario A with x as the stage I decision vector, where z_A^* is the dual multiplier corresponding to inequalities* (5)*. Then, (i) the vector $d = w^{\mathrm{I}} - \sum_A p_A (T^A)^{\mathrm{T}} z_A^*$ is a subgradient at x, (ii) $\|d\| \leq \lambda \|w^{\mathrm{I}}\|$ and (iii) if $\hat{d}$ is a vector such that $d - \omega w^{\mathrm{I}} \leq \hat{d} \leq d$, then $\hat{d}$ is a ω-subgradient at x.*

Proof : By taking the dual, we can write $f_A(x) = h^A \cdot u_A^* + (j^A - T^A x) \cdot z_A^*$. For any other point y, (u_A^*, z_A^*) is a feasible dual solution for scenario A, given the stage I decision vector y. So $f_A(y) \geq h^A \cdot u_A^* + (j^A - T^A y) \cdot z_A^*$ and we have

$$h(y) \geq w^{\mathrm{I}} \cdot y + \sum_A p_A (h^A \cdot u_A^* + j^A \cdot z_A^* - y^{\mathrm{T}} (T^A)^{\mathrm{T}} z_A^*). \tag{6}$$

As $y^{\mathrm{T}}(T^A)^{\mathrm{T}} z_A^*$ is a scalar, we can replace it by its transpose $\big((T^A)^{\mathrm{T}} z_A^*\big)^{\mathrm{T}} y = \big((T^A)^{\mathrm{T}} z_A^*\big) \cdot y$. Substituting this in (6) and combining the terms with y, we get that $h(y) \geq (w^{\mathrm{I}} - \sum_A p_A (T^A)^{\mathrm{T}} z_A^*) \cdot y + \sum_A p_A (h^A \cdot u_A^* + j^A \cdot z_A^*)$. We can write a similar expression for $h(x)$ with equality instead of the inequality. Subtracting, we get that $h(y) - h(x) \geq d \cdot (y - x)$ where $d = w^{\mathrm{I}} - \sum_A p_A (T^A)^{\mathrm{T}} z_A^*$. This shows that d is a subgradient at x.

For every scenario $A \in \mathcal{A}$ we have $z_A^* \geq \mathbf{0}$, so $d \leq w^{\mathrm{I}}$ since $T^A \geq \mathbf{0}$. Observe that the dual of the scenario A (primal) optimization problem has the constraint $(T^A)^{\mathrm{T}} z_A \leq w^A$. Since z_A^* is a feasible dual solution, we have $(T^A)^{\mathrm{T}} z_A^* \leq w^A \leq \lambda w^{\mathrm{I}}$, and since $\sum_A p_A = 1$, this shows that, $d \geq w^{\mathrm{I}} - \lambda w^{\mathrm{I}}$. So we get that $\|d\| \leq \lambda \|w^{\mathrm{I}}\|$. Now by Claim 3.4.10 (which holds regardless of the function $h(.)$), one can set $K = \lambda \|w^{\mathrm{I}}\|$, so that $\ln K$ is polynomially bounded.

Finally to show part (iii), we proceed exactly as in Lemma 3.4.8. $h(y) - h(x) \geq$

$d \cdot (y - x) = \hat{d} \cdot (y - x) + (d - \hat{d}) \cdot (y - x)$. Since $x, y \geq \mathbf{0}$, we have

$$(d - \hat{d}) \cdot (y - x) \geq -(d - \hat{d}) \cdot x \geq -\omega w^{\mathrm{I}} \cdot x \geq -\omega h(x),$$

where the last inequality follows since $f(x) \geq 0$. ■

So as before, using Lemma 3.4.11, one can compute a ω-subgradient at any point x using $O\big(\frac{\lambda^2}{\omega^2} \ln(\frac{m}{\delta})\big)$ samples. Putting the various components together, as in Lemma 3.4.13 and Theorem 3.4.15, we get the following theorem.

Theorem 3.5.4 *Procedure* ConvOpt *can be used to obtain a feasible solution to* (Stoc-P) *of objective function value at most* $(1 + \kappa) \cdot OPT$ *with probability at least* $1 - 2\delta$*, in time polynomial in the input size,* $\frac{1}{\kappa}$*, and* $\ln\big(\frac{1}{\delta}\big)$*.*

3.5.1 2-Stage Programs with a Continuous Distribution

We now consider the class of 2-stage programs specified by (Stoc-P) where the second stage "scenario" is specified by a parameter ξ that is continuously distributed with probability density function $p(\xi)$. So the objective function is $h(x) = w^{\mathrm{I}} \cdot x + \mathrm{E}_{\xi}\big[f(x, \xi)\big]$, where $\mathrm{E}_{\xi}\big[f(x, \xi)\big] = \int p(\xi) f(x, \xi)\, d\xi$ and $f(x, \xi)$ is the cost of scenario ξ as determined by the minimization problem in (Stoc-P) with parameters $\big(w(\xi), q(\xi), h(\xi), j(\xi), B(\xi), D(\xi), T(\xi)\big)$. As before we assume that at every feasible point x and scenario ξ, (a) $T(\xi) \geq \mathbf{0}$, (b) $\int p(\xi) f(x, \xi)\, d\xi \geq 0$, and that the primal and dual problems corresponding to $f(x, \xi)$ are feasible.

We can show that procedure ConvOpt can be used to obtain a $(1 + \kappa)$-optimal solution to this class of 2-stage programs with continuously distributed second-stage parameters. As argued previously, this will follow if we can show that one can compute ω-subgradients in polynomial time, bound the Lipschitz constant K suitably, and obtain a lower bound on OPT. Define $\lambda = \max\big(1, \sup_{\xi, S} \frac{w(\xi)_S}{w^{\mathrm{I}}_S}\big)$. Again, we assume that the algorithm knows the value of λ. The statement and proof of Lemma 3.5.3 extend easily to the continuous setting by substituting each occurrence of $\sum_A p_A(\ldots)$ by $\int d\xi\, p(\xi)(\ldots)$. So at any point x, if $\big(u^*(\xi), z^*(\xi)\big)$ is an optimal solution for the dual problem corresponding to $f(x,\xi)$ with $z^*(\xi)$ being the dual multipliers for inequalities (5), then $d = w^{\mathrm{I}} - \int d\xi\, p(\xi)T(\xi)^{\mathrm{T}}z^*(\xi)$ is a subgradient at x.

Parts (ii) and (iii) of Lemma 3.5.3 hold as is, and one thus obtains a bound on the Lipschitz constant, and the fact that ω-subgradients can be computed by sampling. Finally, under the assumption that $w^{\mathrm{I}} \cdot x + f(x,\xi) \geq 1$ for every (x,ξ), we can detect that OPT is large using Lemma 3.4.14.

Theorem 3.5.5 *Procedure* ConvOpt *can compute a feasible solution to* (Stoc-P) *with a continuous distribution, of value at most* $(1+\kappa)\cdot OPT$ *in polynomial time.*

3.6 A Lower Bound on the Number of Samples Required

Notice that the running time of the algorithm (as also that of Gupta et al. [35]) depends on the parameter λ, the maximum ratio between the costs in the two stages. We argue that in the black box model, this dependence on λ is unavoidable; we will show that a performance guarantee of c for our discrete applications requires $\Omega(\lambda/c)$ samples. In contrast, in [69] it is shown that if one is given a slightly more powerful black box and a limited amount of information about the probability distribution, then this dependence can be avoided for a subclass of the stochastic programs discussed in Section 3.5 that includes stochastic covering problems such as the stochastic set cover problem.

The lower bound is attained on a rather simple instance of SSC with a single set and a single element, which is perhaps suggestive of the inherent hardness of stochastic combinatorial optimization problems. A variety of combinatorial optimization problems can be viewed as set covering problems, perhaps with additional constraints, and the lower bound also holds for these problems. The SSC instance has universe $U = \{e\}$ and just one set $S = U$, where $w^{\mathrm{I}}_S = 1,\ w^{\mathrm{II}}_S = \lambda$. Let p denote the probability that scenario $\{e\}$ occurs (which is unknown by the algorithm that samples from the distribution on scenarios). The only decision here is whether to buy set S in stage I or to defer buying the set to stage II. We formalize the computation of an algorithm on this instance as follows. Let $\mathcal{A}_N$ denote an algorithm that draws exactly N samples. Algorithm $\mathcal{A}_N$ does the following: it draws N samples, computes the number of times that scenario $\{e\}$ occurs (which is a random variable), and depending on this value decides either to pick set S in stage I or wait until stage II. Let $\mathcal{O}^*$ denote the value of the *integer* optimum solution.

Theorem 3.6.1 *If $\mathcal{A}_N$ returns a solution of cost at most $c \cdot \mathcal{O}^*$ with probability at least $1-\delta$ where $1 \leq c < \frac{\lambda}{2}$, then it must be that $N \geq \big(\lambda \ln(\frac{1}{\delta}-1)\big)/2c$. The bound holds even if $\mathcal{A}_N$ returns only a fractional solution of cost at most $c \cdot \mathcal{O}^*$.*

Proof : Let X be a random variable that denotes the number of times scenario $\{e\}$ occurs in the N samples. If $X=0$, then $\mathcal{A}_N$ must choose to defer to stage II, that is return the *integer solution* $x=0$, with probability at least $1-\delta$ (the algorithm may flip a coin to decide). Otherwise, with $p=0$, and hence, $\mathcal{O}^*=0$, $\mathcal{A}_N$ will pick (a non-zero fraction of) set S in stage I with probability at least δ, and thus incur a non-zero cost, that is, a cost greater than $c \cdot \mathcal{O}^*$ with probability at least δ. Choose any $\epsilon > 0$ such that $c \leq \lambda/\big(2(1+\epsilon)\big)$ and consider any $\epsilon' > 0$ where $\epsilon' \leq \epsilon$. Set $p = (1+\epsilon')c/\lambda \leq \frac{1}{2}$ and define $N_0(\epsilon') = \big(\lambda \ln(\frac{1}{\delta}-1)\big)/\big(2(1+\epsilon')c\big)$. Let $r = \Pr[X=0] = (1-p)^N > e^{-2pN}$ (since $p \leq \frac{1}{2}$). The optimal solution is to pick S in stage I, and incur a cost of 1. But if $N < N_0(\epsilon')$, then $r > e^{-2pN_0(\epsilon')} = \frac{\delta}{1-\delta}$, so with probability at least $(1-\delta)r > \delta$, $\mathcal{A}_N$ will choose the solution $x=0$ and incur a cost of $(1+\epsilon')c > c \cdot \mathcal{O}^*$. Therefore for $\mathcal{A}_N$ to satisfy the required performance guarantee we must have $N \geq N_0(\epsilon')$ for every $\epsilon' \in (0, \epsilon]$ which implies that $N \geq \big(\lambda \ln(\frac{1}{\delta}-1)\big)/2c$. ■

Corollary 3.6.2 *If algorithm $\mathcal{A}_N$ returns a (fractional) solution of expected cost at most $c \cdot \mathcal{O}^*$ where $1 \leq c < \frac{\lambda}{6}$, then it must be that $N \geq (\lambda \ln 2)/6c$.*

Proof : By Markov's inequality, $\mathcal{A}_N$ returns a solution of cost at most $3c \cdot \mathcal{O}^*$ with probability at least $\frac{2}{3}$. The claim now follows from Theorem 3.6.1. ■

Part III

Stochastic and Deterministic Applications

Chapter 4

Stochastic Uncapacitated Facility Location

4.1 Introduction

This chapter focuses on the 2-stage stochastic uncapacitated facility location (SUFL) problem and gives a constant-factor approximation algorithm for this problem.

In the deterministic uncapacitated facility location (UFL) problem, one assumes that the client demands and their locations are precisely known in advance, and we want to choose a subset of the locations at which to open facilities and assign clients to facilities, so as to minimize the sum of the facility opening costs and the client assignment costs. In many settings, some of the data, such as the demand or location of clients, may be uncertain or inexactly specified. For example, there may be several macro-economic factors influencing demand, such as the state of the economy, competition, technology, customer purchasing power etc., all of which may lead to uncertainty in the demand. But while one might not have exact information, one may still have some distributional information, such as the probability distribution on the demands of the clients (in addition to deterministic data for facility and assignment costs), on which to base initial (first-stage) decisions regarding which facilities to open. Once the actual input (the client demands) is realized according to the distribution, one has the opportunity to extend (in the second stage) the initial solution, incurring a certain recourse cost, where the recourse costs could depend on the scenario that materializes, and are usually greater than the original costs.

This motivates the 2-stage stochastic uncapacitated facility location problem: one may choose some facilities to open initially given only distributional information about the demand that needs to be served; then once we get to know the actual demands, one may choose to open some additional facilities, and the cost incurred in opening a facility in the second stage might be different, and higher, than the initial (first-stage) opening cost of the facility. For example, consider a classical UFL application, where facilities are warehouses that need to be set up to serve retail stores or customers. The demand from the customers might not be exactly known, but one may estimate from market surveys or simulation models, the likelihood of demands turning up, and may choose to set up some warehouses initially in anticipation of the demand. Once we know the actual demands, we have the option of setting up more warehouses (or raising inventory levels), but setting up a warehouse (or raising inventory levels) at the last minute to take care of excess demand might involve deploying various resources with a much smaller turnaround or lead time, and would thus incur a higher cost.

Formally, in the 2-stage stochastic uncapacitated facility location (SUFL) problem, we are given a set of facilities $\mathcal{F}$, a set of clients $\mathcal{D}$, and a probability distribution over the demands of the clients, i.e., we have a probability distribution on tuples $(d_1, \ldots, d_{|\mathcal{D}|})$ where $d_j \in \{0, 1, \ldots, D\}$, and D is some known upper bound on the demand. We can open some facilities in stage I, incurring a cost of f_i^{I} for opening facility i, then the actual scenario A with demands d_j^A is revealed, and we may choose to open some more facilities in stage II, paying f_i^A for each facility i that we open. The stage II cost incurred for scenario A is the total cost of opening the additional facilities in that scenario and the cost of assigning the clients (with non-zero demand) to open facilities. The aim is to minimize the total expected cost, that is, the sum of the stage I cost and the expected stage II cost, where the expectation is taken over all scenarios A according to the scenario distribution.

4.1.1 Summary of Results

We give a $(3.378 + \epsilon)$-approximation algorithm for SUFL, where ϵ can be made arbitrarily small, whose running time is polynomial in the size of the input and in $\frac{1}{\epsilon}$. This result holds without any restrictions on the first- and second-stage facility costs, and in the black-box model, that is, where one can merely sample scenarios according to the distribution on scenarios, but no direct information about the distributions is given.

There are two principal components that lead to this result. First, we formulate the stochastic problem as a compact convex programming problem (SUFL-P) that belongs to the class of stochastic programs discussed in Section 3.5, and therefore by Theorem 3.5.4, one can obtain a near optimal solution in polynomial time. Next, we show that, given an algorithm for deterministic UFL with certain properties, one can round this fractional solution and thus decide which facilities to open in stage I, losing only a small factor in the cost. We use the primal-rounding algorithm described in Section 2.4 to get a $(3.378 + \epsilon)$-approximation algorithm. This also proves a bound on the integrality gap of the convex programming formulation, i.e., the ratio between the optimal integer and fractional solution values. We show that this bound can be improved to 3.04 (by an existence argument that does not yield a 3.04-approximation algorithm).

We show that the rounding procedure can be adapted to yield approximation algorithms for the stochastic versions of some other facility location problems as well. In Section 4.4 we consider a stochastic version of the uncapacitated facility location problem with penalties, where one has the option of not satisfying the demand of a client in a scenario by incurring a certain penalty (which may be scenario-dependent). Building upon the earlier result, we obtain a $(4.378 + \epsilon)$-approximation algorithm for this problem. In a subsequent chapter, we use a variant of the rounding method to give an approximation algorithm for the stochastic facility location problem with service installation costs (Section 5.6).

4.1.2 Related Work

The 2-stage stochastic uncapacitated facility location problem falls into the 2-stage stochastic optimization with recourse model discussed in Chapter 3. The textbook of Birge and Louveaux [13] deals extensively with models and algorithms for this class of location problems. Although stochastic optimization problems, and in particular, 2-stage problems with recourse, have been well studied, it is only recently that they have been considered from the perspective of designing approximation algorithms with provable worst-case guarantees. Stougie & van der Vlerk [72], in their survey on approximations for stochastic integer programming that focuses mostly on 2-stage stochastic programs, mention the work of Dye, Stougie & Tomasgard [22] on a resource provisioning problem as the first example (and the only known example at the time of writing of that survey) of worst-case analysis of approximation algorithms for discrete 2-stage stochastic problems with recourse. The computer science community has very recently become interested in approximation algorithms for 2-stage stochastic problems. We mention briefly the models that have been considered, all of which entail restricting either the class of probability distributions, or the costs incurred in the two stages, and then talk about specific results that have been obtained in these models.

As discussed in Section 3.1.2, three ways of specifying the probability distribution on scenarios have been considered: (1) the polynomial-scenario model, where one assumes that there are only a polynomial number of scenarios that occur with positive probability, and these are explicitly enumerated; (2) the independent-activation model, where each element (client) is activated independently with a certain (known) probability; and (3) the black-box model, where nothing is assumed of the probability distribution other than the ability to draw independent samples from it. Some of the previous work has focused on the proportional-costs model, where one imposes the rather severe restriction that the weights in the two stages are proportional. For example, in SUFL, this implies, that the cost of *any* facility i in *every* scenario A is $\lambda \cdot f_i^{\mathrm{I}}$ for some parameter λ.

Ravi and Sinha [61] consider SUFL in the polynomial-scenario model and give an LP rounding based 8-approximation algorithm, that can handle scenario-dependent facility opening and client assignment costs, where the assignment cost in scenario A is $c_{ij}^A = \gamma^A c_{ij}$ for all i, j. Their rounding algorithm needs to know the optimal fractional solution for *each stage II scenario* which renders it unsuitable when there are exponentially many scenarios. In contrast our rounding scheme generates an integer solution, that is, decides which facilities to open in stage I, using *only the stage I fractional solution*. Thus, in conjunction with the algorithm in Section 3.4 that returns a near-optimal (stage I) solution to a fractional relaxation of the problem, this yields a polynomial time algorithm for SUFL. In the black-box model, Gupta et al. [35] gave an 8.45-approximation algorithm under the proportional-costs assumption, i.e., $f_i^A = \lambda f_i^{\mathrm{I}}$ for each $i \in \mathcal{F}$ and each scenario A. They also gave an improved algorithm with a performance guarantee of 6 when they focused on the independent-activation model. Previously these were the best known guarantees; no approximation algorithm was known for SUFL in the black-box model *and* with arbitrary, scenario-dependent costs.

We also briefly sketch some related work on approximation algorithms design for a few other combinatorial optimization problems in the 2-stage stochastic framework (with restrictions on the probability distribution or on the costs). The approximation result of [22] is obtained in the polynomial-scenario model. Ravi & Sinha [61] consider stochastic versions of some other problems such as vertex cover, set cover, and Steiner tree, in the polynomial-scenario model. Independently, Immorlica, Karger, Minkoff, and Mirrokni [39] considered some of these problems, in both the polynomial-scenario model, and also in the independent-activation model introduced by them, but here they restrict attention to the proportional-costs setting. Gupta et al. [35] also rely on the proportional-costs assumption, but do not make any assumptions about the distribution; that is, they consider problems in the black-box model and give approximation algorithms for the stochastic versions of the vertex cover, Steiner tree and uncapacitated facility location problems.

The results in this chapter are from Shmoys & Swamy [69] who present the first approximation algorithms for some of the above problems, and others, in the black-box model with no restrictions on the costs, and gives a general technique for lifting guarantees that can be proved in the deterministic case to the stochastic setting. Among the problems considered are stochastic versions of set cover, vertex cover, facility location and some variants of it, multicommodity flow, and the multicut problem on trees.

4.2 The 2-Stage Stochastic Program

One can consider the following convex programming relaxation for SUFL. We use i to index the facilities in $\mathcal{F}$ and j to index the clients in $\mathcal{D}$. Let $\mathcal{A}$ denote the set of all possible scenarios. Throughout we will use A to index the scenarios, and p_A (which could be 0) to denote the probability of scenario A.

$$\min \quad \sum_i f_i^{\mathrm{I}} y_i + g(y) \quad \text{subject to } 0 \leq y_i \leq 1 \qquad \text{for all } i, \tag{SUFL-P}$$

$$\begin{aligned}
\text{where} \quad g(y) &= \sum_{A \in \mathcal{A}} p_A g_A(y), \\
\text{and} \quad g_A(y) &= \min \quad \sum_i f_i^A y_{A,i} + \sum_j d_j^A \sum_i c_{ij} x_{A,ij} \\
&\quad \text{s.t.} \quad \sum_i x_{A,ij} \geq 1 && \text{for all } j \text{ such that } d_j^A > 0, \\
&\qquad\qquad x_{A,ij} \leq y_i + y_{A,i} && \text{for all } i, j, \\
&\qquad\qquad x_{A,ij}, y_{A,i} \geq 0 && \text{for all } i, j.
\end{aligned}$$

Here y_i indicates if facility i is opened in stage I and $y_{A,i}$ indicates if facility i is opened in scenario A in stage II. The variables $x_{A,ij}$ are the usual assignment variables indicating whether client j is assigned to facility i in scenario A. The minimization problem for a scenario A determines the cost, $g_A(y)$, incurred for scenario A and has constraints that enforce that each client j with positive demand d_j^A has to be assigned to a facility that is opened either in stage I or in scenario A. The term $g(y)$ is therefore the expected second-stage cost. Observe that (SUFL-P) lies in the class of 2-stage stochastic programs handled by Theorem 3.5.4, and therefore one can use the algorithm described in Section 3.4 to obtain a solution y of cost at most $(1+\epsilon)$ times the optimal in time polynomial in the size of the input and $\frac{1}{\epsilon}$.

4.3 The Main Theorem

We prove the following theorem in this section. Let ρ_{UFL} denote the integrality gap of UFL which is at most 1.52 [56].

Theorem 4.3.1 *There is a* $(3.378+\epsilon)$*-approximation algorithm for SUFL based on rounding a (near-)optimal solution to* (SUFL-P)*. Moreover, the integrality gap of* (SUFL-P) *is at most* $2\rho_{\text{UFL}} \approx 3.04$*. These results hold even when the second-stage assignment costs* c_{ij} *are scenario-dependent with* $c^A_{ij} = \sigma^A c_{ij}$*.*

4.3.1 The 2-Stage Stochastic Set Cover Problem Revisited

Before we proceed to prove Theorem 4.3.1, we explain the basic rounding idea by considering the 2-stage stochastic set cover problem (SSC) and its formulation given by (SSC-P2) in Section 3.2. Recall that in this problem, we are given a universe of elements $U = \{e_1, \ldots, e_n\}$, a collection of subsets $S_1, \ldots, S_m \subseteq U$, and a probability distribution over subsets of U that determines which subset of elements has to be covered. We want to decide which sets to select in stage I so as to minimize the cost of picking sets in stage I and the expected cost of choosing sets in a stage II scenario. Each set has an *a priori* weight w^{I}_S and an *a posteriori* weight w^A_S that may depend on the scenario A that materializes. Formulation (SSC-P2) seeks to

$$\text{minimize } \sum_S w^{\text{I}}_S x_S + \sum_{A \subseteq U} p_A f_A(x) \quad \text{subject to} \quad 0 \le x_S \le 1 \text{ for all } S,$$

where $f_A(x)$ is given by

$$f_A(x) = \min\Big\{\sum_S w^A_S r_{A,S} : \sum_{S:e\in S} r_{A,S} \ge 1 - \sum_{S:e\in S} x_S \ \ \forall e \in A; \quad r_{A,S} \ge 0 \ \ \forall S\Big\}.$$

The following theorem forms the basis of our methodology for tackling various 2-stage stochastic optimization problems. Let OPT_{Det} denote the optimal value of the LP relaxation (SC-P) for the deterministic set cover problem (Section 3.2), and OPT denote the optimal value of (SSC-P2).

Theorem 4.3.2 *Suppose that we have a procedure that for every instance of the deterministic set cover problem, produces a solution of cost at most $\rho \cdot OPT_{Det}$. Then, one can convert any (fractional) solution x to* (SSC-P2) *to an integer solution losing a factor of at most 2ρ. Thus, a $(1+\epsilon)$-optimal solution to* (SSC-P2) *gives a $(2\rho+\epsilon)$-approximation algorithm.*

Proof : Let r_A denote an optimal solution to the scenario A minimization problem given the first stage vector x, so $f_A(x) = \sum_S w_S^A r_{A,S}$. Let $h(.)$ denote the objective function. We will argue that one can obtain an integer solution $\tilde{x}$, that is, the sets to pick in stage I, of cost no more than $2\rho \cdot h(x)$. Observe the following simple fact: an element e is either covered to an extent of at least $\frac{1}{2}$ in the first stage by the variables x_S, or it is covered to an extent of at least $\frac{1}{2}$ by the variables $r_{A,S}$ in *every scenario* A containing e. Let $E = \{e : \sum_S x_S \geq \frac{1}{2}\}$. Then $(2x)$ is a fractional set cover solution for the instance with universe E and so, using the ρ-approximation algorithm, one can obtain an integer set cover $\tilde{x}$ for this instance of cost at most $\rho \cdot \sum_S 2w_S^{\mathrm{I}} x_S$. Similarly for any scenario A, $(2r_A)$ is a fractional set cover for the elements in $A \setminus E$, since for each such element e we have $\sum_{S:e\in S} r_{A,S} \geq \frac{1}{2}$. Therefore one can cover these elements by a set cover of cost at most $\rho \cdot \sum_S 2w_S^A r_{A,S}$. So if we output $\tilde{x}$ as the first stage decisions, we incur a net cost of at most $2\rho \cdot h(x)$. ■

The proof shows that one can use the fractional solution to "decouple" the two stages (and indeed each of the scenarios for the second stage), and apply the deterministic result to each separately. Thus, the fact that we lose a factor of 2 (off of the deterministic guarantee) is exactly tied into the fact that we are considering a 2-stage problem. A similar decomposition can be applied to a number of other discrete stochastic optimization problems, as we show in Section 4.3.2 and in Section 5.6.

Furthermore, if we consider the case in which there are only a polynomial number of scenarios, then this rounding approach yields strong performance guarantees for a wide range of applications.

4.3.2 Proof of Theorem 4.3.1

For notational simplicity, we shall assume that the demands d_j^A in any scenario A are either 0 or 1, that is, a scenario A is now just a subset of the clients $\mathcal{D}$ that need to be assigned to facilities, and one can write a more compact formulation that only has variables $x_{A,ij}$ and constraints corresponding to clients j in A. The analysis extends in a straightforward way to the setting where we have arbitrary demands.

We first show that the integrality gap of (SUFL-P) is at most $2\rho_{\mathrm{UFL}}$. The proof is similar to the proof of Theorem 4.3.2. Let y be an optimal solution to (SUFL-P) and (x_A, y_A) be the optimal solution for scenario A given the first-stage decision vector y. Let OPT be the optimal solution value. We will show that we can decouple the first-stage and second-stage decisions, so that one can get an integer solution by separately solving a UFL problem for stage I and a UFL problem for each stage II scenario. Fix a scenario A and a client $j \in A$. We write $x_{A,ij} = x^{\mathrm{I}}_{A,ij} + x^{\mathrm{II}}_{A,ij}$ where $x^{\mathrm{I}}_{A,ij} \leq y_i$ and $x^{\mathrm{II}}_{A,ij} \leq y_{A,i}$. Since $x_{A,ij} \leq y_i + y_{A,i}$ we can always split $x_{A,ij}$ in the above way. Observe that j must be assigned to an extent of at least $\frac{1}{2}$ either by the assignment $\{x^{\mathrm{I}}_{A,ij}\}$ or by the assignment $\{x^{\mathrm{II}}_{A,ij}\}$, that is either $\sum_i x^{\mathrm{I}}_{A,ij} \geq \frac{1}{2}$ or $\sum_i x^{\mathrm{II}}_{A,ij} \geq \frac{1}{2}$. In the former case, we will assign j to a facility opened in stage I, and in the latter case we will assign j to a facility opened in stage II.

More precisely, for any client j, consider the set of scenarios $\mathcal{S}_j = \{A \subseteq \mathcal{D} : \sum_i x^{\mathrm{I}}_{A,ij} \geq \frac{1}{2}\}$. For our stage I decisions, we shall construct a feasible fractional solution for a UFL instance in which the facility costs are f_i^{I}, the assignment costs are c_{ij}, and each client j has a demand equal to $\sum_{A \in \mathcal{S}_j} p_A$; we then round this fractional solution to an integer solution using known algorithms for UFL.

In fact, we first construct a feasible solution in which there is a client (j, A) for each scenario $A \in \mathcal{S}_j$, with demand p_A, and then coalesce these scenario-dependent clients into one. Consider (j, A) such that $A \in \mathcal{S}_j$. We can obtain a feasible solution by setting $\hat{x}_{A,ij} = \min\bigl(1, 2x^{\mathrm{I}}_{A,ij}\bigr)$ and $\hat{y}_i = \min\bigl(1, 2y_i\bigr)$ for each $i \in \mathcal{F}$. (Note that a client may be assigned to an extent greater than 1.) However, the fractional facility variables do not depend on the scenario and given the fractional facility variables, we can re-optimize the fractional assignment for each client j: sort the facilities i in non-decreasing order of the assignment cost c_{ij}, and reset $\hat{x}_{A,ij}$ to $\hat{y}_i$ until the total assignment made is equal to 1 (where for the last facility i' to which j is assigned, we set $\hat{x}_{A,i'j}$ to the value needed to make the total assignment exactly 1). But this new fractional assignment is completely determined by the $\hat{y}_i$ values and *does not depend on* A, and so we can now view all of these clients (j, A) as one client j with *demand* $\sum_{A \in \mathcal{S}_j} p_A$. The facility cost of this fractional solution is $2\sum_i f^{\mathrm{I}}_i y_i$, and the assignment cost is no more than the one for the scenario-dependent clients, $2\sum_{i,j}\sum_{A \in \mathcal{S}_j} p_A c_{ij} x^{\mathrm{I}}_{A,ij} \le 2\sum_{i,j}\sum_{A \in \mathcal{S}_j} p_A c_{ij} x_{A,ij}$. Using the fact that the integrality gap of UFL is ρ_{UFL}, given this UFL instance with a fractional solution $(\hat{x}, \hat{y})$, we can now obtain an integer solution $(\tilde{x}, \tilde{y})$ of cost at most $2\rho_{\mathrm{UFL}}\bigl(\sum_i f^{\mathrm{I}}_i y_i + \sum_{i,j}\sum_{A \in \mathcal{S}_j} p_A c_{ij} x_{A,ij}\bigr)$; this determines the set of facilities to open in stage I, and for each client j takes care of the scenarios in $\mathcal{S}_j$.

In any scenario A, each client $j \in A$ such that $A \in \mathcal{S}_j$ is assigned to the stage I facility given by the assignment $\tilde{x}$. To assign the remaining clients, we solve a UFL instance with client set $\{j \in A : A \notin \mathcal{S}_j\}$. Since $A \notin \mathcal{S}_j$, we have that $\sum_i x^{\mathrm{II}}_{A,ij} \ge \frac{1}{2}$, and hence if we reset $\hat{x}_{A,ij} = \min\bigl(1, 2x^{\mathrm{II}}_{A,ij}\bigr)$, $\hat{y}_{A,i} = \min\bigl(1, 2y_{A,i}\bigr)$ for each $i \in \mathcal{F}$, we get a feasible solution for this set of clients. Again, we can get an integer solution of cost at most $2\rho_{\mathrm{UFL}}\bigl(\sum_i f^A_i y_{A,i} + \sum_{i, j \in A: A \notin \mathcal{S}_j} c_{ij} x_{A,ij}\bigr)$. This solution tells us which facilities to open in scenario A and how to assign the clients j in A with $A \notin \mathcal{S}_j$. Hence, the overall cost of the solution with first-stage facilities $\tilde{y}$ is at most $2\rho_{\mathrm{UFL}} \cdot OPT$, which implies that the integrality gap is at most $2\rho_{\mathrm{UFL}}$.

To obtain the approximation algorithm, we first obtain a near-optimal solution y in polynomial time. The difficulty in converting the proof of the integrality gap into a rounding algorithm is that the algorithm that shows that $\rho_{\mathrm{UFL}} \le 1.52$ due to [56] requires knowledge of the client demands, whereas we do not know the demand $\sum_{A \in \mathcal{S}_j} p_A$ of a client j, and might not be able to even estimate it by sampling, since the probability p_A could be extremely small. We therefore need an approximation algorithm for UFL that works without explicit knowledge of the client demands. The primal-rounding algorithm described in Section 2.4 has this property; the algorithm converts any fractional solution to an integer solution increasing the cost by a factor of at most 1.858 (with $\gamma = 1/1.858$).

We use this algorithm to obtain the approximation algorithm. We modify the definition of $\mathcal{S}_j$ slightly, so as to balance the contribution from stages I and II. Let $\theta = \frac{1.858}{1.858+1.52}$. Let $\mathcal{S}_j = \{A \subseteq \mathcal{D} : \sum_i x^{\mathrm{I}}_{A,ij} \geq \theta\}$. So now we have a fractional solution in which we set $\hat{y}_i = \min(1, y_i/\theta)$, and using the re-optimization procedure described earlier, we can find the optimal fractional assignment $\hat{x}$ corresponding to the $\hat{y}_i$ values. We round this using the algorithm of Section 2.4 to get a solution $(\tilde{x}, \tilde{y})$ of cost at most $\frac{1.858}{\theta} \cdot \big(\sum_i f^{\mathrm{I}}_i y_i + \sum_{i,j} \sum_{A \in \mathcal{S}_j} p_A c_{ij} x_{A,ij}\big)$. This determines the facilities to open in stage I. In any scenario A, each client $j \in A$ such that $A \in \mathcal{S}_j$ is taken care of by a stage I facility. Next, we determine which facilities to open in scenario A and how to assign the remaining clients in A by constructing a feasible fractional solution for a deterministic subproblem with client set $\{j \in A : A \notin \mathcal{S}_j\}$ and "rounding" this solution. We set $\hat{y}_{A,i} = \min\big(1, y_{A,i}/(1-\theta)\big)$ and for each client $j \in A$ such that $A \notin \mathcal{S}_j$, set $\hat{x}_{A,ij} = \min\big(1, x^{\mathrm{II}}_{A,ij}/(1-\theta)\big)$. We "round" this solution using the algorithm of Mahdian et al. [56] (which is not an LP rounding algorithm) since the issue of the demands does not apply to this stage, to get an integer solution of cost at most $\frac{1.52}{1-\theta} \cdot \big(\sum_i f^A_i y_{A,i} + \sum_{i,j \in A : A \notin \mathcal{S}_j} c_{ij} x_{A,ij}\big)$. So the total cost incurred if we open the facilities given by $\tilde{y}$ in stage I is at most $3.378 \cdot OPT$.

Remark 4.3.3 The only change with arbitrary demands d^A_j, and/or scenario-dependent assignment costs $c^A_{ij} = \gamma^A c_{ij}$, is that in the feasible fractional solution we exhibit to bound the cost of the stage I decisions, each client j now has demand equal to $\sum_{A \in \mathcal{S}_j} p_A d^A_j \gamma^A$, and in the fractional solution constructed for a stage II scenario A, each client $j \in A$ such that $A \notin \mathcal{S}_j$ has demand $d^A_j \gamma^A$.

Remark 4.3.4 It is possible to prove an integrality gap of at most 3 by adapting the primal-dual algorithm of Jain & Vazirani (Section 2.2). But this requires explicit knowledge of the probability of *every* scenario p_A, and it seems difficult to convert the proof to obtain a polynomial-time algorithm.

4.4 An Extension

We consider an extension of SUFL where in any scenario, there is an option of not satisfying the demand of a client by incurring a certain penalty (which may be scenario-dependent). The deterministic version of this problem where the demands are known in advance was introduced by Charikar, Khuller, Mount & Narsimhan [17] under the name of facility location with penalties. A rounding approach similar to the above procedure yields a $(4.378 + \epsilon)$-approximation algorithm for this problem.

Let $\ell_j^A \geq 0$ denote the penalty incurred for not assigning client j in a scenario A. It is reasonable to assume that if client j has zero demand to be assigned in a scenario, then one does not incur any penalty for not assigning (the zero demand of) the client, i.e., if $d_j^A = 0$ then $\ell_j^A = 0$. The relaxation of this problem has extra variables $v_{A,j}$ that indicate whether we incur the penalty for client j in scenario A. The minimization problem for scenario A is now given by

$$
\begin{aligned}
g_A(y) \; = \; \min \quad & \sum_i f_i^A y_{A,i} + \sum_j d_j^A \sum_i c_{ij} x_{A,ij} + \sum_j \ell_j^A v_{A,j} & & \\
\text{s.t.} \quad & \sum_i x_{A,ij} + v_{A,j} \geq 1 & \text{for all } j, & \quad (1) \\
& x_{A,ij} \leq y_i + y_{A,i} & \text{for all } i, j, & \\
& x_{A,ij}, v_{A,j}, y_{A,i} \geq 0 & \text{for all } i, j. &
\end{aligned}
$$

Constraint (1) says that we either assign a client to a facility, or we incur the penalty for that client. The resulting stochastic program can be solved to obtain a $(1 + \epsilon)$-optimal solution using the algorithm of Section 3. For simplicity we again focus on the case where the demands d_j^A are 0-1 values.

We first show that the integrality gap of the formulation is at most $2\rho_{\text{UFL}}+1$. Let y be an optimal solution. The rounding procedure is very similar to the earlier rounding procedure. Let (x_A, y_A, v_A) denote the optimal solution for scenario A given the first stage vector y. We again write $x_{A,ij} = x^{\text{I}}_{A,ij} + x^{\text{II}}_{A,ij}$ where $x^{\text{I}}_{A,ij} \le y_i$ and $x^{\text{II}}_{A,ij} \le y_{A,i}$. Let $\theta = \frac{\rho_{\text{UFL}}}{2\rho_{\text{UFL}}+1}$. Note that either $\sum_i x^{\text{I}}_{A,ij} \ge \theta$ or $\sum_i x^{\text{II}}_{A,ij} \ge \theta$ or $v_{A,j} \ge \frac{1}{2\rho_{\text{UFL}}+1}$. Define $\mathcal{S}_j = \{A \subseteq \mathcal{D} : \sum_i x^{\text{I}}_{A,ij} \ge \theta\}$. To decide which facilities to open in stage I, we consider a UFL instance where client j has demand $\sum_{A\in\mathcal{S}_j} p_A$ and a feasible fractional solution $(\hat{x}, \hat{y})$ for this instance, where $\hat{y}_i = \min(1, y_i/\theta)$, and $\hat{x}_{ij}$ is set so as to re-optimize the fractional assignment for client j given this setting of the $\hat{y}_i$ variables. We round this solution to get an integer solution $(\tilde{x}, \tilde{y})$ of cost at most $\frac{\rho_{\text{UFL}}}{\theta} \cdot \big(\sum_i f^{\text{I}}_i y_i + \sum_{i,j}\sum_{A\in\mathcal{S}_j} p_A c_{ij} x_{A,ij}\big)$; this determines the set of facilities to open in stage I. In any scenario A, each client $j \in A$ such that $A \in \mathcal{S}_j$ is assigned to the stage I facility given by the assignment $\tilde{x}$. For each remaining client $j \in A$, we have either $\sum_i x^{\text{II}}_{A,ij} \ge \theta$ or $v_{A,j} \ge \frac{1}{2\rho_{\text{UFL}}+1}$. In the latter case, we simply incur the penalty for j. So the total penalty incurred is bounded by $(2\rho_{\text{UFL}}+1)\sum_{j\in A: A\notin\mathcal{S}_j} v_{A,j}\ell^A_j$. To assign the remaining clients, we solve a UFL instance with client set $D_A = \{j \in A : A \notin \mathcal{S}_j, v_{A,j} < \frac{1}{2\rho_{\text{UFL}}+1}\}$. If we set $\hat{x}_{A,ij} = \min(1, x^{\text{II}}_{A,ij}/\theta)$ and $\hat{y}_{A,i} = \min(1, y_{A,i}/\theta)$ for each $i \in \mathcal{F}$, we get a feasible fractional solution for this UFL instance. So we can get an integer solution of cost at most $\frac{\rho_{\text{UFL}}}{\theta} \cdot \big(\sum_i f^A_i y_{A,i} + \sum_{i,j\in A: A\notin\mathcal{S}_j} c_{ij} x_{A,ij}\big)$. This solution tells us which facilities to open in scenario A and how to assign the clients in D_A. The overall cost of the solution with first-stage facilities $\tilde{y}$ is at most $(2\rho_{\text{UFL}}+1)\cdot OPT$, showing that the integrality gap is at most $2\rho_{\text{UFL}}+1$.

To get an approximation algorithm, we obtain a $\big(1+\frac{\epsilon}{5}\big)$-optimal solution y, and then round the solution. In the rounding procedure, as we did earlier, we use the primal-rounding algorithm from Section 2.4 to determine the first stage facilities. We now set $\theta = \frac{1.858}{1.858+1.52+1}$. The set $\mathcal{S}_j$ consists of scenarios $\{A \subseteq \mathcal{D} : \sum_i x^{\text{I}}_{A,ij} \ge \theta\}$. The stage I facilities $\tilde{y}$ are determined by using the primal-rounding algorithm which rounds a fractional solution increasing the cost by a factor of at most 1.858. In a stage II scenario A, we incur the penalty for a client $j \in A$ such that $A \notin \mathcal{S}_j$, if

$v_{A,j} \geq \frac{1}{1.858+1.52+1}$. For the remaining clients in A we solve a UFL instance as above. Doing the routine calculations, we get that the cost of the solution with first-stage facilities $\tilde{y}$ is at most $(4.378+\epsilon) \cdot OPT$.

Theorem 4.4.1 *There is a* $(4.378+\epsilon)$*-approximation algorithm for SUFL with penalties.*

Chapter 5

Facility Location with Service Installation Costs

5.1 Introduction

Consider a classical application of the uncapacitated facility location problem (UFL), the warehouse or plant location problem, where facilities are warehouses and the clients are retail stores or customers that request different types of items. Typically in UFL, it is assumed that a client may be assigned to any facility; translated to the warehouse location problem, this means that any warehouse may service any customer. However, this is usually not true; a warehouse might only have supplies of specific items, and hence, to satisfy a customer we need to assign it to a warehouse that holds inventory of the item requested by the customer. To model such settings where clients request specific services and have to be assigned to facilities that can provide the requested services, we introduce the *facility location with service installation costs* (FLSIC) problem.

In addition to a set of facilities $\mathcal{F}$, and a set of clients or demands $\mathcal{D}$, we also have a set of services $\mathcal{S}$. Each client j in $\mathcal{D}$ requests a specific service $g(j) \in \mathcal{S}$. To satisfy client j we have to assign it to an open facility on which service $g(j)$ is installed. Further, if we install service l at an open facility i, we incur a *service installation cost* of f_i^l. This is in addition to the usual *facility cost* f_i that we incur to open facility i. We want to open a set of facilities, install services at the open facilities, and assign each client j to an open facility i such that service $g(j)$ is installed at i. The cost of a solution is the sum of the facility opening costs, the service installation costs and the client assignment costs, and the goal is to find a solution with minimum total cost. This problem is a generalization of UFL, since with just one service type the problem reduces to UFL. In the warehouse location problem, the service installation cost corresponds to the initial cost of setting up the warehouse to store the particular kind of inventory. The notion of service-dependent fixed costs is also used in inventory problems where there is a joint setup cost to start a new order and an item-dependent fixed cost to order a specific item, so that one needs to coordinate the placement of item orders; see [4] for a survey.

This problem can also be used to model a caching application. We are given a network of locations. The facilities correspond to caches that may be built at certain locations, and the clients are processes sitting at nodes of the network requesting data items. Each process requests a specific data item and must be assigned to a cache that stores the requested data item. A data item can therefore be viewed as a service. The cost of accessing the item is proportional to the distance between the process site and the cache location. There is a (location-dependent) cost associated with building a cache and a (location- and item-dependent) cost for storing a data item in a cache at a particular location. The goal is to decide where to locate caches and the set of data items to store in each cache, and assign each process to a cache containing its requested data item, so as to minimize the total cost. Observe that this is precisely the facility location problem with service installation costs.

5.1.1 Summary of Results

The main result of this chapter is a primal-dual 6-approximation algorithm for the problem under the assumption that the facilities can be sorted so that if i comes before i' in the ordering, then the cost of installing any service at i is no more than the cost of installing that service at i', i.e., for *every* service type l, $f_i^l \leq f_{i'}^l$. This is reasonable in many settings; for example, one might expect the inventory setup cost of a warehouse in New York city to be less than the inventory setup cost in a remote town like Ithaca, regardless of the kind of inventory. As special cases, this includes the cases where the service installation cost f_i^l depends only on the location i, or only on the service type l. For this latter special case, we give an LP rounding algorithm that attains a much improved approximation ratio of 2.391. The algorithm combines both clustered randomized rounding [18] and the filtering based technique of [54, 71]. It uses the bounds obtained by complementary slackness and filtering in conjunction to bound the assignment cost, and thus get a better performance guarantee than that obtained by using either of the two approaches separately. With arbitrary service installation costs, we show that the problem becomes as hard as the set-cover problem.

In Section 5.6 we consider a stochastic version of the problem that fits in the 2-stage recourse model discussed in Chapters 3 and 4, and devise an approximation algorithm for this problem when the service installation cost depends only on the service type (but could vary across the different scenarios).

Building upon these results, in Chapter 7 we consider the k-median version of the problem where we require that at most k facilities be opened. We use our primal-dual algorithm to give a constant-factor approximation for this problem when the service installation cost depends only on the service type. This algorithm is presented in Section 7.5.

5.1.2 Related Work

The work that is most closely related to our problem is a paper by Baev & Rajaraman [8] which looks at a variant of the caching application above, called the *data placement* problem. Here caches have a fixed capacity and are already built at certain locations. The goal is to find a placement of data items to caches that respects the cache capacities and minimizes the sum of the access costs and the cost of storing data items. Baev & Rajaraman gave a 20.5-approximation algorithm and the ratio has been recently improved to 10 [74]. Ravi & Sinha [63] consider a similar model under the name of *multicommodity facility location*, where they model the service installation cost at facility i by an arbitrary set function $s_i : \mathcal{S} \mapsto \mathbb{R}_+$, and require that the function value be explicitly given for each subset of services making the input exponentially large. Since this problem contains, as a special case, FLSIC with arbitrary service installation costs (where the input can be concisely specified), it is set-cover hard, and [63] complements this with a $O(\log |\mathcal{S}|)$-approximation algorithm. Also related is work on a class of inventory problems called *joint replenishment problems* (see [4] for a survey). In the basic setting, there is a time line and demands for items specified at various points of time, and one has to determine the times at which to place orders and decide which items to order at these times, so that all demand can be met by orders that are placed at earlier points of time. Placing an order for a subset of items incurs both an item-dependent fixed cost, and a joint ordering cost to start a new order that is independent of the items ordered, and there is a cost incurred to hold inventory for demands that may occur later. This is an instance of FLSIC where the candidate facility locations are points on the time line and the services correspond to items. The holding cost can be charged against the demand for which the inventory is held, and translates into a client assignment cost. This problem deals with an asymmetric metric (the directed line metric), however it is also more specialized than FLSIC in that it considers one specific metric, and this additional structure was exploited in [53] to give a 2-approximation algorithm for this problem recently.

5.2 Hardness with Arbitrary Service Installation Costs

We show that FLSIC with arbitrary service installation costs is at least as hard as the set cover problem. Intuitively, the reason is that, by making the service installation cost $f_i^l = \infty$ (i.e., a suitably high value), we can encode the constraint that a client j requesting service l cannot be assigned to i.

In the set cover problem we have a ground set of elements $U = \{e_1, e_2, \ldots e_n\}$, and a collection of subsets of U, $S_1, \ldots S_m$, and we want to choose as few sets as possible so that every element e_j is included in some chosen set (a minimum set cover). Given such an instance, we create the following instance of our problem: the sets correspond to facilities and the elements becomes the clients. All the facilities and clients are co-located at the same point, that is, $c_{ij} = 0$ for all i, j (or very small, say ϵ). Each client e_j requests a *distinct service* j. The facility opening costs are all set to 1, and the service installation cost f_i^l is 0 if element $e_l \in S_i$ and ∞ otherwise. Now we have the following correspondence: a set cover of size k yields an FLSIC solution of cost k and vice versa, hence the problem is set-cover hard. Combined with the result of Raz & Safra [64], this shows there is some absolute constant $c < 1$, such that for any $\epsilon > 0$, no polynomial time algorithm with a ratio of $(c - \epsilon) \ln |\mathcal{D}|$ exists for this problem in the general case unless *P=NP*, while the result of Feige [25] shows that there is no polynomial-time algorithm for this problem with a ratio of $(1 - \epsilon) \ln |\mathcal{D}|$ unless $NP \subseteq DTIME[n^{O(\log \log n)}]$.

5.3 A Linear Program

We formulate the problem as an integer program and relax the integrality constraints to get a linear program. We use i to index the facilities in $\mathcal{F}$, j to index the clients in $\mathcal{D}$ and l to index the services in $\mathcal{S}$. As done previously, we will assume for simplicity that each client j has unit demand. The analysis extends in a straightforward way to the case where clients have arbitrary demands, and all of the results carry over.

$$\begin{aligned} \min \quad & \sum_i f_i y_i + \sum_i \sum_l f_i^l y_i^l + \sum_j \sum_i c_{ij} x_{ij} && \text{(FLS-P)} \\ \text{s.t.} \quad & \sum_i x_{ij} \geq 1 && \text{for all } j, \\ & x_{ij} \leq y_i^{g(j)} && \text{for all } i, j, \\ & x_{ij} \leq y_i && \text{for all } i, j, \\ & x_{ij}, y_i, y_i^l \geq 0 && \text{for all } i, j, l. \end{aligned}$$

Variable y_i indicates if facility i is open, y_i^l indicates if service type l is installed at i, and x_{ij} indicates if client j is connected to facility i. The first constraint states that each client must be assigned to a facility, the second and the third constraints say that if client j is assigned to facility i, then service $g(j)$ must be installed on i and i must be open. An integral solution corresponds exactly to a solution to our problem. Let G_l be the set of clients requesting service l. The dual program is

$$
\begin{aligned}
\max \quad & \sum_j \alpha_j && && \text{(FLS-D)} \\
\text{s.t.} \quad & \alpha_j \leq c_{ij} + \beta_{ij} + \theta_{ij} && \text{for all } i, j, && (1) \\
& \sum_{j \in G_l} \theta_{ij} \leq f_i^l && \text{for all } i, l, && \\
& \sum_j \beta_{ij} \leq f_i && \text{for all } i, && (2) \\
& \alpha_j, \beta_{ij}, \theta_{ij} \geq 0 && \text{for all } i, j. &&
\end{aligned}
$$

We can interpret α_j as the *budget* that j is willing to spend to get itself assigned to an open facility. Constraint (1) says that a part of this goes towards paying for the assignment cost c_{ij}. The rest gets divided into a payment for the service installation cost θ_{ij}, and a payment for the facility opening cost β_{ij}.

5.4 The Primal-Dual Algorithm

We consider instances of the problem where there is an ordering on the facilities in $\mathcal{F}$ such that if i comes before i' in this ordering then for *every* service type l, $f_i^l \leq f_{i'}^l$. This is equivalent to saying that for any two locations i, i', the service installation cost vectors $\left(f_i^l\right)^{\mathrm{T}}_{l=1\ldots|\mathcal{S}|}$ and $\left(f_{i'}^l\right)^{\mathrm{T}}_{l=1\ldots|\mathcal{S}|}$, are comparable (under the usual $\leq$ relation on vectors). Let $\mathcal{O}$ denote this total ordering on the facilities. We say that $i \prec i'$ if i comes before i' in the ordering $\mathcal{O}$.

The algorithm is strongly motivated by the primal-dual algorithm of Jain and Vazirani (JV) for UFL (see Section 2.2). If there were no facility opening costs we could decouple the problem into several UFL instances, one for each service type, and run the JV algorithm on each instance separately. With facility opening costs, this approach fares badly since we may end up opening a lot of facilities and incur a huge facility opening cost. The JV algorithm relies on the fact (as do other algorithms for UFL) that a client j can be moved from a facility i to another nearby facility i' without increasing its assignment cost by much, and leaving the facility opening cost unchanged. However in our case, reassigning j to i' may now require us to install service $g(j)$ on i' causing us to pay the installation cost $f_{i'}^{g(j)}$ which could be large. The hard part is to find a way to reassign clients to nearby facilities while ensuring that we do not pay too much to install services at the new locations. With arbitrary service installation costs such a reassignment need not be possible since we can encode the constraint that a client may only be assigned to a specific set of facilities, making the problem set-cover hard.

5.4.1 The Dual Ascent Procedure

As in the JV algorithm, there is a notion of time around which the algorithm is specified. We start at time $t = 0$. All dual variables are initialized to 0, each demand j is said to be *unfrozen*, and all facilities are closed. As time increases, we *tentatively install services* at facilities, *tentatively open* facilities, and *freeze* demand points. For every demand j, we increase the dual variable α_j at unit rate, and for any facility i, we first increase the θ_{ij} variable and then the β_{ij} variable. We say that demand j is *tight* with facility i, or has *reached* i, if $\alpha_j \geq c_{ij}$. We continue to increase α_j until j becomes tight with a tentatively open facility on which the service it requests is tentatively installed, at which point we freeze demand point j. The primal-dual process ends when all clients are frozen. At this point, a demand point may be paying for opening multiple facilities and installing services at multiple facilities, so we have a cleanup step, to decide which tentatively open facilities to finally open and and what services to install at each open facility. The cleanup phase is somewhat involved since we have to simultaneously ensure that (1) we can pay for opening facilities and installing services and, (2) if a client j has to be reassigned, there is a nearby open facility on which service $g(j)$ is installed. We show that we can achieve properties (1) and (2) if we consider the tentatively open facilities in a particular order and greedily pick a maximal subset that satisfies certain properties (analogous to an independent set in the JV algorithm). This gives us a 6-approximation algorithm. A key property that we exploit is the fact that in the JV algorithm, one can choose *any maximal independent set of tentatively open facilities* for opening.

We next describe the algorithm in detail. At time $t = 0$ we start increasing the α_j variables at unit rate, so for any unfrozen demand j, α_j is always equal to the time t. We increase the α_j of each demand j until one of the following events happens (if several events happen simultaneously, consider them in any order):

1. Suppose that demand j becomes tight with facility i. If service $g(j)$ is not tentatively installed at i, then we start increasing θ_{ij} at the same rate as α_j. If service $g(j)$ is tentatively installed, but i is not tentatively open, we instead increase β_{ij} at the same rate as α_j, i.e., if $\alpha_j = t$, then θ_{ij} remains 0, but $\beta_{ij} = t - c_{ij}$. Otherwise, that is, if service $g(j)$ is tentatively installed and i is tentatively open, we freeze client j (and no longer increase α_j).

2. Suppose that for a facility i and a service type l, we have $\sum_{j \in G_l} \theta_{ij} = f_i^l$. In this case, we tentatively install service l at i. If i is also tentatively open, then we freeze each demand $j \in G_l$ that is tight with i. If i is not yet tentatively open, then for each demand $j \in G_l$ that is tight with i, we no longer increase θ_{ij}, but instead start increasing β_{ij} at the same rate as α_j.

3. Suppose that for a facility i, $\sum_j \beta_{ij} = f_i$: in this case, we tentatively open i. For each demand j, we do not increase β_{ij} from now on. If demand j is tight with i and service $g(j)$ is tentatively installed at i, we freeze j.

We only raise the $\alpha_j, \beta_{ij}, \theta_{ij}$ of unfrozen demands. Frozen demands do not participate in any events. We continue this process until all demands become frozen. Let (α, β, θ) denote the final dual solution obtained by the above process. Observe that if

i is the facility that caused j to freeze, then service $g(j)$ must be tentatively installed at i and i must be tentatively open.

5.4.2 Opening Facilities, Installing Services, and Assigning Demands

We now specify which facilities to open, how to install services on facilities, and how to assign demands to facilities. Our goal is to ensure that a client pays for (a) opening at most one open facility, and (b) installing service on at most one open facility. Correspondingly we will have two cleanup phases. The first phase will open facilities so as to ensure property (a). A guiding principle used to choose between two conflicting facilities i and i' (conflicting in the sense that opening both would violate (a) or (b)), is that if we prefer i' to i, then we should be able to relocate, if necessary, every service installed at i to i' *without paying any extra service installation cost*, that is, we want every service l installed at i to satisfy $f^l_{i'} \leq f^l_i$. The ordering $\mathcal{O}$ on facilities comes in handy here. We consider facilities in the order given by $\mathcal{O}$ and greedily choose a maximal non-conflicting set. This guarantees that if facility i is not chosen then there is some conflicting facility i' that was picked before it (so it must be that $i' \prec i$); in particular this implies that $f^l_{i'} \leq f^l_i$ for every service l.

In the second phase, for each service l we pick a non-conflicting set of facilities to install service l. However it could be that a facility i we pick was not chosen in the Phase 1, and hence is not open. But in this case we know that there must be an open facility i' due to which i was not opened in Phase 1, and we can install service l on this facility i' instead. Our rule for choosing facilities in Phase 1 ensures that the cost of installing service l at i' is bounded by the installation cost at i — this is exactly why we wanted Phase 1 to guarantee this property. The final algorithm is a bit more involved. The two cleanup steps are interlinked because we want to guarantee that a demand j does not pay for both opening a facility i, and installing service $g(j)$ at *some other* facility i'. This will allow us to give a stronger performance guarantee that is crucially used in the k-median variant of the problem discussed in Section 7.5.

We define four kinds of dependence between facilities. Say that the *ordered pair* (i, i') is,

(1) ff-dependent (f for facility) if there is a demand j such that $\beta_{ij}, \beta_{i'j} > 0$.

(2) sf-l dependent (s for service) if there exists $j \in G_l$ such that $\theta_{ij}, \beta_{i'j} > 0$.

(3) ss-l dependent if there exists $j \in G_l$ such that $\theta_{ij}, \theta_{i'j} > 0$.

(4) fs-l dependent if there exists $j \in G_l$ such that $\beta_{ij}, \theta_{i'j} > 0$.

Let F be the set of tentatively open facilities, and let $F_l \subseteq F$ be the set of tentatively open facilities on which service l is tentatively installed. For facility $i \in F$, let t_i be the time at which i became tentatively open. If $i \in F_l$, let t_{il} be the time at which service l was tentatively installed at i. Initially for each facility $i \in F$, let S_i be the set of services that are tentatively installed at i.

P1. We first pick a set $F' \subseteq F$ of facilities to open, and for each $i \in F'$ a set $T_i \subseteq S_i$ of services to install at facility i. Initialize $F' = \emptyset$ and $T_i = \emptyset$ for all i. We consider facilities in F in the order given by $\mathcal{O}$. While $F \neq \emptyset$,

1. Let $i \in F$ be the currently considered facility. Remove i from F, set $F' \leftarrow F' \cup \{i\}$, $T_i = S_i$.

2. For each $i' \in F$ we do the following.

   a) If (i, i') is ff-dependent OR $\exists l \in T_i$ such that (i, i') is sf-l dependent OR $\exists l \in S_{i'}$ such that (i, i') is fs-l dependent and $t_{i'l} < t_{i'}$, set $F \leftarrow F - \{i'\}$. Call i the neighbor of i' and denote it by $\mathsf{nbr}(i')$. Otherwise,

   b) For every $l \in S_{i'}$, if (i, i') is fs-l dependent (so $t_{i'l} \geq t_{i'}$) OR $l \in T_i$ and (i, i') is ss-l dependent, set $S_{i'} \leftarrow S_{i'} - \{l\}$.

We open the facilities in F'. For each $i \in F'$ install all the services in T_i at i.

P2. We now install services at some more facilities. Consider service type l. Let A_l be the facilities in F' at which service l is installed (i.e., i such that $l \in T_i$). Note that $A_l \subseteq F' \cap F_l$. Let $B_l = F_l - F'$. We remove from B_l every facility

i' for which there is some facility $i \in F'$ such that (1) (i, i') is fs-l dependent, OR (2) $i \in A_l$ and (i, i') is ss-l dependent. We say that a set of facilities is ss-l independent if no pair (i, i') of facilities from the set is ss-l dependent. We pick a maximal ss-l independent subset $F'_l \subseteq B_l$. Initially set $F'_l = \emptyset$. We consider facilities in B_l in increasing order of t_i and add facility i to F'_l if $F'_l \cup \{i\}$ remains ss-l independent. For every $i \in F'_l$ we install service l on the nearest facility $i' \in F'$ such that $i' \prec i$.

P3. Each client j is assigned to the nearest open facility at which service $g(j)$ is installed.

5.4.3 Analysis

We now bound the performance of our algorithm. The following lemma just says what it means for a demand j to get frozen.

Lemma 5.4.1 *Let i be the facility that causes a demand j to freeze. Then, i is tentatively open, service $g(j)$ is tentatively installed at i, and $\alpha_j = \max(c_{ij}, t_i, t_{ig(j)})$.*

We start by bounding the cost incurred in opening facilities and installing services. Let D' be the subset of demands $\{j : \exists i \in F' \text{ s.t. } \beta_{ij} > 0\}$. By the construction of F', we know that for each demand j there is at most one $i \in F'$ such that $\beta_{ij} > 0$; for $j \in D'$ let $o(j)$ denote this unique facility in F'.

Lemma 5.4.2 *The cost of opening facilities is $\sum_{j \in D'} \beta_{o(j)j}$. The cost of installing services is at most $\sum_{j \in D'} \theta_{o(j)j} + \sum_{j \notin D'} \alpha_j$.*

Proof : For each facility i that is tentatively opened, we have $f_i = \sum_{j \in \mathcal{D}} \beta_{ij}$. If i is in F' then β_{ij} is positive only for $j \in D'$ with $o(j) = i$. So $\sum_{i \in F'} f_i = \sum_{i \in F'} \sum_{j \in D' : o(j) = i} \beta_{ij} = \sum_{j \in D'} \beta_{o(j)j}$.

A service l is tentatively installed at facility i only when $f_i^l = \sum_{j \in G_l} \theta_{ij}$. Consider service l. Consider a demand j in G_l. We claim that there is at most one facility $i \in A_l \cup F'_l$ for which $\theta_{ij} > 0$ and further that if $j \in D'$ then $i = o(j)$ may be the only such facility. Since F'_l is an ss-l independent subset, θ_{ij} is positive for at most one facility in F'_l. If $\theta_{ij} > 0$ for some facility $i \in F'_l$, then it must be that $\theta_{i'j} = 0$ for every $i' \in A_l$ and $\beta_{i''j} = 0$ for every $i'' \in F'$, otherwise i would be not be included in B_l and we have $F'_l \subseteq B_l$. In particular, this shows that if $j \in D'$ then $\theta_{ij} = 0$ for all $i \in F'_l$. Now suppose $\theta_{ij} > 0$ for some $i \in A_l$. Then we must have $\theta_{i'j} = 0$ for all other $i' \in A_l$, otherwise we get a contradiction. Since service l is installed on both i and i', $l \in S_i \cap S_{i'}$ throughout step P1. If i was considered before i', since (i, i') is ss-l dependent we would have removed l from $S_{i'}$; if i' is considered earlier then we would not have $l \in S_i$, obtaining a contradiction. A similar reasoning shows that if $\theta_{ij} > 0$ for $i \in A_l$, then $\beta_{i'j} = 0$ for every facility $i' \neq i$ in F'.

For each $i \in A_l \cup F'_l$, we install service l either at i or at a facility i' such that $f^l_{i'} \leq f^l_i$. So the cost of installing service l is at most $\sum_{i \in A_l \cup F'_l} f^l_i = \sum_{j \in G_l} \sum_{i \in A_l \cup F'_l} \theta_{ij}$ which by the above claim is upper bounded by $\sum_{j \in G_l \cap D'} \theta_{o(j)j} + \sum_{j \in G_l \setminus D'} \alpha_j$. Summing over all services l, gives the lemma. ■

We next bound the assignment cost incurred by the solution computed. The following facts, which follow directly from the construction of the algorithm, will be useful in this analysis.

Fact 5.4.3 *Suppose that β_{ik} is positive. Then $c_{ik} \leq \alpha_k - \beta_{ik}$ and $\alpha_k \leq t_i$.*

Fact 5.4.4 *Suppose that θ_{ik} is positive. Then $c_{ik} \leq \alpha_k - \theta_{ik}$ and $c_{ik} < t_{ig(k)}$. If $\beta_{ik} = 0$ then $\alpha_k \leq t_{ig(k)}$.*

We use these to derive the following bounds.

Claim 5.4.5 *Let $i, i' \in F_l$ be such that (i, i') (and hence (i', i)) is ss-l dependent due to demand $k \in G_l$. Then $c_{ii'} < 2\max(t_{il}, t_{i'l})$ and both c_{ik} and $c_{i'k}$ are strictly less then α_k.*

Proof : From the dependence of i and i', it follows that θ_{ik} and $\theta_{i'k}$ are positive. Applying Fact 5.4.4 for both of these, and using the triangle inequality, we get that $c_{ii'} < 2\max(t_{il}, t_{i'l})$, $c_{ik} < \alpha_k$ and $c_{i'k} < \alpha_k$. ■

Claim 5.4.6 *Let $i \notin F'$ and $i' = \mathsf{nbr}(i) \in F'$. Then $c_{i'i} < 2t_i$.*

Proof : By the definition of $\mathsf{nbr}(.)$ in step 2a), we have that (i', i) is either (1) ff-dependent, OR (2) sf-l dependent for some $l \in S_{i'}$, OR (3) fs-l dependent for some $l \in S_i$ with $t_{il} < t_i$. In either case there is some demand j such that $\max(\beta_{i'j}, \theta_{i'j}) > 0$ and either $\beta_{ij} > 0$ in cases (1) and (2) OR, if cases (1) and (2) do not apply and case (3) applies, $\beta_{ij} = 0,\ \theta_{ij} > 0,\ t_{ig(j)} < t_i$ and $\beta_{i'j} > 0$. Using Facts 5.4.3 and 5.4.4 we get that $c_{ij} < \alpha_j \leq t_i$ and $c_{i'j} < \alpha_j$, implying that $c_{i'i} < 2t_i$. ∎

Claim 5.4.7 *Suppose $i \in (F' \cap F_l) \setminus A_l$ with $t_{il} < t_i$. Then there is a facility $i' \in A_l$ such that $c_{i'i} < 2t_{il}$.*

Proof : Since $l \in S_i$ at the beginning of step P1, there is some $i' \in F'$ due to which service l was removed from S_i in step 2b). Either (i', i) is fs-l dependent and $t_{il} \geq t_i$ OR $l \in T_{i'}$ and (i', i) is ss-l dependent. The former case cannot happen since $t_{il} < t_i$. In the latter case i' lies in A_l. Let $k \in G_l$ be a demand such that $\theta_{ik}, \theta_{i'k}$ are positive. It must be that $\beta_{ik} = 0$ otherwise (i', i) would be sf-l dependent and we would have removed i from F in step 2a). So by Fact 5.4.4, $\alpha_k \leq t_{il}$ and by Claim 5.4.5 $c_{ik}, c_{i'k} < \alpha_k$ showing that $c_{i'i} < 2t_{il}$. ∎

We are now ready to bound the assignment cost incurred. We will show that for any demand j there always exists some open facility with service $g(j)$ installed that is no further from j than the claimed bound, implying that the closest one, to which j is assigned, is no further away.

Lemma 5.4.8 *If $j \in D'$, the assignment cost of j is at most $3(\alpha_j - \beta_{o(j)j})$.*

Proof : Consider $j \in D'$ with $g(j) = l$. Let $i = o(j) \in F'$. By Fact 5.4.3, $c_{ij} \leq \alpha_j - \beta_{ij}$. If $i \in A_l$, then service l is installed at i. Otherwise i lies in $(F' \cap F_l) \setminus A_l$ and since $\beta_{ij} > 0$, $t_{il} \leq \alpha_j - \beta_{ij} < t_i$ (by Fact 5.4.3). Applying Claim 5.4.7, $\exists i' \in A_l$ such that $c_{i'i} < 2t_{il}$. So service l is installed at i' and $c_{i'j} < 3(\alpha_j - \beta_{ij})$. ∎

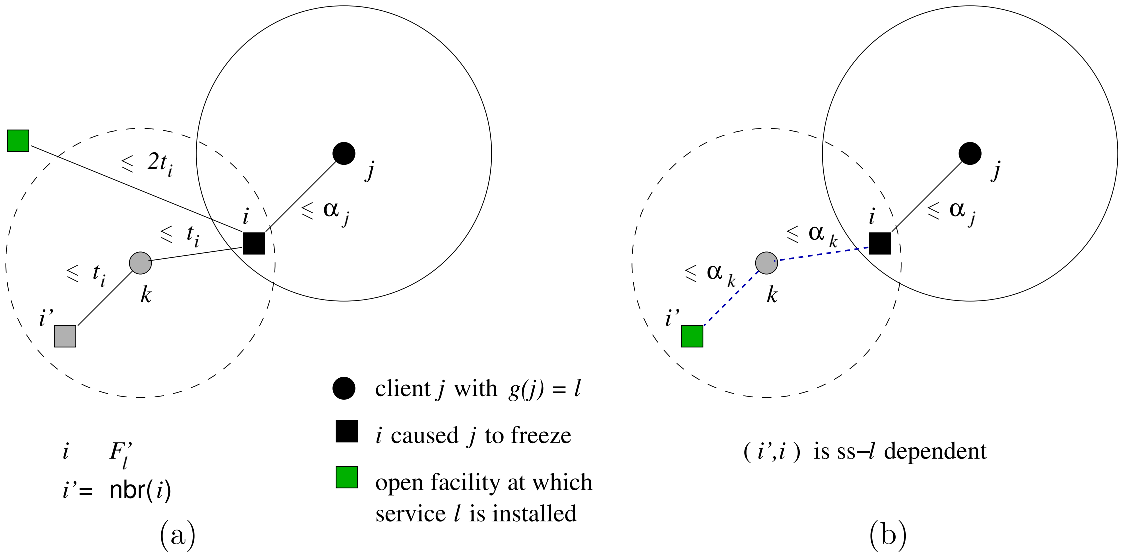

Figure 5.1: The 3-hop cases encountered in Lemma 5.4.9.

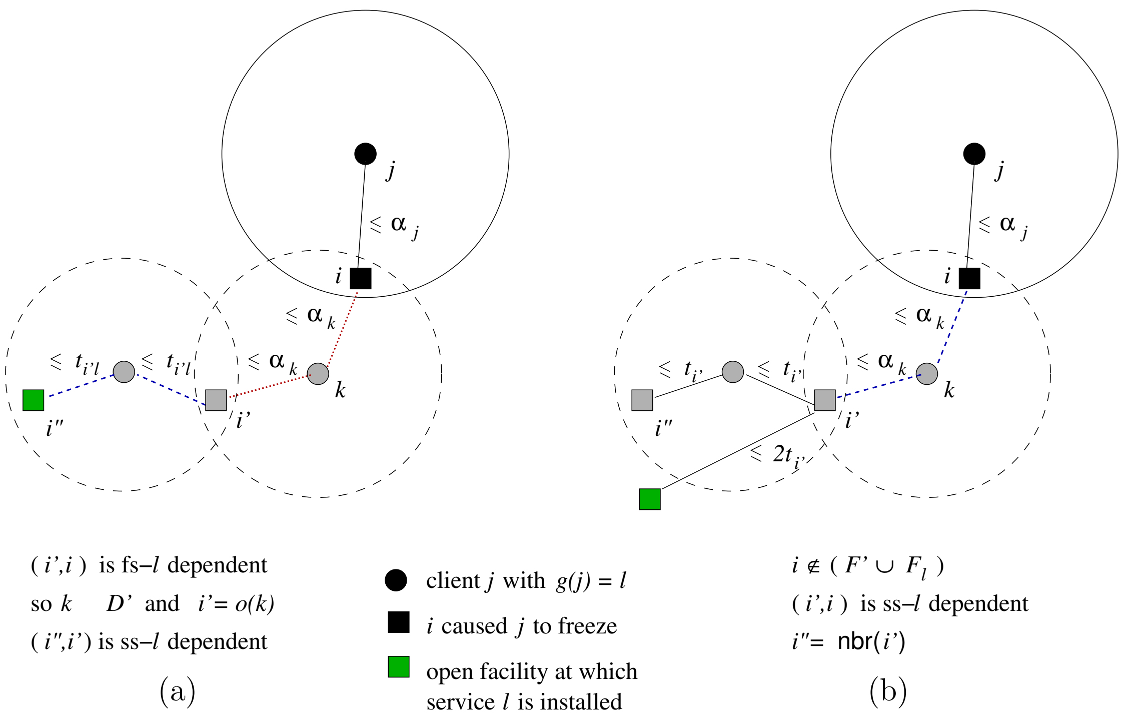

Figure 5.2: The 5-hop cases encountered in Lemma 5.4.9.

Lemma 5.4.9 *If $j \notin D'$, the assignment cost incurred for j is at most $5\alpha_j$.*

Proof : Let $j \notin D'$ with $g(j) = l$. Let i be the facility that caused j to freeze, and so $\alpha_j = \max(c_{ij}, t_i, t_{il})$. There are 4 cases depending on which of the 4 sets i lies in:

$A_l, F'_l, (F' \cap F_l) \setminus A_l$ or $F_l \setminus (F' \cup F'_l)$.

If $i \in A_l$ then service l is installed at i, and $c_{ij} \leq \alpha_j$. If $i \in F'_l$ then service l is installed in step P2, at a facility no further away from i than $i' = \mathsf{nbr}(i)$ since $i' \prec i$ and $i' \in F'$. So the assignment cost is bounded by $c_{ij} + c_{i'i}$. By Claim 5.4.6, $c_{i'i} < 2t_i$, and hence $c_{ij} + c_{i'i} < 3\alpha_j$ (Fig. 5.1a).

Suppose $i \in F' \setminus A_l$. Then there is some $i' \in F'$ due to which service l was removed from S_i in step 2b). If $l \in T_{i'}$ and (i', i) is ss-l dependent due to a demand $k \in G_l$, then service l is installed at i' (see Fig. 5.1b). Applying Claim 5.4.5 we get that $c_{i'k}$ and c_{ik} are both less then $\alpha_k \leq \max(t_i, t_{il}) \leq \alpha_j$, so $c_{i'j} < 3\alpha_j$. In the case where $t_{il} \geq t_i$ and (i', i) is fs-l dependent due to demand $k \in G_l$, we have $k \in D'$, and hence $i' = o(k)$, since $\beta_{i'k} > 0$ (see Fig. 5.2a). Also $\alpha_k \leq \alpha_j$ as above and therefore $c_{kj} < 2\alpha_j$. By Lemma 5.4.8, k is assigned to a facility i'' (so service l is installed at i'') with $c_{i''k} < 3\alpha_k$, so $c_{i''j} < 5\alpha_j$. See Figure 5.2a.

Next suppose that $i \notin F' \cup F'_l$. Since $i \in F_l \setminus F'$ there is some facility i' such that either (1) $i' \in F'$ and (i', i) is fs-l dependent, OR (2) $i' \in A_l$ and (i', i) is ss-l dependent, OR (3) $i' \in F'_l$, (i', i) is ss-l dependent and $t_{i'} \leq t_i$. In either case let $k \in G_l$ be a demand due to which (i', i) is fs-l- or ss-l-dependent. Since $\theta_{ik} > 0$, $\alpha_k \leq \max(t_i, t_{il}) \leq \alpha_j$, so $c_{kj} < 2\alpha_j$. In case (1), $k \in D'$ and by Lemma 5.4.8, k is assigned to a facility i'' with $c_{i''k} < 3\alpha_k \implies c_{i''j} < 5\alpha_j$ (Fig. 5.2a). In case (2), service l is installed at i' and applying Claim 5.4.5, $c_{i'k} < \alpha_k$ which implies $c_{i'j} < 3\alpha_j$ (Fig. 5.1b). Finally in case (3), service l is installed on a facility at least as close to i' as $i'' = \mathsf{nbr}(i')$. Since $c_{i''i'} < 2t_{i'}$ (Claim 5.4.6), and $c_{i'j} < 3\alpha_j$ as in case (2), we get that $c_{i'j} + c_{i'i''} < 5\alpha_j$ (Fig. 5.2b). ■

Theorem 5.4.10 *Let O, I, C denote respectively the facility opening, service installation, and client assignment cost of the solution returned. Then $6O + I + C \leq 6\sum_j \alpha_j \leq 6 \cdot OPT$.*

Proof : Follows from Lemmas 5.4.2, 5.4.8 and 5.4.9 and since for $j \in D'$, $\alpha_j = c_{o(j)j} + \beta_{o(j)j} + \theta_{o(j)j}$. ■

For the case where the service installation cost depends only on the service type, that is, $f_i^l = f^l$, any ordering can serve as the ordering $\mathcal{O}$. Therefore, to install services in step P2 of the algorithm, for every $i \in F'_l$, we can simply install service l at the nearest facility $i' \in F'$ (since every $i' \prec i$). This gives us the following stronger guarantee.

Corollary 5.4.11 *If $f_i^l = f^l$, the algorithm above returns a solution of cost (O, I, C) such that $5O + I + C \leq 5\sum_j \alpha_j \leq 5 \cdot OPT$.*

Proof : Observe that the worst case for the analysis above occurs when a demand j has to "pay" α_j towards the cost of installing services and also incurs an assignment cost of $5\alpha_j$; in all other cases j is charged an amount of at most $5\alpha_j$ (since now a client $j \in D'$ has to pay only $5\beta_{o(j)j}$).

We will show that for any demand j, if there is some facility $i \in A_{g(j)} \cup F'_{g(j)}$ such that $\theta_{ij} > 0$, then the assignment cost incurred for j is at most $5\alpha_j - \theta_{ij}$. Let $\mathcal{C}$ be the set of all such demands, i.e., $\mathcal{C} = \{j : \exists i \in A_{g(j)} \cup F'_{g(j)} \text{ such that } \theta_{ij} > 0\}$; for $j \in \mathcal{C}$ let $s(j)$ denote this unique facility. The proof of Lemma 5.4.2 is easily modified to show that $O = \sum_{j \in D'} \beta_{o(j)j}$, and $I \leq \sum_{j \in \mathcal{C}} \theta_{s(j)j}$. Note that as argued in Lemma 5.4.2, it must be the case that if $j \in D' \cap \mathcal{C}$, then $s(j) = o(j)$. Using Lemmas 5.4.8 and 5.4.9, the quantity $5O+I+C$ is therefore bounded by $\sum_{j \in D'} 5\beta_{o(j)j} + \sum_{j \in \mathcal{C}} \theta_{s(j)j} + \sum_{j \in D'} 3(\alpha_j - \beta_{o(j)j}) + \sum_{j \in \mathcal{C} \setminus D'} (5\alpha_j - \theta_{s(j)j}) + \sum_{j \notin D' \cup \mathcal{C}} 5\alpha_j \leq 5\sum_j \alpha_j$.

Consider $j \in \mathcal{C}$ with $g(j) = l$ and let $i' = s(j)$. If $i' \in A_l$, then the assignment cost is at most $c_{i'j} \leq \alpha_j - \theta_{i'j}$. If $i' \in F'_l$, let i be the facility that caused j to freeze; we have $c_{ii'} \leq 2\alpha_j - \theta_{i'j}$. If $i \in F'$, then service l is installed on a facility at most $c_{ii'}$ distance away from i', so the assignment cost is at most $c_{i'j} + c_{ii'} \leq 3\alpha_j - 2\theta_{i'j}$. If $i \notin F'$, then $\mathsf{nbr}(i) \in F'$ and $c_{i,\mathsf{nbr}(i)} \leq 2t_i \leq 2\alpha_j$. So the assignment cost is at most $c_{i'j} + c_{i'i} + c_{i,\mathsf{nbr}(i)} \leq 5\alpha_j - 2\theta_{i'j}$. ■

5.5 An Improved Algorithm when $f_i^l = f^l$

We use LP rounding to obtain an algorithm with an improved approximation guarantee for the case where the service installation cost f_i^l depends only on the service l and not on the location i. The algorithm is along the lines of the primal rounding algorithm described in Section 2.4. We use randomized rounding along with the bound due to complementary slackness as in the Chudak-Shmoys algorithm (Section 2.3), in conjunction with the bound obtained by filtering [54, 71], to bound the assignment costs. This gives a better performance guarantee than that obtained by using either of the two bounds separately. Sviridenko [73] used a similar idea to improve the approximation ratio for UFL from $\left(1+\frac{2}{e}\right)$ to 1.58.

Let (x, y) and (α, β, θ) be the optimal solutions to (FLS-P) and (FLS-D), respectively. We may assume that each $y_i^l \leq y_i$ since one can always set $y_i^l = \min(y_i^l, y_i)$ and get a feasible solution of no greater cost. We may further assume, by making clones of facilities if necessary as in the CS algorithm, that the optimal solution is *complete*, that is, for every i, j and l, $x_{ij} = 0$ or $x_{ij} = y_i^{g(j)}$ and $y_i^l = 0$ or $y_i^l = y_i$. We will round (x, y) to an integer solution losing a factor of at most 2.391.

Suppose that in the optimal solution it so happened that for every facility i and service l, there is at most one client $j \in G_l$ such that $x_{ij} > 0$. We call a solution with this property a *separable solution*. Then one can view (x, y) as essentially a fractional solution to a UFL instance with distances $c'_{ij} = c_{ij} + f_i^{g(j)}$ (which satisfy the triangle inequality[1] since the service installation cost depends only on the service). Observe the important fact that because (x, y) is separable, the cost of this UFL solution is at most OPT (in fact, the costs are exactly equal due to completeness). Further, an integer solution to the UFL instance yields a solution to the FLSIC instance of no greater cost. So now we simply have to round this UFL solution to an integer solution while losing only a small factor.

The algorithm is based roughly on the above idea. For each service type l, we

[1] $c'_{ij} \leq c'_{i'j} + c'_{i'k} + c'_{ik}$: the triangle inequality takes this form because the metric is the shortest path metric on the complete *bipartite graph* on $\mathcal{F} \cup \mathcal{D}$.

group the facilities on which service l is installed into *disjoint* clusters. Each cluster is centered around a client in G_l and consists of the facilities serving the client; every non-center client in G_l is assigned to a "nearby" center which, in some sense, acts as a representative for all of the clients assigned to it. The instance comprising only the cluster centers has a separable solution induced by (x, y). We now round this solution using a modification of the CS algorithm that incorporates filtering. A point worth emphasizing is that *we do not actually reduce the original FLSIC instance to a UFL instance* on the cluster centers and solve this as a black box. Instead the algorithm performs this reduction only implicitly, whereas the analysis is more refined and relies on the fact that the service installation cost f_i^l is a function of only l to get good bounds.

We define some notation first. Let $F_j = \{i : x_{ij} > 0\}$. Let $0 < \gamma < 1$ be a parameter that we will set later and $r = \frac{1}{\gamma}$. Sort the facilities in F_j by increasing c_{ij} value. Let i' be the first facility in this ordering such that the x_{ij} weight of facilities in F_j that come before i' (including i') is at least γ, that is, $\sum_{i:i=i' \text{ or } i \text{ comes before } i'} x_{ij} \geq \gamma$. Let $N_j \subseteq F_j$ consist of the facilities in F_j up to and including i' in this sorted order. As in Section 2.4, we may assume, by splitting and cloning facilities if necessary, that each $y_i \leq \gamma$ and for any j, $\sum_{i \in N_j} y_i$ is exactly γ. Define $R_j(\gamma) = c_{i'j}$ and let $\bar{C}_j = \sum_i c_{ij} x_{ij}$ denote the cost incurred by the LP solution to assign client j.

R1. For every service type l, we consider the clients in G_l and cluster the facilities on which service l is installed around some cluster centers. Pick $j \in G_l$ with smallest $2\alpha_j + R_j(\gamma) + \bar{C}_j$ value and form a cluster around j consisting of the facilities in F_j. We assign every client $k \in G_l$ (including j) that is served (fractionally) by some facility in the cluster created (i.e., F_j) to j, and remove it from G_l (see Fig 5.3a). Recurse on the remaining set of clients until no client in G_l is left. This gives a set of cluster centers D_l for each service l. For a client $k \notin D_l$ let $\sigma(k)$ denote the cluster center in D_l to which it is assigned.

R2. Let $D = \bigcup_l D_l$. We cannot open a facility in every cluster since different clusters could share the same fractional facility weight (y_i) if the cluster centers request

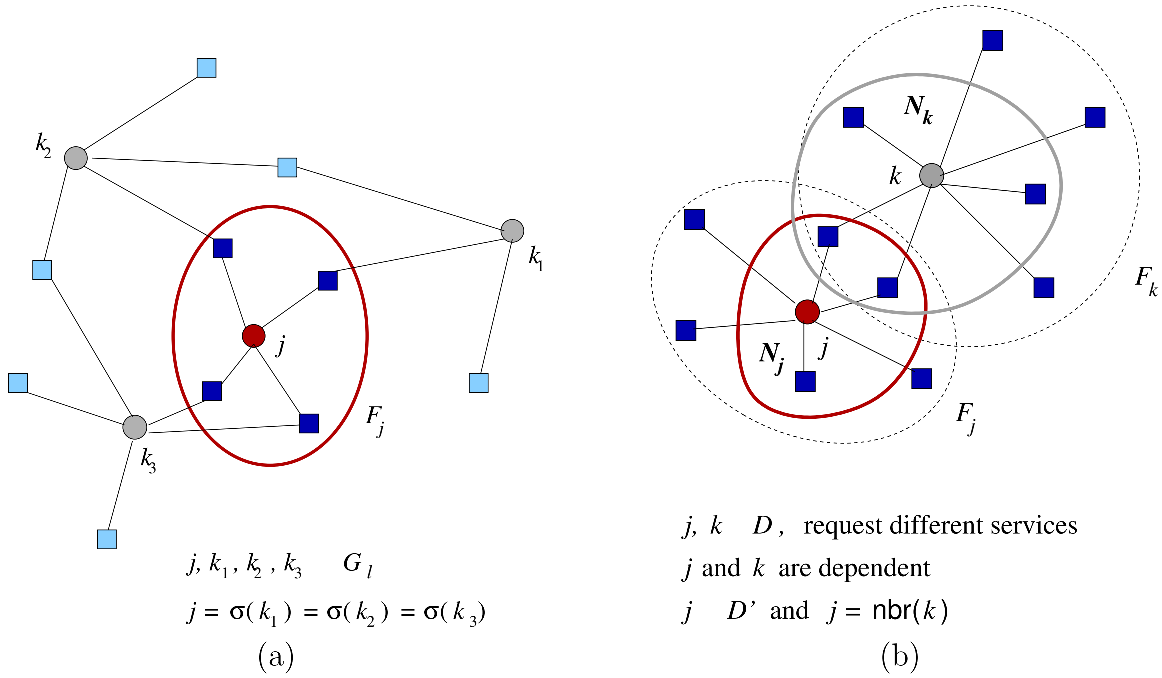

Figure 5.3: (a) An iteration of step R1. (b) Picking a maximal independent set in step R2.

different services. Say that $j, k \in D$ are dependent if $N_j \cap N_k \neq \emptyset$. Note that this can only happen if j and k request different services. Consider clients in D in increasing order of $R_j(\gamma) + \bar{C}_j$ and greedily pick a maximal independent subset D'. We denote the facilities in N_j for clients $j \in D'$ as *central facilities*, and the rest as *non-central facilities*. For every client $k \in D \setminus D'$, there is some $j \in D'$ that was picked before k such that j and k are dependent. Call j the neighbor of k and denote it by $\mathsf{nbr}(k)$ (see Fig. 5.3b). For convenience, we set $\mathsf{nbr}(j) = j$.

R3. For every client $j \in D'$ we randomly open exactly one facility in N_j by choosing facility i with probability $y_i / \sum_{i \in N_j} y_i = r \cdot y_i$. We denote this facility as the *backup facility* for every client k with $\mathsf{nbr}(k) = j$.

R4. Independent of step R3, each non-central facility i is opened independently with probability $r \cdot y_i$.

R5. For *any* facility i, be it a central or a non-central facility, if i is opened (in R3 or

R4), we install on it all services that are installed on it in the fractional solution, i.e., all l such that $y_i^l > 0$.

R6. For every client $j \in D \setminus D'$, if no facility from F_j is open, we install service $g(j)$ on its backup facility, i.e., the facility opened in R3 from $N_{\mathsf{nbr}(j)}$.

R7. We assign client j to the nearest open facility at which service $g(j)$ is installed.

5.5.1 Analysis

Let Z_j denote the event that no facility in F_j is open, and $\bar{Z}_j$ denote the complementary event that some facility in F_j is open. The following claim will be used repeatedly.

Claim 5.5.1 *For any client j, $\Pr[Z_j] \leq e^{-r}$.*

Proof : If $j \in D'$ then $Z_j = \emptyset$, since we always open a facility from $N_j \subseteq F_j$. Otherwise, we can express Z_j in terms of the following events. For every non-central facility $i \in F_j$, let E_i be the event that i is opened in step R4 and $p_i = \Pr[E_i] = r \cdot y_i$. For every cluster center $k \in D'$ such that $S_k = F_j \cap N_k \neq \emptyset$, let E_k be the event that a facility in S_k is open after step R3. Let $p_k = \Pr[E_k] = \frac{\sum_{i \in S_k} y_i}{\sum_{i \in N_k} y_i} = r \cdot \sum_{i \in S_k} y_i$. Let m be the total number of events. Observe that the events E_i are all independent and $Z_j = \bigcap_{n=1}^m \bar{E}_n$. Therefore, $\Pr[Z_j] = \prod_{n=1}^m (1 - p_n) \leq e^{-\sum_n p_n} = e^{-r}$. ■

Lemma 5.5.2 *The expected facility opening cost is $r \cdot \sum_i f_i y_i$. The expected cost of installing services is at most $(r + e^{-r}) \cdot \sum_{i,l} f_i^l y_i^l$.*

Proof : Each facility i is opened with probability $r \cdot y_i$. The expected cost of installing services in step R5 is bounded by,

$$\sum_i \Pr[i \text{ is opened (in R3 or R4)}] \sum_{l: y_i^l > 0} f_i^l = \sum_i r \cdot y_i \sum_{l: y_i^l > 0} f_i^l = r \cdot \sum_{i,l} f_i^l y_i^l,$$

since $y_i^l > 0 \implies y_i^l = y_i$ by completeness.

For a client $j \in D \setminus D'$, the probability that none of the facilities in F_j is open is $\Pr[Z_j] \le e^{-r}$ by Claim 5.5.1. So the expected cost of installing services in step R6 is at most $e^{-r} \sum_{j \in D \setminus D'} f^{g(j)} \le e^{-r} \sum_{i,l} f_i^l y_i^l$ since $\sum_{i \in F_j} y_i^{g(j)} = 1$ and any two clients j and j' in D_l must have $F_j \cap F_{j'} = \emptyset$. ■

To bound the assignment cost, we consider a provably worse way of assigning clients to facilities and bound the cost incurred under this scheme. We assign client j as follows. If $j \in D'$ we assign it to the facility opened from N_j. If $j \notin D'$, we assign j to the nearest open facility in F_j if some facility in F_j is open. Otherwise if $j \in D \setminus D'$, we assign it to its backup facility; if $j \notin D$, we assign it to the same facility as $k = \sigma(j)$, so it may be assigned either to a facility in F_k or, if no facility from F_k is open, to the backup facility for k in $N_{\mathsf{nbr}(k)}$. Observe that service $g(j)$ is always installed on the facility to which j is assigned. Let X_j be the random variable denoting the assignment cost of j under the above scheme. Recall the following lemma from Section 2.3.

Lemma 2.3.3 *Let $d_1 \le d_2 \le \ldots \le d_m$ and $0 \le p_n \le 1$ for $n = 1, \ldots, m$. Then,*

$$p_1 d_1 + (1-p_1)p_2 d_2 + \cdots + (1-p_1)(1-p_2)\ldots(1-p_{m-1})p_m d_m$$
$$\le \frac{\sum_{n \le m} p_n d_n}{\sum_{n \le m} p_n} \Big(1 - \prod_{n \le m}(1-p_n)\Big).$$

Lemma 5.5.3 *For any client j, $\mathrm{E}\big[X_j | \bar{Z}_j\big] \le \bar{C}_j$.*

Proof : If $j \in D'$, then event $\bar{Z}_j$ always occurs, so

$$\mathrm{E}\big[X_j | \bar{Z}_j\big] = \mathrm{E}\big[X_j\big] = \frac{\sum_{i \in N_j} c_{ij} x_{ij}}{\sum_{i \in N_j} x_{ij}} \le \bar{C}_j,$$

since every facility in $F_j \setminus N_j$ is farther from j than every facility in N_j.

So consider $j \notin D'$. For a non-central facility $i \in F_j$, let p_i and E_i be as defined in Claim 5.5.1 and let $d_i = c_{ij}$. For every $k \in D'$ such that $S_k = F_j \cap N_k \neq \emptyset$, let p_k, E_k be as defined in Lemma 5.5.2 and define $d_k = \sum_{i \in S_k} c_{ij} y_i / \sum_{i \in S_k} y_i$, which is the expected distance between j and the facility opened from S_k conditioned on event E_k. Let the distances be ordered so that $d_1 \leq d_2 \leq \ldots \leq d_m$ where m is the total number of events. We will obtain an upper bound on $\mathrm{E}\big[X_j\big]$ by considering a suboptimal way of assigning j to an open facility in F_j (if one exists). Instead of assigning j to the nearest open facility in F_j, we will assign it to the open "facility" with smallest d_i, where when we say that a "facility" of type $S_k, k \in D'$ is open, we mean that some facility $i \in S_k$ is open, and assigning j to facility S_k means that we assign it to the open facility i in S_k. Let $p = \Pr[Z_j]$. Since the events E_i are independent and $y_i = x_{ij}$,

$$\begin{aligned}
\mathrm{E}\big[X_j\big] &\leq p_1 d_1 + (1-p_1)p_2 d_2 + \cdots + (1-p_1)\ldots(1-p_{m-1})p_m d_m + p \cdot \mathrm{E}\big[X_j | Z_j\big] \\
&\leq \frac{\sum_{n \leq m} p_n d_n}{\sum_{n \leq m} p_n}(1-p) + p \cdot \mathrm{E}\big[X_j | Z_j\big] \\
&= \Pr[\bar{Z}_j] \cdot \bar{C}_j + \Pr[Z_j] \cdot \mathrm{E}\big[X_j | Z_j\big].
\end{aligned}$$

But we also know that $\Pr[\bar{Z}_j] \cdot \mathrm{E}\big[X_j | \bar{Z}_j\big] + \Pr[Z_j] \cdot \mathrm{E}\big[X_j | Z_j\big] = \mathrm{E}\big[X_j\big]$. Combining this with the above inequality, we get that $\mathrm{E}\big[X_j | \bar{Z}_j\big] \leq \bar{C}_j$. ■

Lemma 5.5.4 *For a client $j \notin D'$, we have $\mathrm{E}\big[X_j | Z_j\big] \leq 3\alpha_j + R_j(\gamma) + \bar{C}_j$.*

Proof : First suppose that j is in $D \setminus D'$. Let $k = \mathsf{nbr}(j)$ and $A = F_j \cap N_k \neq \emptyset$. For any facility $i \in A$ we have $c_{ij} \leq \alpha_j$ due to complementary slackness, since $i \in F_j$. Also, for any facility $i \in N_k$, we have $c_{ik} \leq R_k(\gamma)$ by the definition of $R_k(\gamma)$. Event Z_j implies that j is assigned to its backup facility in N_k, so conditioned on Z_j, X_j is at most $c_{jk} + X_k \leq \alpha_j + c(A,k) + X_k$ where $c(A,k)$ denotes $\min_{i \in A} c_{ik}$. If there is some facility $i \in A$ such that $c_{ik} \leq \bar{C}_k$, then $c(A,k) \leq \bar{C}_k$ and we have a bound of $X_j \leq \alpha_j + \bar{C}_k + R_k(\gamma)$. Otherwise, since the unconditional expectation $\mathrm{E}\big[X_k\big]$ is at most $\bar{C}_k$, by conditioning on Z_j, we are only removing weight from facilities that have a larger c_{ik} value than the average. So the conditional expectation $\mathrm{E}\big[X_k | Z_j\big]$ is at most $\bar{C}_k$ and it follows that $\mathrm{E}\big[X_j | Z_j\big] \leq \alpha_j + R_k(\gamma) + \bar{C}_k$. In either case, since $R_k(\gamma) + \bar{C}_k \leq R_j(\gamma) + \bar{C}_j$ as k was picked before j in step R2, we get that $\mathrm{E}\big[X_j | Z_j\big] \leq \alpha_j + R_j(\gamma) + \bar{C}_j$.

The argument for the case when j is not in D is similar. Let $j' = \sigma(j)$, so $F_j \cap F_{j'} \neq \emptyset$. Event Z_j implies that j is assigned to the same facility as j'. If a facility in $F_{j'}$ is open then we have a bound of $X_j \leq \alpha_j + 2\alpha_{j'}$. Otherwise, we are in event $Z_{j'}$. Let $k = \mathsf{nbr}(j')$. Clients j and j' are assigned to the backup facility for j' in N_k, so we have $\mathrm{E}\big[X_j | Z_j \cap Z_{j'}\big] \leq c_{jk} + \mathrm{E}\big[X_k | Z_j \cap Z_{j'}\big]$. Taking $A = (F_j \cup F_{j'}) \cap N_k$, we get that $\mathrm{E}\big[X_j | Z_j \cap Z_{j'}\big]$ is at most $\alpha_j + 2\alpha_{j'} + c(A, k) + \mathrm{E}\big[X_k | Z_j \cap Z_{j'}\big]$ which, by reasoning exactly as above, we can bound by $\alpha_j + 2\alpha_{j'} + R_k(\gamma) + \bar{C}_k$. So, in either case, we obtain that $\mathrm{E}\big[X_j | Z_j\big] \leq \alpha_j + 2\alpha_{j'} + R_k(\gamma) + \bar{C}_k \leq \alpha_j + 2\alpha_{j'} + R_{j'}(\gamma) + \bar{C}_{j'}$. Finally, since $j' = \sigma(j)$ we have that $2\alpha_{j'} + R_{j'}(\gamma) + \bar{C}_{j'} \leq 2\alpha_j + R_j(\gamma) + \bar{C}_j$ by our rule of picking cluster centers in step R1. So $\mathrm{E}\big[X_j | Z_j\big] \leq 3\alpha_j + R_j(\gamma) + \bar{C}_j$. ■

Theorem 5.5.5 *The randomized algorithm returns a solution of expected cost at most,* $\Big(\max\big(r + e^{-r}, 1 + \frac{e^{-r}}{1-\gamma}\big) + 3e^{-r}\Big) \cdot OPT$*, where* $r = 1/\gamma$*. Setting* $\gamma = 0.67674$*, we get a solution of cost at most* $2.391 \cdot OPT$*.*

Proof : We have $\sum_i f_i y_i + \sum_{i,l} f_i^l y_i^l + \sum_j \bar{C}_j = OPT = \sum_j \alpha_j$. By Lemmas 5.5.3 and 5.5.4, we get that

$$\mathrm{E}\big[X_j\big] \leq \Pr[\bar{Z}_j] \cdot \bar{C}_j + \Pr[Z_j] \cdot \big(3\alpha_j + R_j(\gamma) + \bar{C}_j\big) \leq \bar{C}_j + e^{-r}\Big(3\alpha_j + \frac{\bar{C}_j}{1-\gamma}\Big)$$

where we use the fact that $\Pr[Z_j] \leq e^{-r}$ (Claim 5.5.1) and $R_j(\gamma) \leq \frac{\bar{C}_j}{1-\gamma}$ by Markov's inequality. Using Lemma 5.5.2, the theorem now follows. ■

5.6 The 2-Stage Stochastic FLSIC Problem

In this section we study the 2-stage stochastic facility location with service installation costs problem. In this problem, we are given a set of facilities $\mathcal{F}$, a set of clients $\mathcal{D}$ and a set of services $\mathcal{S}$. Each client j requests a specific service $g(j) \in \mathcal{S}$. The set of clients that have to be assigned is not known in advance but is specified by a probability distribution; each scenario specifies a set of clients $A \subseteq \mathcal{D}$ that have to be assigned[2]. We may choose to open some facilities and install some services in stage I. The cost that we pay in stage I is f_i^{I} for opening facility i, and $f_{i,l}^{\mathrm{I}}$ for installing service l at (an open) facility i. Once we know the scenario A that materializes, we may open some more facilities paying a cost of f_i^A for opening facility i, and install some services paying a cost of $f_{i,l}^A$ for installing service l at facility i (this service could be installed at a facility opened either in stage I or in scenario A), and have to assign each client j that is activated to a facility at which service $g(j)$ is installed. The objective is to decide which facilities to open and which services to install in stage I, so as to minimize the sum of the stage I cost and the expectation over all scenarios of the stage II cost.

We can write the following stochastic program for this problem. Variable y_i denotes if facility i is opened in stage I, and variable $y_{i,l}$ denotes if service l is installed at facility i in stage I. As usual, we use i to index the facilities, j to index the clients and l to index the services in $\mathcal{S}$.

$$\min \quad \sum_i f_i^{\mathrm{I}} y_i + \sum_{i,l} f_{i,l}^{\mathrm{I}} y_{i,l} + h(y) \quad \text{subject to} \quad 0 \le y_{i,l} \le y_i \le 1 \quad \text{for all } i,l, \tag{SFLS-P}$$

$$\begin{aligned}
\text{where} \quad h(y) &= \sum_{A \subseteq \mathcal{D}} p_A h_A(y), \\
\text{and} \quad h_A(y) &= \min \quad \sum_i f_i^A y_{A,i} + \sum_{i,l} f_{i,l}^A y_{A,i,l} + \sum_{j \in A} \sum_i c_{ij} x_{A,ij}
\end{aligned}$$

$$\begin{aligned}
\text{s.t.} \quad \sum_i x_{A,ij} &\ge 1 && \text{for all } j \in A, \\
x_{A,ij} &\le y_{i,g(j)} + y_{A,i,g(j)} && \text{for all } i, j \in A, && (3) \\
x_{A,ij} &\le y_i + y_{A,i} && \text{for all } i, j \in A, && (4) \\
x_{A,ij}, y_{A,i}, y_{A,i,l} &\ge 0 && \text{for all } i, j \in A, l.
\end{aligned}$$

In any given scenario A, we solve a minimization problem to decide which facilities to open and services to install in that scenario, and how to assign the activated clients. Variable $y_{A,i}$ indicates whether we open facility i in scenario A, $y_{A,i,l}$ indicates if service

[2]In general there may be arbitrary demands associated with the clients, but for simplicity, we consider the 0-1 demand setting. So a client is either activated or not activated in a scenario.

l is installed at facility i in scenario A, and $x_{A,ij}$ indicates if client j is assigned to facility i. Constraints (3) and (4) state that a client that is activated in a scenario must be assigned to a facility opened either in stage I or in that particular scenario, and the service that it requires should be installed on that facility, again either in stage I or in that scenario. In stage I, of course, we may only install a service l at a facility i if we open that facility in stage I, which is captured by the constraint $y_{i,l} \leq y_i$. (SFLS-P) lies in the general class of 2-stage programs described in Section 3.5, and using Theorem 3.5.4 one can therefore compute a near-optimal solution y to (SFLS-P) in polynomial time.

5.6.1 Rounding the Near-Optimal Solution

We show that if the service installation cost depends only on the service type and not on the location, both in stage I and in every stage II scenario, that is, $f^{\mathrm{I}}_{i,l} = f^{\mathrm{I}}_{l}$ and $f^{A}_{i,l} = f^{A}_{l}$, then one can round y using ideas from the rounding procedure of Section 4.3.2 and get a 11.363-approximation algorithm.

The challenging aspect in the rounding is the "decoupling" of the first-stage and second-stage decisions which turns out to be somewhat involved, as compared to the rounding scheme in Section 4.3.2, because of the following artifact: it might be that a client j in scenario A is "mostly" assigned to facilities opened in stage I by the fractional solution, but the service $g(j)$ required by the client is however "mostly" installed only in scenario A. Our main theorem is the following.

Theorem 5.6.1 *The fractional solution y can be rounded losing a factor of at most* 11.363*. This gives a $(11.363 + \epsilon)$-approximation algorithm for the stochastic version of facility location with service installation costs.*

The following decomposition technique of Shmoys, Tardos & Aardal [71] will play a useful role in our rounding procedure. Given a fractional UFL solution $(\hat{x}, \hat{y})$ the STA algorithm looks at a subset of the facilities serving each client j that are "close" to j, and clusters these facilities around some centers. The clustering is performed essentially as in step A1 of the CS algorithm in Section 2.3 but using a different cluster selection rule.

A Generic Decomposition Algorithm. The algorithm takes two parameters: $\gamma \in [0,1]$ and a number $w(j) \geq 0$ for each client j. Let $\bar{C}_j = \sum_i c_{ij}\hat{x}_{ij}$. For each client j, order the facilities with $\hat{x}_{ij} > 0$ by increasing distance from j, and let T_j be the minimal set of facilities considered in this order that gather an $\hat{x}_{ij}$-weight of at least γ. Let i' be the facility in T_j farthest from j, and let $R_j(\gamma)$ denote $c_{i'j}$. The expression $\sum_i c_{ij}\hat{x}_{ij}$ assigns an $\hat{x}_{ij}$-weight of at least $1-\gamma$ to facilities that facilities that are at least $c_{i'j}$ distance away from j, so we have $\bar{C}_j \geq (1-\gamma)R_j(\gamma)$.

Let $\mathcal{L}$ be the list of clients ordered by increasing $\bar{C}_j/(1-\gamma)+w(j)$ value. We pick the first client in $\mathcal{L}$, that is, the one with minimum $\bar{C}_j/(1-\gamma)+w(j)$ value, and create a cluster around it consisting of the facilities in T_j. Each client k such that $T_k \cap T_j \neq \emptyset$ is then removed from $\mathcal{L}$, and we designate j as the representative of each such client k and set $\sigma(k) = j$. For the cluster center j, we set $\sigma(j) = j$. We then continue with the remaining list of clients until $\mathcal{L}$ becomes empty.

By construction, the clusters are disjoint and for each cluster T_j we have $\sum_{i \in T_j} y_i \geq \gamma$. The following lemma will be used repeatedly.

Lemma 5.6.2 *For any client k, we have $c_{\sigma(k)k} \leq (\bar{C}_{\sigma(k)} + \bar{C}_k)/(1-\gamma)$.*

Proof : Let $j = \sigma(k)$. So, $T_k \cap T_j \neq \emptyset$. Let $i \in T_k \cap T_j$. Then $c_{jk} \leq c_{ij} + c_{ik} \leq R_j(\gamma) + R_k(\gamma) \leq (\bar{C}_j + \bar{C}_k)/(1-\gamma)$. ∎

Now fix a scenario A and a client $j \in A$. Let $g(j) = l$. We write $x_{A,ij} = x^{\mathrm{I}}_{A,ij} + x^{\mathrm{II}}_{A,ij} + t_{A,ij}$ where $x^{\mathrm{I}}_{A,ij} = \min(x_{A,ij}, y_{i,l})$, $x^{\mathrm{II}}_{A,ij} = \min(x_{A,ij} - x^{\mathrm{I}}_{A,ij}, y_{A,i,l}, y_{A,i})$ and $t_{A,ij} = x_{A,ij} - x^{\mathrm{I}}_{A,ij} - x^{\mathrm{II}}_{A,ij}$. Note that, $0 \leq x^{\mathrm{I}}_{A,ij} \leq y_{i,l} \leq y_i$, $0 \leq x^{\mathrm{II}}_{A,ij} \leq \min(y_{A,i,l}, y_{A,i})$ and $t_{A,ij} \geq 0$. Moreover we have, $x^{\mathrm{I}}_{A,ij} + t_{A,ij} \leq y_i$, and $x^{\mathrm{II}}_{A,ij} + t_{A,ij} \leq y_{A,i,l}$. Observe that j must be assigned to an extent of at least $\frac{1}{3}$ by at least one of the assignments $\{x^{\mathrm{I}}_{A,ij}\}$, $\{x^{\mathrm{II}}_{A,ij}\}$, or $\{t_{A,ij}\}$. Intuitively, if j is assigned to an extent of at least $\frac{1}{3}$ by the assignment $\{x^{\mathrm{I}}_{A,ij}\}$ then we can take care of j by assigning it to facilities opened and services installed in stage I; otherwise, if j is assigned to an extent of at least $\frac{1}{3}$ by the assignment $\{x^{\mathrm{II}}_{A,ij}\}$ then we can take care of j by the facilities that we open and services that we install in scenario A. It is the last case, where j is "mostly" assigned due to $\{t_{A,ij}\}$ that is complicated. Here we need to open a facility in stage I to serve j, but we will install the service required by j only in scenario A. Therefore, in this case, we are only able to partially decouple the two stages.

Let $\mathcal{R}_j$ be the collection of scenarios $\{A \subseteq \mathcal{D} : \sum_i x^{\mathrm{I}}_{A,ij} \geq \frac{1}{3}\}$, $\mathcal{T}_j = \{A \subseteq \mathcal{D} : A \notin \mathcal{R}_j, \sum_i x^{\mathrm{II}}_{A,ij} \geq \frac{1}{3}\}$ and let $\mathcal{U}_j$ be the remaining collection of scenarios $\{A \subseteq D : j \in A \text{ and } A \notin (\mathcal{R}_j \cup \mathcal{T}_j)\}$. Define $\rho(\gamma) = 1 + e^{-1/\gamma} \cdot \frac{2+\gamma}{1-\gamma}$.

Opening facilities in stage I. To decide which facilities to open in stage I, we will ignore the different service requirements momentarily, and solve a UFL problem in which the facility costs are f^{I}_i and each client j has demand $\sum_{A \in \mathcal{R}_j \cup \mathcal{U}_j} p_A$. We shall construct a feasible fractional solution for this instance and use the primal-rounding algorithm of Section 2.4, which does not require any knowledge of the client demands, to round this fractional solution to an integer solution. First, consider each scenario $A \in \mathcal{R}_j \cup \mathcal{U}_j$ separately and create a client (j, A) for each such scenario with demand p_A. Since j is assigned to an extent of at least $\frac{1}{3}$ by the assignment $\{x^{\mathrm{I}}_{A,ij} + t_{A,ij}\}$, we can obtain a feasible assignment $\hat{x}$ (that assigns each client j to an extent of at least 1) by setting $\hat{x}_{A,ij} = \min\big(1, 3(x^{\mathrm{I}}_{A,ij} + t_{A,ij})\big)$ for each $i \in \mathcal{F}$. Since $x^{\mathrm{I}}_{A,ij} + t_{A,ij} \leq y_i$ for each facility i, we can set $\hat{y}_i = \min(1, 3y_i)$, to get a feasible fractional solution $(\hat{x}, \hat{y})$ for the input with client set $\{(j, A) : j \in \mathcal{D}, A \in \mathcal{R}_j \cup \mathcal{U}_j\}$. But given these fractional $\hat{y}_i$ values, one can re-optimize and get a fractional assignment $\hat{x}_{A,ij}$ that minimizes $\sum_i p_A c_{ij} \hat{x}_{A,ij}$ for each (j, A). Observe that this fractional assignment is independent of the scenario A, so we can coalesce all the clients (j, A) for $A \in \mathcal{R}_j \cup \mathcal{U}_j$ into one single client j with demand $\sum_{A \in \mathcal{R}_j \cup \mathcal{U}_j} p_A$. If $\hat{C}_j$ denotes the re-optimized per-demand assignment cost $\sum_i c_{ij} \hat{x}_{ij}$, then we have

$$\hat{C}_j \leq 3 \sum_i c_{ij}(x^{\mathrm{I}}_{A,ij} + t_{A,ij}) \leq 3 \sum_i c_{ij} x_{A,ij} \quad \text{for every scenario } A \in \mathcal{R}_j \cup \mathcal{U}_j \tag{5}$$

The fractional solution so constructed has facility cost at most $3\sum_i f_i^{\mathrm{I}} y_i$ and assignment cost at most $3\sum_{i,j}\sum_{A\in\mathcal{R}_j\cup\mathcal{U}_j} p_A c_{ij} x_{A,ij}$.

Let $0<\gamma<1$ be a parameter that we will set later. We round $(\hat{x},\hat{y})$ using the primal-rounding algorithm with parameter γ to get an integer solution $(\tilde{x},\tilde{y})$ of facility cost at most $\frac{3}{\gamma}\cdot\sum_i f_i^{\mathrm{I}} y_i$ and assignment cost at most $3\rho(\gamma)\cdot\sum_{i,j,A\in\mathcal{R}_j\cup\mathcal{U}_j} p_A c_{ij} x_{A,ij}$ (Theorem 2.4.2); this determines the set of facilities to open in stage I. Let $i(j)$ denote the open facility that is nearest to j. By Lemma 2.4.1, we also know that for every client j, the expected distance $\mathrm{E}\left[c_{i(j)j}\right]$ is at most $\rho(\gamma)\cdot\hat{C}_j$.

Installing services in stage I. Next we determine where to install services in stage I. Fix a service l and consider the clients in G_l. Consider a UFL instance with client set G_l, and $\mathcal{F}$ as the set of facilities. Each client $j\in G_l$ has demand $\sum_{A\in\mathcal{R}_j} p_A$. We construct a feasible fractional solution (x',y') for this instance by setting $y'_i=\min(1,3y_{i,l})$, which consequently also determines the assignment variables x'_{ij}. For any scenario $A\in\mathcal{R}_j$, we have $\sum_i x^{\mathrm{I}}_{A,ij}\ge\frac{1}{3}$ and $x^{\mathrm{I}}_{A,ij}\le y_{i,l}$, therefore arguing as before we get that the per-unit-demand assignment cost of j, given by $C'_j=\sum_i c_{ij}x'_{ij}$, is at most $3\sum_i c_{ij}x_{A,ij}$. Also, since for every facility $i\in\mathcal{F}$ we have that $y'_i\le\hat{y}_i$ (since $y_{i,l}\le y_i$), $\hat{C}_j$, which was obtained by re-optimizing the assignment distance with respect to the $\hat{y}_i$ values, is at most C'_j. Now we run the decomposition algorithm described above with parameter γ and with $w(j)=c_{i(j)j}$ (which is a random variable). Since the clusters are all disjoint and each cluster has a y'_i-weight of at least γ, we can afford to install service l for each cluster created. For every cluster center j, we install service l on facility $i=i(j)$, that is, we set $\tilde{y}_{i,l}=1$. This determines the facilities at which we install service l in stage I. Doing this for every service l, tells us where to install services in stage I. Note that since the service installation cost does not depend on the location, we can pay for installing service l by the $y_{i,l}$-weight contained in the cluster around j. So, the cost for installing service l is at most $f_l^{\mathrm{I}}\sum_i y'_i/\gamma=\frac{3}{\gamma}\cdot\sum_i f_{i,l}^{\mathrm{I}} y_{i,l}$ and the total cost of installing services in stage I is at most $\frac{3}{\gamma}\cdot\sum_{i,l} f_{i,l}^{\mathrm{I}} y_{i,l}$.

Observe that for any client $k \in G_l$ with $j = \sigma(k)$, in any scenario $A \in \mathcal{R}_k$, there is an open facility i at which service l is installed at a distance of at most $c_{jk}+c_{i(j)j} \leq (C'_j+C'_k)/(1-\gamma)+w(j)$ by Lemma 5.6.2, which in turn is at most $2C'_k/(1-\gamma) + w(k)$ because we picked j before k as a cluster center in our decomposition procedure. So we can bound the expected per-unit-demand assignment cost of k by $\frac{2C'_k}{1-\gamma} + \mathrm{E}\big[c_{i(k)k}\big] \leq \big(\frac{2}{1-\gamma} + \rho(\gamma)\big)C'_k$. So in every scenario $A \in \mathcal{R}_k$, we can assign client k to facility i and the net cost we incur over all scenarios in $\mathcal{R}_k$ is bounded by $\big(\frac{6}{1-\gamma} + 3\rho(\gamma)\big)\big(\sum_{i,A\in\mathcal{R}_k} p_A c_{ik} x_{A,ik}\big)$. This takes care of scenarios in $\mathcal{R}_k$ for each client k.

Opening facilities, installing services in a stage II scenario. Consider a scenario A. We will show that given the first stage decisions $\tilde{y}$, one can assign the clients in A without incurring a large cost. We have already taken care of each client j such that $A \in \mathcal{R}_j$ by assigning it to a stage I facility at which service $g(j)$ is installed (in stage I). To assign the remaining clients we will again solve a UFL instance to decide which facilities to open, and then use the decomposition algorithm to guide the installation of services.

Consider the client set $D_A = \{j \in A : A \in \mathcal{T}_j\}$. To decide which facilities to open, we will again ignore the service requirements. We solve a UFL problem with client set D_A, and $\mathcal{F}$ as the set of facilities where the cost of facility i is f_i^A. We can construct a feasible fractional solution $(\hat{x}_A, \hat{y}_A)$ for this instance as follows: set $\hat{x}_{A,ij} = \min(1, 3x^{\mathrm{II}}_{A,ij})$ and $\hat{y}_{A,i} = \min(1, 3y_{A,i})$ for each $i \in \mathcal{F}$. We round $(\hat{x}_A, \hat{y}_A)$ using the primal-rounding algorithm with parameter γ to get an integer solution $(\tilde{x}_A, \tilde{y}_A)$ such that

$$\sum_i f_i^A \tilde{y}_{A,i} \;\leq\; \tfrac{3}{\gamma} \cdot \sum_i f_i^A y_{A,i} \quad \text{and} \quad \sum_{i,j\in D_A} c_{ij}\tilde{x}_{A,ij} \;\leq\; 3\rho(\gamma) \cdot \sum_{i,j\in D_A} c_{ij}x_{A,ij}. \qquad (6)$$

This determines which facilities to open in scenario A.

To decide where to install services we will again use the decomposition algorithm. Let $i(j)$ denote the facility nearest to j that is open, either in stage I or in scenario A. Observe that for a client $j \in D_A$ we have $c_{i(j)j} \leq \sum_i c_{ij}\tilde{x}_{A,ij}$. For a client $j \in A \setminus D_A$

such that $A \notin \mathcal{R}_j$ we have $A \in \mathcal{U}_j$; the distance $c_{i(j)j}$ is at most the distance to the nearest stage I facility, therefore $\mathrm{E}\big[c_{i(j)j}\big] \leq \rho(\gamma) \cdot \hat{C}_j$, and (5) provides a bound on $\hat{C}_j$. For every service l, we consider the client set $G'_l = \{j \in G_l \cap A : A \notin \mathcal{R}_j\}$. As before, we construct a feasible solution (x'_A, y'_A) for this instance and feed this into the decomposition algorithm to get a clustering. We set $y'_{A,i} = \min(1, 3y_{A,i,l})$ and $x'_{A,ij} = \min\big(1, 3(x^{\mathrm{II}}_{A,ij} + t_{A,ij})\big)$ for each client $j \in G'_l$. Observe that this gives a feasible UFL solution. We run the decomposition algorithm on (x'_A, y'_A) with parameters γ and $w(j) = c_{i(j)j}$ to create a set of clusters. For each cluster centered around client j, we install service l on facility $i = i(j)$ (which is open), that is, we set $\tilde{y}_{A,i,l} = 1$. This tells us the facilities at which to install service l in scenario A. Repeating this for every service type determines the services that we install in scenario A.

Bounding the cost for a scenario We now show that the cost incurred for scenario A is bounded. For any service l, each cluster created by the decomposition procedure contains a $y_{A,i,l}$-weight of at least $\frac{\gamma}{3}$, therefore the total cost of installing all the services is bounded by $\frac{3}{\gamma} \cdot \sum_{i,l} f^A_{i,l} y_{A,i,l}$. To bound the assignment cost, consider a client $k \in A$ such that $A \notin \mathcal{R}_k$, with $j = \sigma(k)$. We know that service l is installed at a facility at a distance of at most $c_{jk} + c_{i(j)j}$. Therefore, arguing as we did earlier, we can bound this distance $2\big(\sum_i c_{ik} x'_{A,ik}\big)/(1-\gamma) + c_{i(k)k}$, and for $k \in A \setminus D_A$, we can bound $\mathrm{E}\big[c_{i(k)k}\big]$ by $\rho(\gamma) \cdot \hat{C}_k$. So the expected cost incurred for scenario A ignoring the clients $j \in A$ such that $A \in \mathcal{R}_j$, is at most

$$\sum_i f^A_i \tilde{y}_{A,i} + \tfrac{3}{\gamma} \cdot \sum_{i,l} f^A_{i,l} y_{A,i,l} + \tfrac{6}{1-\gamma} \cdot \sum_{i,j \in A : A \notin \mathcal{R}_j} c_{ij} x_{A,ij} + \sum_{i,j \in D_A} c_{ij} \tilde{x}_{A,ij} + 3\rho(\gamma) \cdot \sum_{i,j \in A : A \in \mathcal{U}_j} c_{ij} x_{A,ij} \quad (7)$$

where we use (5) to bound $\hat{C}_j$ for clients $j \in A$ such that $A \in \mathcal{U}_j$. Substituting the bounds from (6) in the above expression, and since the expected assignment cost of a client j such that $A \in \mathcal{R}_j$ is at most $\big(\frac{6}{1-\gamma} + 3\rho(\gamma)\big) \sum_i c_{ij} x_{A,ij}$, the total cost incurred for scenario A is at most

$$3\max\left(\tfrac{1}{\gamma}, \tfrac{2}{1-\gamma} + \rho(\gamma)\right)\left(\sum_i f_i^A y_{A,i} + \sum_{i,l} f_{i,l}^A y_{A,i,l} + \sum_{i,j\in A} c_{ij} x_{A,ij}\right).$$

Bounding the total cost. Adding the facility opening and service installation costs incurred in stage I and the expected total cost incurred in the stage II scenarios, we get that the overall cost is at most $\left(3\max\left(\frac{1}{\gamma}, \frac{2}{1-\gamma} + \rho(\gamma)\right) + \epsilon\right) \cdot OPT$. Setting $\gamma = 0.2641$ we get a ratio of at most $11.363 + \epsilon$.

Chapter 6

Connected Facility Location

6.1 Introduction

In this chapter we consider the *connected facility location* problem, that captures settings where the open facilities want to communicate with each other, or with a common central authority. For example, the facilities may represent caches in a distributed network that need to be able to communicate with each other to ensure consistency of data. In such cases, one desires a two-layered solution, where the demand points are first clustered around hubs (facilities) and the hubs are then interconnected to allow them to communicate with one another.

We model such settings by requiring that the open facilities be interconnected via a *Steiner tree*, i.e., a tree that connects all of the open facilities but may also include other non-facility nodes. A Steiner tree is less restrictive than a spanning tree, yet offers a simple and scalable network. This is the *connected facility location* (ConFL) problem. More precisely, we are given a graph $G = (V, E)$ with costs $\{c_e\}$ on the edges, a set of facilities $\mathcal{F} \subseteq V$, and a set of demand nodes or clients $\mathcal{D} \subseteq V$. Client j has d_j units of demand and facility i has an opening cost of f_i. We are also given a parameter $M \geq 1$. We want to open a set of facilities F, assign each demand to an open facility, and connect the open facilities by a Steiner tree T. If c_{ij} denotes the shortest path distance between nodes i and j in G (with respect to the costs

c_e), then assigning client j to facility $i(j)$ incurs a cost equal to $d_j c_{i(j)j}$. The cost of connecting facilities is simply the cost of the Steiner tree T scaled by a factor of M. Our objective is to minimize the total cost which is the sum of the costs of opening the facilities in F, the assignment costs of demands, and the cost of connecting the open facilities, that is, $\sum_{i \in F} f_i + \sum_{j \in \mathcal{D}} d_j c_{i(j)j} + M \sum_{e \in T} c_e$.

An application modeled nicely by the above framework is telecommunication network design [5, 60]. A common model of a telecommunication network consists of a *central core* and a set of *endnodes*. The core consists of a set of interconnected core nodes which have switching capability. Each core node also incurs some switch cost. Designing the network involves selecting a subset of core nodes, connecting the core nodes to each other and routing traffic from the endnodes to the selected core nodes. Here the clients are the endnodes of the network, and the facilities are the core nodes. The opening cost of a facility corresponds to the switch cost of the corresponding core node, while the parameter M reflects the more expensive cost of interconnecting the core nodes with high bandwidth links.

The Rent-or-Buy Problem. A useful special case of connected facility location arises if we allow a facility to be opened at any location and set all facility opening costs to 0, i.e., $\mathcal{F} = V$ and $f_i = 0$ for all i. This is known as the *rent-or-buy problem*. The cryptic name can be explained as follows. Suppose we *guess* a facility v, denoted as the *sink*, that is opened by the optimal solution. Since we can open facilities anywhere without incurring any cost, we will open facilities exactly at those locations where at least M clients are gathered and pay a cost of M per unit length to connect this open facility to the sink. This gives an alternate way to view the problem. We want to route demand from the clients to the sink by constructing a tree that connects the clients to v and installing sufficient capacity on the tree edges. We can either *rent* capacity on an edge by paying a cost per unit length proportional to the amount of capacity rented (which will equal the demand routed along the edge), or pay a one-time fixed cost of M (per unit length) and *buy* unlimited capacity. The objective is to find a tree with minimum cost.

6.1.1 Summary of Results

Our main results are a primal-dual 4.55-approximation algorithm for the rent-or-buy problem and an 8.55-approximation algorithm for the connected facility location problem. We present these algorithms in Section 6.4 and Section 6.5 respectively. In Section 6.6 we extend the algorithms to handle an edge-capacitated generalization of the problem. We now require clients to be connected to facilities via cables of capacity u that have a fixed cost σ per unit length. Multiple cables may be laid along an edge if necessary to route the demand along the edge. We give a constant-factor approximation algorithm for this generalization. In a subsequent chapter, we consider a variant of ConFL where we require that a feasible solution open at most k facilities. We show in Section 7.4 how to use the algorithm of Section 6.5 to obtain a constant-factor approximation algorithm for this problem.

6.1.2 Related Work

Connected Facility Location arises as a natural problem in various important applications. Krick, Räcke & Westermann [48] arrive at the problem by considering a data management/caching application. We have a set of users issuing read and write requests for data objects. Each object has to be stored in a memory module by paying a certain storage cost — an object may be replicated and stored in multiple locations. Given a placement of objects, a read request for an object issued at node j is served by the nearest location, $i(j)$, that has a copy of the object; a write request however needs to update *all* copies of the object. Krick et al. show that with a small loss in performance, this can be modeled by a single multicast tree connecting all locations that hold a copy of the object. A write request at j first sends a message to $i(j)$ which then initiates the update of all copies via the multicast tree. The goal is to find a placement of objects to memory modules that minimizes the sum of the storage, read and write request costs. This is exactly the connected facility location problem where the clients are the nodes issuing read/write requests, the facilities correspond to memory modules and the facility cost is the associated storage cost. The demand of a client is the number of requests issued by the node and the scaling parameter M corresponds to the total number of write requests for an object. Here the connectivity requirement is imposed by the need to maintain consistency of data.

Krick et al. gave a combinatorial constant-factor approximation algorithm for this problem with a large constant guarantee of the order of several hundred. Ravi & Selman [60] consider a closely related problem called the *traveling purchaser problem*, where the open facilities have to be connected by a cycle instead of a tree. They obtain a constant-factor guarantee by rounding the optimal solution to an exponential size LP using the ellipsoid method, which makes the algorithm very inefficient. Gupta, Kleinberg, Kumar, Rastogi & Yener [32] gave an algorithm with an approximation guarantee of 10.66 for ConFL and 9.001 for the rent-or-buy problem. Their algorithm is also based on rounding an exponential size LP as in [60]. Previously these were the best known guarantees. Subsequent to the publication [75] of the results presented in this chapter, Gupta, Kumar and Roughgarden [34] gave randomized approximation algorithms with ratios of 10.1 for ConFL and 3.55 the rent-or-buy problem.

The special case of ConFL in which $M = 1$ has been more widely studied in the computer science and operations research communities. Labbé, Laporte, Martín & González [51] gave a branch and bound procedure to exactly solve the cycle variant of the problem. Kim, Lowe, Tamir & Ward [45] gave a dynamic programming algorithm for the problem on a tree. Lee, Chiu & Ryan [52] considered the setting where the open facilities have to be connected by a spanning tree and gave a branch and bound algorithm. Khuller & Zhu [44] obtain a 5-approximation algorithm for this variant.

The rent-or-buy problem is an interesting special case that crops up in diverse scenarios. It abstracts a setting in which demand points need to be clustered around centers and the centers also have to be connected. Karger & Minkoff [43] introduced the *maybecast* problem which is a probabilistic version of the Steiner tree problem. Each terminal j is activated independently with probability p_j, and the goal is to find a Steiner tree connecting each terminal to the root v such that the expected cost of the subtree on the activated terminals is minimized. Gupta et. al. [32] arrived at the rent-or-buy problem by considering the problem of provisioning a virtual private network (VPN) where each VPN endpoint specifies only an upper bound on the amount of incoming and outgoing traffic. In both cases, it is shown that there is an optimal or near-optimal solution in which the demand points are clustered around hubs using shortest paths, and the hubs are connected to the root by a Steiner tree. Thus, both these problems reduce to the rent-or-buy problem. Karger & Minkoff [43] gave a combinatorial algorithm with a constant approximation ratio of around 20. Kumar, Rastogi, Silbershatz & Yener [50] implemented a heuristic for the problem and used it to construct VPN trees. They report that the algorithm outperforms standard heuristics over a wide range of parameter values, but do not give any worst-case performance guarantees.

The single-sink buy-at-bulk problem is a generalization of the rent-or-buy problem where one seeks a minimum-cost way of routing all demand to the sink by installing capacity on the edges, and the per-unit length capacity installation cost is a concave, increasing function of the capacity. Guha, Meyerson and Munagala [30] gave a constant-factor approximation algorithm for this problem. The constant was improved by Talwar [76] and subsequently by Gupta, Kumar & Roughgarden [34] to 73.

An orthogonal extension of the rent-or-buy problem is the multicommodity rent-or-buy problem where instead of a common sink, there are multiple commodities represented by source-sink pairs and the goal is to install capacity on the edges so that one can simultaneously route the traffic between the source and sink of every commodity. As in the single sink case, we may either rent or buy capacity on the edges. Kumar, Gupta & Roughgarden [49] gave the first constant-factor approximation algorithm for the multicommodity rent-or-buy problem. Very recently, Gupta, Kumar, Pál & Roughgarden [33] gave an algorithm with an improved ratio of 12.

6.2 A Linear Programming Relaxation

In what follows, i will be used to index facilities, j to index the clients and e to index the edges in G. We use the terms client and demand point interchangeably.

We assume that we know one facility v that is opened and hence belongs to the Steiner tree constructed by the optimal solution (since we can try all $|\mathcal{F}|$ different possibilities for v). We can now write the following integer program (IP) for ConFL.

$$\begin{aligned}
\min & \sum_i f_i y_i + \sum_j d_j \sum_i c_{ij} x_{ij} + M \sum_e c_e z_e & & \text{(IP)} \\
\text{s.t.} \quad & \sum_i x_{ij} \geq 1 & \text{for all } j, & \\
& x_{ij} \leq y_i & \text{for all } i, j, & \\
& y_v = 1 & & \\
& \sum_{i \in S} x_{ij} \leq \sum_{e \in \delta(S)} z_e & \text{for all } S \subseteq V, v \notin S,\ j \in \mathcal{D}, & \quad (1) \\
& x_{ij}, y_i, z_e \in \{0, 1\} & \text{for all } i, j, e. & \quad (2)
\end{aligned}$$

Here y_i indicates if facility i is open, x_{ij} indicates if client j is connected to facility i and z_e indicates if edge e is included in the Steiner tree. The first and second constraints say that each client must be assigned to an open facility, and constraint (1) encodes the requirement that the open facilities should be connected to v. Consider any set $S \subseteq V$ that does not contain v. If there is some client j that is getting served by some (open) facility in S, then to connect this facility to the root there must be some outgoing edge from this set S that is included in the Steiner tree, and this is enforced by (1). Relaxing the integrality constraints (2) to $x_{ij}, y_i, z_e \geq 0$ gives us a linear program (LP). For simplicity, in the sequel we assume that all demands d_j are equal to 1. We show how to get rid of this assumption in Section 6.7. The quantity $\mathcal{O}^*$ will always denote the cost of an optimal *integer* solution, i.e., the value of the integer program (IP). We use OPT to denote the value of the optimal solution to the linear program (LP) which may be obtained by a fractional solution.

6.3 The High Level Idea

Let us first give some intuition. Observe that connected facility location has elements of both the facility location problem and the Steiner tree problem. Without the connectivity requirement, the problem is just uncapacitated facility location, and if we know which facilities to open then we can simply assign each demand to the closest open facility and connect the open facilities by a Steiner tree.

Consider first the naive algorithm where we decide which facilities to open using a good algorithm for uncapacitated facility location, and then connect the open facilities by a Steiner tree. However, this fails immediately. For example, in the rent-or-buy problem, we would just open a facility at each demand point, and so connecting the open facilities might incur a huge cost. Thus there is an implicit cost (besides the facility opening cost) associated with opening a facility due to the connectivity requirement: once we open a facility, we have to connect it to the other open facilities by buying edges at a cost of M per unit length. Since the rental cost is less than the buying cost when there are fewer than M demand points, it seems reasonable, at least in the rent-or-buy problem where we can open a facility anywhere without incurring any cost, to open a facility only if there are at least M demand points using that facility. This is exactly what we do. Our strategy will be to open facilities and assign clients to facilities paying a small cost relative to the optimal cost, ensuring that we *cluster at least M demand points around each open facility*, and then connect the open facilities. We make the above intuition precise in Lemma 6.4.1 and show that indeed the added clustering requirement allows us to bound the cost of connecting the facilities relative to the optimal cost and the assignment cost incurred by our algorithm.

The algorithm consists of a *facility location* phase and a *Steiner phase*. The ConFL dual program can be interpreted as consisting of a part resembling the dual of the facility location problem and a part corresponding to the dual of the Steiner tree problem. In the facility location phase, we open facilities and assign clients to facilities satisfying the demand lower bound of M; in the Steiner phase, we simply connect the open facilities by a Steiner tree. In the facility location phase, we simultaneously construct an integer primal solution and a feasible dual solution and are able to meet the demand lower bound by charging some of the cost incurred to the Steiner tree portion of the dual solution. Thus we exploit the fact that any ConFL solution also needs to connect the facilities it opens. This is the key point where we depart from previous approaches [43, 31], in which the clustering requirement is only approximately satisfied using a bicriteria approximation algorithm for the Lower Bounded Facility Location (LBFL) problem, which is a facility location problem where each open facility is required to serve a certain threshold number of clients. The disadvantage of this approach is that the LBFL instance is solved by a black box that (a) makes no use of the fact that the need to cluster demand points is imposed by the connectivity requirement, and (b) gives an inferior performance guarantee because it is only able to approximately meet the clustering requirement.

6.4 The Rent-or-Buy Problem

We first consider the rent-or-buy case where a facility can be opened at any vertex of V and all facility opening costs are 0, i.e., $\mathcal{F} = V$ and $f_i = 0$ for all i.

The linear program (LP) now simplifies to:

$$\begin{aligned}
\min \quad & \sum_j \sum_i c_{ij} x_{ij} + M \sum_e c_e z_e & & \text{(RB-P)} \\
\text{s.t.} \quad & \sum_i x_{ij} \geq 1 & & \text{for all } j, \\
& \sum_{i \in S} x_{ij} \leq \sum_{e \in \delta(S)} z_e & & \text{for all } S \subseteq V, v \notin S,\ j \in \mathcal{D}, \\
& x_{ij}, z_e \geq 0 & & \text{for all } i, j, e.
\end{aligned}$$

The dual of this linear program is:

$$
\begin{aligned}
\max \quad & \sum_j \alpha_j & & \text{(RB-D)}\\
\text{s.t.} \quad & \alpha_j \le c_{ij} + \sum_{\substack{S\subseteq V: i\in S\\ v\notin S}} \theta_{S,j} & \text{for all } i \neq v,\ j\in\mathcal{D}, & \quad (3)\\
& \alpha_j \le c_{vj} & \text{for all } j, & \quad (4)\\
\sum_j \sum_{\substack{S\subseteq V: e\in\delta(S)\\ v\notin S}} & \theta_{S,j} \le M c_e & \text{for all } e, & \quad (5)\\
& \alpha_j, \theta_{S,j} \ge 0 & \text{for all } j, S. &
\end{aligned}
$$

Intuitively, α_j is the *payment* that demand j is willing to make towards constructing a feasible primal solution. Constraint (3) says that a part of the payment α_j goes towards assigning j to a facility i. The remaining part goes towards constructing the part of the Steiner tree that joins i to v. The algorithm runs in two phases. First we cluster demands in groups of M; once we have this, we run the second phase where we build the Steiner tree.

We begin with a simplifying assumption. We assume that a facility can be opened *anywhere along an edge*. We collectively refer to vertices in V and internal points on an edge as *locations*. We reserve the term facility for a vertex in $\mathcal{F}$. We may assume that for an edge $e = (u, w)$, the value of c_e is equal to c_{uw} which is shortest path distance from u to w (otherwise we may simply set $c_e = c_{uw}$ without changing any shortest path distances). We extend the metric c to a metric on locations by considering e to be composed of infinitely many edges of infinitesimal length. So for points p on an edge e, the distance c_{up} varies continuously and monotonically from 0 to $c_e = c_{uw}$ as we go from u to w, and $c_{wp} = c_e - c_{up}$. For any other vertex $r \neq u, w$, we set $c_{rp} = \min(c_{ru} + c_{up}, c_{rw} + c_{wp})$. Finally for any two points p, q on edges $e_1 = (u, w), e_2$ respectively, $c_{pq} = \min(c_{uq} + c_{up}, c_{wq} + c_{wp})$.

Phase 1: The Facility Location Phase

We build a (partial) integer primal solution and a feasible dual solution simultaneously. The primal-dual process is conceptually quite simple: each demand j keeps raising its dual variable, α_j, till it gets assigned to a location at which M clients are clustered. All other variables simply respond to this change trying to maintain feasibility or complementary slackness.

We have a notion of time, t. Initially $t = 0$ and all dual variables are initialized to 0. As time increases, we raise the dual variables α_j at unit rate (i.e., $\alpha_j = t$ at time t). We shall also *tentatively open* some locations. At $t = 0$, v is tentatively open and all other locations are closed. At some point of time, we say that demand j is *tight* with a location i if $\alpha_j \geq c_{ij}$. Let S_j be the set of vertices with which j is tight at some point of time. When we raise α_j, we also raise $\theta_{S_j,j}$ at the same rate. This will ensure feasibility of constraints (3). So, it is enough to describe how to raise the dual variables α_j.

Clients can be in two states: *frozen* or *unfrozen*. When a client j gets frozen, we stop raising its dual variable α_j. So if client j is unfrozen at time t, $\alpha_j = t$. After j is frozen, it does not become tight with any new location, i.e., a location not in S_j. Initially, all clients are unfrozen. We start raising the α_j of all demand points at unit rate until one of the following events happens (if several events happen simultaneously, consider them in any order):

1. j becomes tight with a tentatively open location i: j becomes frozen.

2. There is a closed location i with which at least M demand points are tight: tentatively open i. All of the demand points tight with i become frozen.

We now raise the α_j of unfrozen clients only. We continue this process until all clients become frozen. Figure 6.1 shows a sample run of the algorithm with $M = 2$ and 5 demand points. Note that although there is a continuum of points along an edge, to implement the above process we only need to know the time when the next event will take place. This can be obtained by keeping track of, for every edge and every demand point j, the portion of the edge that is tight with j.

Now we decide which locations to open. Let L be the set of tentatively open

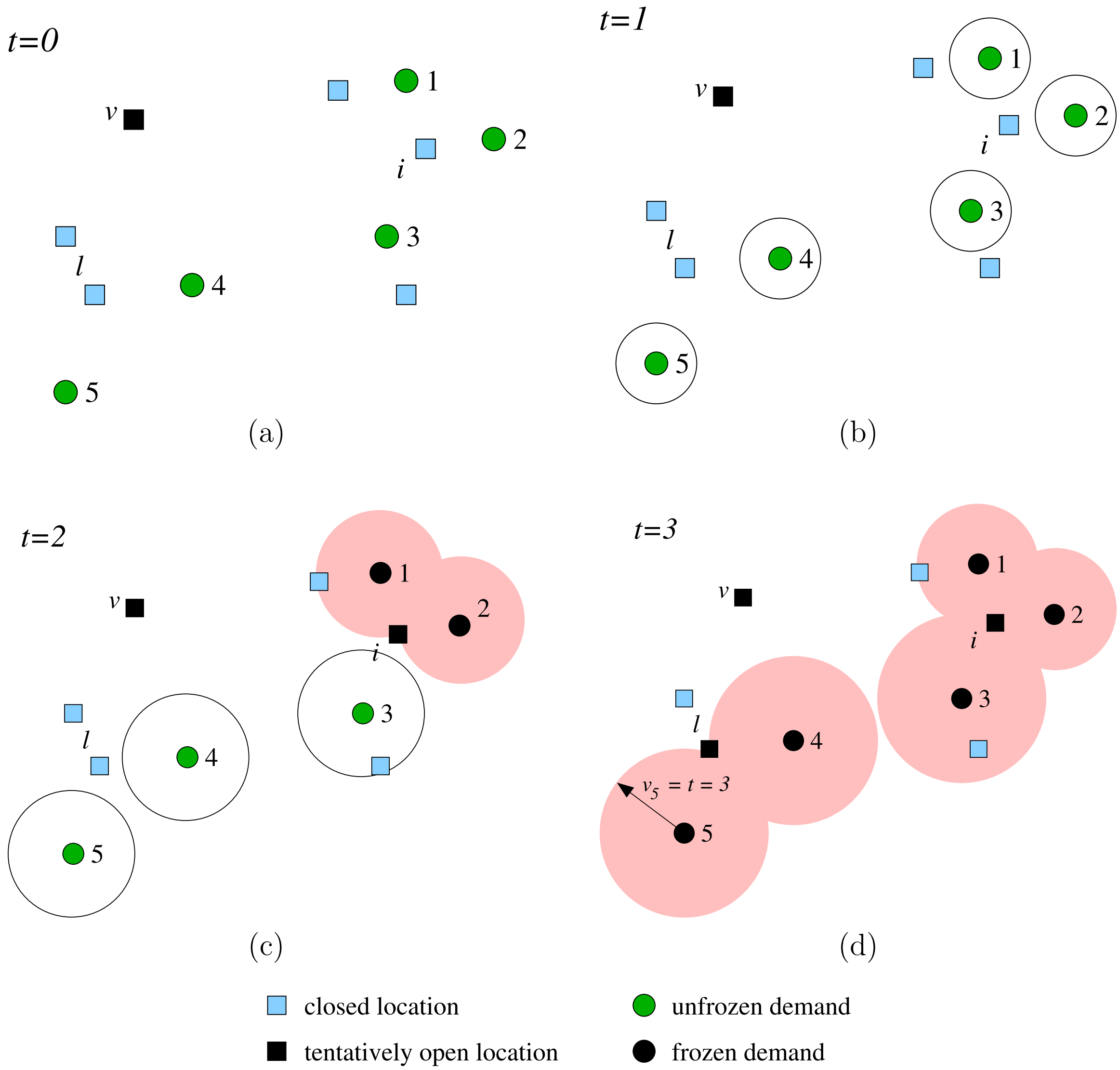

Figure 6.1: A sample run with $M = 2$. (a) The initial state, (b) $t = 1$, (c) i becomes tight with clients 1 and 2; i is tentatively opened and 1, 2 become frozen, (d) The final solution. Demand point 3 reaches i and gets frozen; l becomes tight with clients 4 and 5 and is tentatively opened, causing clients 4 and 5 to freeze.

locations. We say that $i, i' \in L$ are dependent if there is demand point j which is tight with both these locations. We say that a set of locations is *independent* if no two locations in this set are dependent. We find a maximal independent set L' of locations in L as follows: arrange the locations in L in the order they were tentatively opened. Consider the locations in this order and add a location to L' if no dependent location is already present in L'. We open the locations in L'. Observe that $v \in L'$.

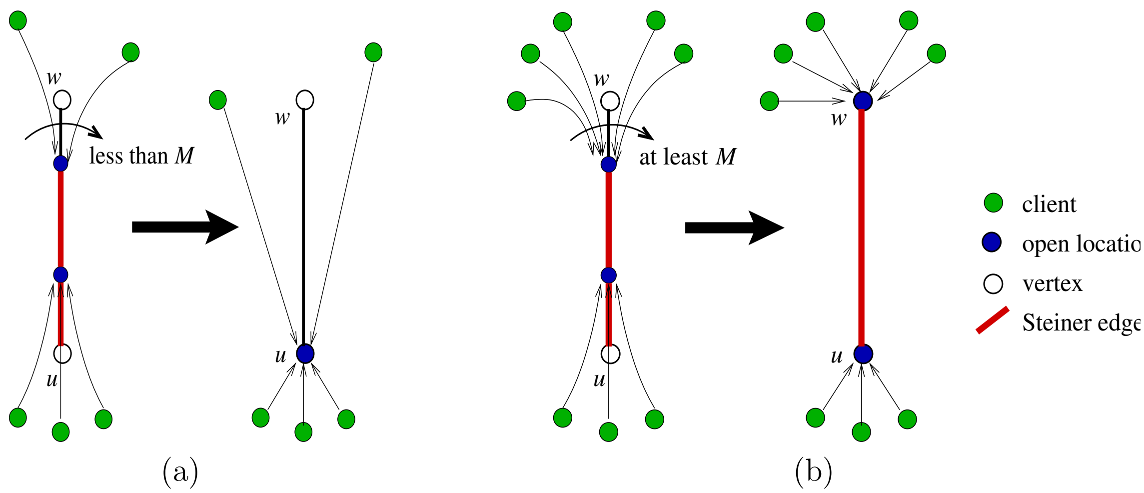

Figure 6.2: Moving intermediate facilities to vertices. (a) $|D_w| \leq M$, (b) $|D_w| \geq M$

We assign a demand point j to an open location as follows. If j is tight with some $i \in L'$, assign j to i. Otherwise let i be the location in L that caused j to become frozen. So j is tight with i. There must be some previously opened location $i' \in L'$ such that i and i' are dependent. We assign j to i'. Let $\sigma(j)$ denote the location to which j is assigned.

Phase 2: The Steiner Phase

First we augment the graph G to include edges incident on open non-vertex locations. Let $\{i_1, \ldots, i_k\}$ be the open locations on edge $e = (u, w)$ ordered by increasing distance from u, with $i_1 \neq u, i_k \neq w$. We add edges $(u, i_1), (i_1, i_2), \ldots, (i_{k-1}, i_k)$, (i_k, w) to G. We now build a Steiner tree with L' as the set of terminals using a ρ_{ST}-approximation algorithm for the Steiner tree problem. It is a well known fact that a minimum cost Steiner tree can be approximated to within a factor of 2 by a minimum spanning tree, therefore we assume from now on that $\rho_{ST} \leq 2$.

The solution obtained may be infeasible since a non-vertex location may be opened as a facility. Consider an edge $e = (u, w)$ whose internal points contain open locations. Let D_e be the set of demand points which are assigned to such locations. Let $D_u \subseteq D_e$ be the set of demand points that reach their assigned location on e via u, i.e., $c_{\sigma(j)j} = c_{uj} + c_{\sigma(j)u}$ for $j \in D_u$; D_w is defined similarly. The Steiner tree T must contain at least one of u or w. If both $u, w \in T$, we assign clients in D_u to u and clients in D_w to w without increasing the cost. Suppose $u \in T, w \notin T$. Let l be the open location which is farthest from u (and hence, nearest to w) on e. We assign all clients in D_u to u. If $|D_w| < M$, we assign clients in D_w to u and remove edges in T that lie along e (see Fig. 6.2a). This only decreases the cost, because considering the net cost due to edge e, earlier the cost incurred was at least $Mc_{ul} + |D_w|c_{wl} > |D_w|c_{uw}$ to buy Steiner edges along e connecting location l to u, and to assign clients in D_w to an open location on e, whereas now we pay a cost of $|D_w|c_{u_w}$ to assign the clients in D_w to u. If $|D_w| \geq M$, we reassign all clients in D_w to w and add all of e to T (see Fig. 6.2b). It is easy to see that the total cost only decreases and that T remains a Steiner tree on the open locations. Thus, we can shift all open locations to vertices of G without increasing the total cost.

6.4.1 Analysis

Let C^*, S^* denote the assignment cost and Steiner tree cost of an optimal (integer) solution (that opens v). Recall that $\mathcal{O}^* = C^* + S^*$ is the optimal cost. We will sometimes abuse notation and use $\mathcal{O}^*$ to also denote an optimal solution. We show that the solution returned has cost at most $(3 + \rho_{ST}) \cdot \mathcal{O}^*$. Let $\left(\alpha^{(1)}, \theta^{(1)}\right)$ be the value of the dual variables at the end of Phase 1. We start by making the intuition of Section 6.3 precise.

Lemma 6.4.1 *Let A be a set of locations. Let D_l be a set of clients associated with each location $l \in A$ such that $|D_l| \geq M$ and the sets D_l are all disjoint, i.e., $D_l \cap D_{l'} = \emptyset$ for $l \neq l'$. Then the cost of an optimal Steiner tree connecting the locations in A is at most $S^* + C^* + \sum_{l \in A \setminus \{v\}} \sum_{j \in D_l} c_{ij}$.*

Proof : We will show that the optimal tree can be extended to yield a Steiner tree on the locations in A of cost no greater than the claimed cost, the optimal tree spanning A can only cost less. Note that the optimal tree contains v. We obtain such a tree by connecting each location $l \in A \setminus \{v\}$ to the optimal tree with cost S^* via the

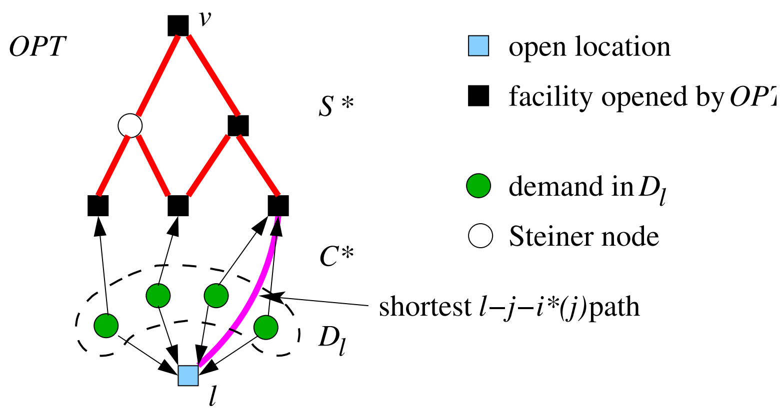

Figure 6.3: Extending the optimal tree to a Steiner tree on the open locations.

shortest $l - j - i^*(j)$ path for $j \in D_l$, where $i^*(j)$ is the facility to which j is assigned in $\mathcal{O}^*$. (see Fig. 6.3). For any $l \in A \setminus \{v\}$ the cost of adding the connecting edges is at most M(length of the shortest $l - j - i^*(j)$ path for $j \in D_l$) $\leq \sum_{j \in D_l}(c_{i^*(j)j} + c_{lj})$ since $|D_l| \geq M$. Summing over all such locations l, the tree obtained has cost at most $S^* + C^* + \sum_{l \in A \setminus \{v\}} \sum_{j \in D_l} c_{lj}$. ∎

Observe that the set of open locations L' is a set that with all the properties stated in Lemma 6.4.1. Let $D' = \bigcup_{i \in L' \setminus \{v\}} D_i$. Recall that $\sigma(j)$ is the location to which j is assigned. Note that the algorithm assigns every demand point $j \in D_i$ to i. Using the above lemma, the cost of the Steiner tree constructed on L' by a ρ_{ST}-approximation algorithm is at most $\rho_{ST} \cdot \mathcal{O}^* + 2\sum_{j \in D'} c_{\sigma(j)j}$. We will show that $3\sum_{j \in D'} c_{\sigma(j)j} + \sum_{j \notin D'} c_{\sigma(j)j} \leq 3 \cdot \mathcal{O}^*$. Since the assignment cost incurred is $\sum_j c_{\sigma(j)j}$, we get that the total cost is at most $(3 + \rho_{ST}) \cdot \mathcal{O}^*$.

Lemma 6.4.2 *The dual solution* $\left(\alpha^{(1)}, \theta^{(1)}\right)$ *is feasible.*

Proof : It is easy to see that (3) is satisfied. Indeed, once j gets tight with i, α_j and $\sum_{S: i \in S, v \notin S} \theta_{S,j}$ are raised at the same rate. Similarly, (4) is satisfied.

Now consider an edge $e = (u, w)$. Let $l(j)$ be the contribution of j to the left hand side of (5) for this edge, i.e., $l(j) = \sum_{S: e \in \delta(S), v \notin S} \theta^{(1)}_{S,j}$. Suppose $c_{ju} \leq c_{jw}$. So, j becomes tight with u before it becomes tight with w. Consider a point p on the edge (u, w) at distance x from u. If p were the last point on this edge with which j

became tight (before it became frozen), then $l(j) \le x$. Define $f(j,x)$ as 1 if j is tight with p and j was not frozen at the time at which it became tight with p, otherwise $f(j,x)$ is 0. So, we can write $l(j) \le \int_0^{c_e} f(j,x)dx$. Interchanging the summation and the integral in (5), we get

$$\sum_j \sum_{S \subseteq V: e \in \delta(S), v \notin S} \theta_{S,j}^{(1)} \le \sum_j \int_0^{c_e} f(j,x)dx = \int_0^{c_e} \sum_j f(j,x)dx$$

Now, we argue that for any x, $\sum_j f(j,x) \le M$. Otherwise, we have more than M demand points that are tight with a point such that none of these demand points are frozen — a contradiction. So $\int_0^{c_e} \sum_j f(j,x)dx$ is at most Mc_e, which proves the lemma. ■

Lemma 6.4.3 *The assignment cost of client j is at most $\alpha_j^{(1)}$ if $j \in D'$, and at most $3\alpha_j^{(1)}$, otherwise.*

Proof : If $j \in D'$, the claim clearly holds since j is tight with location $\sigma(j) \in L'$. Otherwise let j be assigned to i. Let i' be the tentatively open facility that caused j to become frozen. It must be the case that i and i' are dependent. So there is a client k that is tight with both i and i'. Let $t_{i'}$ be the time at which i' was tentatively opened. Define t_i similarly. It is clear that $\alpha_j^{(1)} \ge t_{i'}$.

Now, $c_{ij} \le c_{ik} + c_{ki'} + c_{i'j} \le 2\alpha_k^{(1)} + \alpha_j^{(1)}$. Also, $\alpha_k^{(1)} \le t_{i'}$. Otherwise, at time $t = \alpha_k^{(1)}$, k is tight with both i and i'. Suppose it becomes tight with i first (the other case is analogous). If i is tentatively open at this time, then k will freeze and so it will never become tight with i'. Therefore, i cannot be tentatively open at this time. But then, k must freeze by the time i becomes tentatively open, i.e., $\alpha_k^{(1)} \le t_i \le t_{i'}$. So, $\alpha_k^{(1)} \le t_{i'} \le \alpha_j^{(1)}$. This implies that $c_{ij} \le 3\alpha_j^{(1)}$. ■

Taking $\rho_{ST} = 1.55$ [65], we obtain the following.

Theorem 6.4.4 *The algorithm produces a solution of cost at most $4.55 \cdot \mathcal{O}^*$.*

Proof : The connection cost is bounded by $\rho_{ST} \cdot \mathcal{O}^* + 2\sum_{j \in D'} c_{\sigma(j)j}$. Adding this to the assignment cost $\sum_j c_{\sigma(j)j}$ and using Lemma 6.4.3 proves the result. ■

Bounding the Integrality Gap

Instead of the 1.55-approximation algorithm, if we run the primal-dual Steiner tree algorithm due to [2, 26] with $\rho_{ST} = 2$ in Phase 2, we get a solution of cost at most $5 \cdot OPT$. Recall that OPT is the cost of a (possibly) fractional optimum solution to (RB-P). This shows that the integrality gap of this LP relaxation is at most 5.

We will simulate the algorithm of [2, 26] for the Steiner tree problem with root v and terminal set $L' \setminus \{v\}$ by raising some dual variables in Phase 2. First, set $\alpha_j = 0$ for all j. We raise the α_j value of clients in D' only. The tree T that we construct is empty to begin with. Initially, the minimal violated sets (MVS) are the singleton sets $\{i\}$ for $i \in L' - \{v\}$. For a set S, define $D_S = \bigcup_{i \in S \cap L'} D_i$. For each MVS S, $j \in D_S$, we raise α_j at rate $1/|D_S|$. We also raise $\theta_{S,j}$, at the same rate. This ensures that $\sum_j \theta_{S,j}$ grows at rate 1 for any MVS S. Note that we are *not* ensuring feasibility of constraints (3), (4).

We raise the dual variables till inequality (5) holds with equality for some edge e; we say that edge e goes tight when this happens. We add e to T and update the minimal violated sets. This process continues till there is no violated set, i.e., we have only one component (so v is in this component). Now we consider edges of T in the reverse order they were added and remove any redundant edges. This is our final solution. Let $\left(\alpha^{(2)}, \theta^{(2)}\right)$ be the dual solution constructed by this process.

Lemma 6.4.5 $\mathsf{cost}(T) \leq 2 \cdot \sum_{j \in D'} \alpha_j^{(2)}$.

Proof : At any point of time, define the variable θ_S, where S is a minimal violated set, as $\sum_j \theta_{S,j}$. Since θ_S grows at rate 1, Phase 2 simulates the primal-dual algorithm for the rooted Steiner tree problem with v as the root. So, the cost of the tree is bounded by $2 \cdot \sum_S \theta_S$ (see [26, 2, 80]), where the sum is over all subsets of vertices S. But $\sum_S \theta_S = \sum_{j \in D'} \alpha_j^{(2)}$. ■

Lemma 6.4.6 *For any client j and $i \neq v$, $\alpha_j^{(2)} \leq c_{\sigma(j)j} + c_{ij} + \sum_{S \subseteq V : i \in S, v \notin S} \theta_{S,j}^{(2)}$. Further, $\alpha_j^{(2)} \leq c_{\sigma(j)j} + c_{vj}$.*

Proof : If $j \notin D'$, $\alpha_j^{(2)} = \theta_{S,j}^{(2)} = 0$ and the inequalities above hold. So fix a demand point $j \in D'$ and facility i, $i \neq v$. During the execution of Phase 2, let S_t be the component to which j contributes at time t. Consider the earliest time t' for which $i \in S_{t'}$. After this time, both the left hand side and right hand side of (3) increase at the same rate, so we only need to bound the increase in α_j by time t'. Let $l = \sigma(j)$. Since we are raising α_j, it must be the case that $j \in D_l$ and so, $c_{lj} \leq \alpha_j^{(1)}$. We claim that $t' \leq Mc_{li}$. This is true since S_t always contains l, and by time $t = Mc_{li}$ all of the edges along the shortest path between l and i would have grown tight. Since α_j increases at a rate of at most $1/M$, the increase in α_j by time t' is at most $\frac{Mc_{li}}{M} \leq c_{lj} + c_{ij}$. This proves the first inequality. The second inequality holds because we stop increasing α_j once v lies in S_t. ■

Theorem 6.4.7 *The above algorithm produces a solution of cost at most* $5 \cdot OPT$.

Proof : Define $\alpha'_j = \max(\alpha_j^{(2)} - c_{\sigma(j)j}, 0)$. It is clear that the $\theta_{S,j}^{(2)}$ values satisfy (5), by the above lemma, $\left(\alpha', \theta^{(2)}\right)$ is a feasible dual solution. By Lemma 6.4.5, $\mathsf{cost}(T) \leq 2\sum_j \alpha_j^{(2)} \leq 2\sum_j \alpha'_j + 2\sum_{j \in D'} c_{\sigma(j)j} \leq 2 \cdot OPT + 2\sum_{j \in D'} \alpha_j^{(1)}$. Combining this with the assignment cost and using Lemma 6.4.3, we see that the cost of our solution is at most $5 \cdot OPT$. ■

6.5 The General Case

We now consider the case where $\mathcal{F}$ need not be V and facility i has an opening cost $f_i \geq 0$. Since facilities may only be opened at specific locations, it is possible that an edge is used both to route demand from a client to a facility, and also as an edge in the Steiner tree to connect facilities. We call the former type of edge a *facility location edge* and the latter a *Steiner edge*. For convenience, we assume that $f_v = 0$. Clearly, this does not affect the approximation ratio of the algorithm. As usual, i indexes the facilities in $\mathcal{F}$ and j indexes the clients in $\mathcal{D}$. The primal and dual LPs are:

$$
\begin{aligned}
\min \ & \sum_{i \neq v} f_i y_i + \sum_j \sum_i c_{ij} x_{ij} + M \sum_e c_e z_e && && \text{(ConFL-P)} \\
\text{s.t.} \ & \sum_i x_{ij} \geq 1 && \text{for all } j, \\
& x_{ij} \leq y_i && \text{for all } i \neq v,\ j \in \mathcal{D}, \\
& x_{vj} \leq 1 \\
& \sum_{i \in S} x_{ij} \leq \sum_{e \in \delta(S)} z_e && \text{for all } S \subseteq V, v \notin S,\ j \in \mathcal{D}, \\
& x_{ij}, y_i, z_e \geq 0 && \text{for all } i, j, e.
\end{aligned}
$$

$$
\begin{aligned}
\max \ & \sum_j \alpha_j - \sum_j \beta_{vj} && && \text{(ConFL-D)} \\
\text{s.t.} \ & \alpha_j \leq c_{ij} + \beta_{ij} + \sum_{\substack{S \subseteq V : i \in S \\ v \notin S}} \theta_{S,j} && \text{for all } i \neq v,\ j \in \mathcal{D}, && (6) \\
& \alpha_j \leq c_{vj} + \beta_{vj} && \text{for all } j, \\
& \sum_j \beta_{ij} \leq f_i && \text{for all } i \neq v, && (7) \\
& \sum_j \sum_{\substack{S \subseteq V : e \in \delta(S) \\ v \notin S}} \theta_{S,j} \leq M c_e && \text{for all } e, \\
& \alpha_j, \beta_{ij}, \theta_{S,j} \geq 0 && \text{for all } i, j, S.
\end{aligned}
$$

An Overview of the Modifications

The basic idea is similar: we still want to gather at least M clients at every facility that we open so that the cost of connecting this facility to other open facilities by Steiner edges can be amortized against the gathered demand. However, whereas earlier where we could tentatively open any location with which M clients are tight, we cannot do that here since the set of candidate facility locations $\mathcal{F}$ may be a very small subset of V. Also, we need to pay a facility opening cost before we can open a facility.

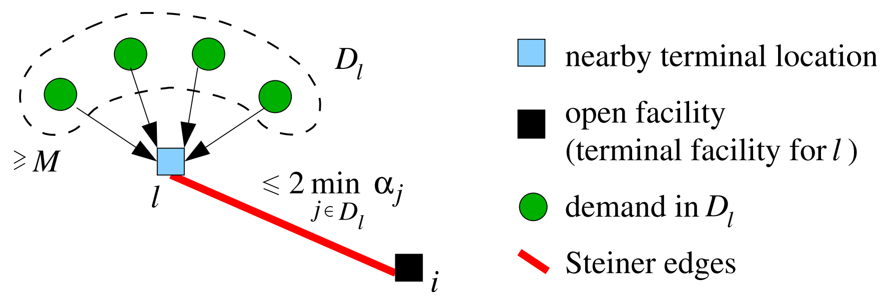

Figure 6.4: Steiner edges connecting an open facility to the "nearby" point where M clients are gathered.

Consequently, we will not quite be able to meet the demand requirement of M at every facility we open, but we will ensure that for every open facility, there are M demand points gathered at a point "near" the facility (see Fig. 6.4). In Phase 1 of the algorithm, we will open facilities and assign each client to an open facility. Additionally, for each open facility we will connect it to the point near it at which M demand points are gathered using Steiner edges, and we will argue that we can pay for the cost of buying this path by the combined dual of the gathered clients. These components act as the terminals upon which the Steiner tree is constructed in Phase 2.

Phase 1: The Facility Location Phase

Most of the changes are in this phase. A location still refers to a vertex in V or a point along an edge. We will only open facilities at locations in $\mathcal{F} \subseteq V$. Initially all dual variables are 0 and only facility v is tentatively open. We also declare location v to be a *terminal location*. Recall that client j is said to be tight with location i if $\alpha_j \geq c_{ij}$. As in the previous section, we will grow each dual variable α_j till j becomes tight with a location, referred to as a terminal location, with which at least M clients are tight. Once this happens however, we do not freeze j yet. Since we have to assign client j to an open facility and also have to pay for opening facilities, we continue to increase α_j till j becomes tight with a tentatively open facility. While doing so, if j becomes tight with a facility that is not yet open, then it starts contributing toward the facility opening cost of this facility.

To describe the primal-dual process in detail, we define a few additional concepts. As before, a client can be frozen or unfrozen. Further, a client j could either be *free* or be a *slave*. At $t = 0$, each client j is free and unfrozen. We say that client j is *bound* to a location l if j is tight with l *and was free when it became tight with* l. Define the weight of a location l as the number of clients that are *bound* to l. We say that a facility i has been *paid* for if $\sum_j \beta_{ij} = f_i$.

At any point in time, define S_j to be the set of vertices with which client j is tight. When j becomes tight with a facility i, we have two options — we can raise β_{ij} or we can raise $\theta_{S_j,j}$. We raise $\theta_{S_j,j}$ at the same rate and continue this till j becomes tight with a terminal location, that is, a location that has at least M clients *bound* to it[1]. At this point we say that j becomes a *slave* — it is no longer free. Similarly, when j becomes tight with a location l that is *not* a facility, we may or may not raise $\theta_{S_j,j}$ (we have this option since constraint (6) applies only to facilities i). We first increase $\theta_{S_j,j}$ until j becomes tight with a terminal location and is declared to be a slave. After this point, we start raising β_{ij} for each facility $i \in S_j$, and do not raise $\theta_{S_j,j}$ any more. More precisely, we raise the α_j of *every* unfrozen client, be it free or a slave, at unit rate until one of the following events happens:

1. The weight of some location l becomes at least M: declare l to be a terminal location. If j is free and tight with l, it now becomes a *slave*. From this point on we raise only β_{ij} for facilities i in S_j (there may be none if the current $\alpha_j < \min_i c_{ij}$) as described above.

2. A free j becomes tight with a terminal location l: j becomes a slave. If $l = v$, connect j to l and freeze j. Otherwise, we stop raising $\theta_{S_j,j}$ and raise β_{ij} for facilities i in S_j.

3. A facility i gets paid for, i.e., $\sum_j \beta_{ij} = f_i$: tentatively open i. If an (unfrozen) slave client j is tight with i, connect it to i and freeze j.

[1]The reverse — raising β_{ij} first until j gets connected to a facility and then increasing $\theta_{S_j,j}$ also works — but we raise the dual variables in this fashion in order to prove a guarantee for the connected k-median problem.

4. A slave client j becomes tight with a tentatively open facility i: connect j to i, freeze j.

We continue this process until all j become frozen. Frozen clients do not participate in any new events. Note that every client j starts out as free and unfrozen, then becomes a slave by becoming tight with a terminal location, and finally gets frozen by getting connected to exactly one tentatively open facility. Let $\left(\alpha^{(1)}, \beta^{(1)}, \theta^{(1)}\right)$ be the dual solution obtained. Clearly, $\beta_{vj}^{(1)} = 0$ for all j.

Let L be the set of all terminal locations. Let t_l be the time at which l was declared a terminal location. Let D_l be the set of clients *bound to* l. We associate a *terminal facility* with each $l \in L$. Consider the client in D_l with smallest $\alpha_j^{(1)}$ value. We call this the *representative client* of location l. Let i be the tentatively open facility to which the representative client is connected. We denote i as the terminal facility corresponding to l. Let the terminal facility corresponding to v be v itself. Let F be the set of all terminal facilities. We will only open facilities from the set F.

We will pick a subset of terminal locations and open the terminal facilities corresponding to these locations. For each location l that we pick, we will connect l to its terminal facility i by buying Steiner edges along a shortest $l - i$ path (see Fig. 6.4). We choose the subset of terminal locations carefully so as to ensure that a client j does not pay for opening or connecting more than one facility. Say that two facilities i, i' are dependent if either (1) there is a client j with both $\beta_{ij}^{(1)}, \beta_{i'j}^{(1)} > 0$, or (2) there is a location $l \in L$ and a client j such that i is the terminal facility corresponding to l, j is in D_l, and $\beta_{i'j}^{(1)} > 0$. The second condition is added to ensure that j does not pay for both opening i' and for connecting i to l via Steiner edges. We also have a notion of dependence between *locations* in L. We say that *locations* l and l' in L are dependent if either there is a demand point that is bound to both l and l', or the terminal facilities corresponding to l and l' are dependent. Now we greedily select a maximal independent set of locations by looking at locations in a particular order. With each $l \in L$ we associate a value ϕ_l. Let j be the representative client of l. Define $\phi_l = \max(\alpha_j^{(1)}, t_l)$, set $\phi_v = 0$. We look at the locations in L in increasing order of

ϕ_l, and select a maximal independent subset L' of L as before. Let F' be the set of terminal facilities corresponding to locations in L'. We open all the facilities in F'. Note that $v \in F'$.

We associate a terminal location $\sigma(j)$ with each demand point j. If $j \in D_l$ where $l \in L'$, set $\sigma(j) = l$. Note that $\sigma(j)$ is well defined due to our independent set construction. Otherwise let l be the location in L that caused j to become a slave. There is a previously selected location $l' \in L'$ such that l and l' are dependent. Set $\sigma(j) = l'$. Client j is assigned to a facility $i(j) \in F'$ as follows: if there is a facility $i \in F'$ such that $\beta_{ij}^{(1)} > 0$, assign j to i. Otherwise assign j to the terminal facility corresponding to $\sigma(j)$.

Let $D' = \bigcup_{l \in L' - \{v\}} D_l$. We now form some components by adding edges connecting each l in L' to its terminal facility via a shortest path. Break any cycles by deleting edges. Let T' be the set of edges added.

Phase 2: The Steiner Phase

This phase is similar to that of the previous section. G is augmented as before to include edges incident on locations $l \in L'$. We construct a Steiner tree T'' connecting all the components of T' using a ρ_{ST}-approximation algorithm for the Steiner tree problem, that is, we contract each component of T' and build a Steiner tree where the terminals are the contracted components. We assume that $\rho_{ST} \leq 2$. Let $T = T' \cup T''$ denote the complete tree on the open facilities.

Remark 6.5.1 It is possible that the tree T contains an edge with a non-vertex location as an end-point — this will happen if such a location is a leaf of the tree. We delete such edges to get a new tree that only uses edges of the original graph.

6.5.1 Analysis

Let F^*, C^*, S^* denote respectively the facility cost, assignment cost and connection cost of an optimal (integer) solution, $\mathcal{O}^* = F^* + C^* + S^*$ denotes the optimal cost.

Applying Lemma 6.4.1 with L' as the set A, we obtain that the cost of an optimal Steiner tree on L' is at most $\mathcal{O}^* + \sum_{j \in D'} c_{\sigma(j)j}$. Clearly the cost of the optimal tree on the *components* of T' is no more since each component of T' contains at least one terminal location in L'. So the cost of tree T'' constructed in Phase 2 is at most $\rho_{ST} \cdot \mathcal{O}^* + 2\sum_{j \in D'} c_{\sigma(j)j}$. In Lemma 6.5.6 we bound the sum of $2\sum_{j \in D'} c_{\sigma(j)j}$ and the remaining cost of opening facilities, assigning clients, and buying the Steiner edges in T' by $7 \cdot \mathcal{O}^*$, showing that the total cost incurred is at most $(7 + \rho_{ST}) \cdot \mathcal{O}^*$.

The proof of the following lemma is very similar to the proof of Lemma 6.4.2.

Lemma 6.5.2 $\left(\alpha^{(1)}, \beta^{(1)}, \theta^{(1)}\right)$ *is a feasible dual solution.*

Lemma 6.5.3 *Let l be a terminal location and i be its corresponding terminal facility. Then $c_{il} \leq \min_{j \in D_l} 2(\alpha_j^{(1)} - \beta_{ij}^{(1)}) \leq 2\phi_l$.*

Proof : Let j be any client in D_l and k be the representative client of l, so k is connected to i. Then, $c_{il} \leq 2\alpha_k^{(1)} \leq 2\phi_l$. So if $\beta_{ij}^{(1)} = 0, c_{il} \leq 2(\alpha_j^{(1)} - \beta_{ij}^{(1)})$. Otherwise, let t_j be the time at which j became a slave. Note that $\alpha_j^{(1)} = \max(t_j, c_{ij}) + \beta_{ij}^{(1)}$ and $c_{lj} \leq t_j$, so $c_{il} \leq 2(\alpha_j^{(1)} - \beta_{ij}^{(1)})$. ∎

Lemma 6.5.4 *Let l and l' be dependent terminal locations with $\phi_l \leq \phi_{l'}$. If i is the terminal facility corresponding to l, $c_{il'} \leq 6\phi_{l'}$.*

Proof : Let k be the representative client of location l. Let i' be the terminal facility for l' and k' be the representative client of l'. By Lemma 6.5.3, $c_{il} \leq 2\phi_l$ and $c_{i'l'} \leq 2\phi_{l'}$. Let t_i and $t_{i'}$ be the times at which i and i' were tentatively opened respectively. There are four cases to consider, depending on why l and l' are dependent.

1. $\exists j \in D_l \cap D_{l'}$. Since j was free when it became tight with l and l', $c_{lj}, c_{l'j} \leq \max(t_l, t_{l'}) \leq \max(\phi_l, \phi_{l'}) = \phi_{l'}$. If we apply Lemma 6.5.3, we obtain that $c_{il'} \leq c_{il} + c_{ll'} \leq 4\phi_{l'}$.

2. $\exists j$ such that $\beta_{ij}^{(1)}, \beta_{i'j}^{(1)} > 0$. This implies that $c_{i'j}, c_{ij} \leq \alpha_j^{(1)} \leq t_{i'}, t_i$. So $c_{ii'} \leq 2t_{i'} \leq 2\alpha_{k'}^{(1)} \leq 2\phi_{l'}$, and $c_{il'} \leq 4\phi_{l'}$.

3. There is a terminal location r (which could be l), client $j \in D_r$ such that i is the terminal facility for r and $\beta^{(1)}_{i'j} > 0$. By the above argument, $c_{i'j} \leq \alpha^{(1)}_j \leq t_{i'} \leq \phi_{l'}$, and $c_{ij} \leq c_{ir} + c_{jr} \leq 3\alpha^{(1)}_j$ using Lemma 6.5.3. So $c_{ii'} \leq 4\phi_{l'} \implies c_{il'} \leq 6\phi_{l'}$.

4. There is a terminal location r (which could be l'), client $j \in D_r$ such that i' is the terminal facility for r and $\beta^{(1)}_{ij} > 0$. As above, $c_{ii'} \leq 4\phi_l \implies c_{il'} \leq 6\phi_{l'}$.

■

For an open facility i, define $\mathcal{C}_i$ as the set of demand points j for which $\beta^{(1)}_{ij} > 0$. Let $\mathcal{C}_{F'} = \cup_{i \in F'} \mathcal{C}_i$. Note that by our independent set construction, the sets $\mathcal{C}_i$ are disjoint, and all clients in $\mathcal{C}_i$ are assigned to i. Recall that T' is the set of Steiner edges added in Phase 1, and $i(j)$ is the facility to which j is assigned.

Lemma 6.5.5 $\mathsf{cost}(T') \leq 2\sum_{j \in D'} \alpha^{(1)}_j - 2\sum_{j \in D' \cap \mathcal{C}_{F'}} \beta^{(1)}_{i(j)j}$.

Proof : $\mathsf{cost}(T') \leq \sum_{l \in L'} Mc_{i_l l}$ where i_l is the terminal facility corresponding to l. Consider any terminal location $l \in L'$ with terminal facility i. By Lemma 6.5.3, $c_{il} \leq 2(\alpha^{(1)}_j - \beta^{(1)}_{ij})$ for any $j \in D_l$. Since $|D_l| \geq M$, $Mc_{il} \leq \sum_{j \in D_l} 2(\alpha^{(1)}_j - \beta^{(1)}_{ij}) = 2\sum_{j \in D_l} \alpha^{(1)}_j - 2\sum_{j \in D_l \cap \mathcal{C}_{F'}} \beta^{(1)}_{i(j)j}$ since $\beta^{(1)}_{ij} > 0 \implies j \in \mathcal{C}_{F'}$ and $i(j) = i$ for $j \in D_l$ by our independent set construction. Summing over all $l \in L'$ proves the lemma. ■

Lemma 6.5.6 *The solution obtained satisfies*

$$7\sum_{i \in F'} f_i + \sum_j c_{i(j)j} + \mathsf{cost}(T') + 2\sum_{j \in D'} c_{\sigma(j)j} \leq 7\sum_j \alpha^{(1)}_j.$$

Proof : We will charge each j an amount $\mathsf{charge}(j)$ such that

$$7\sum_{i \in F'} f_i + \sum_j c_{i(j)j} + \mathsf{cost}(T') + 2\sum_{j \in D'} c_{\sigma(j)j} \leq \sum_j \mathsf{charge}(j) \leq 7\sum_j \alpha^{(1)}_j. \tag{8}$$

$$\text{Set } \mathsf{charge}(j) = \begin{cases} c_{i(j)j} + 7\beta^{(1)}_{i(j)j} & \text{if } j \in \mathcal{C}_{F'} - D' \\ c_{i(j)j} + 7\beta^{(1)}_{i(j)j} + 2(\alpha^{(1)}_j - \beta^{(1)}_{i(j)j}) + 2c_{\sigma(j)j} & \text{if } j \in \mathcal{C}_{F'} \cap D' \\ c_{i(j)j} + 2\alpha^{(1)}_j + 2c_{\sigma(j)j} & \text{if } j \in D' - \mathcal{C}_{F'} \\ c_{i(j)j} & \text{if } j \notin D' \cup \mathcal{C}_{F'} \end{cases}.$$

The first inequality in (8) follows from Lemma 6.5.5 and the fact that for each $i \in F'$, all j in $\mathcal{C}_i$ are assigned to i and $\sum_{j\in\mathcal{C}_i} \beta_{ij}^{(1)} = f_i$. To prove the second inequality, note that if $j \in \mathcal{C}_{F'}$ then $c_{i(j)j} + \beta_{i(j)j}^{(1)} \leq \alpha_j^{(1)}$. If $j \in D'$ then $c_{\sigma(j)j} \leq t_{\sigma(j)j} \leq \alpha_j^{(1)} - \beta_{i(j)j}^{(1)}$ as argued in Lemma 6.5.3. Also if $j \in D' \setminus \mathcal{C}_{F'}$ then $c_{i(j)j} \leq 3\alpha_j^{(1)}$. So if $j \in D' \cup \mathcal{C}_{F'}, \mathsf{charge}(j) \leq 7\alpha_j^{(1)}$.

Consider $j \notin D' \cup \mathcal{C}_{F'}$. We show that $c_{i(j)j} \leq 7\alpha_j^{(1)}$. Let $l' \in L - L'$ be the location that caused j to become a slave and let $\sigma(j) = l \in L'$. Clearly $\alpha_j^{(1)} \geq t_{l'}$ and since $j \in D_{l'}, \alpha_j^{(1)} \geq \phi_{l'}$. Since $\sigma(j) = l$, l and l' are dependent with $\phi_l \leq \phi_{l'}$, and $i(j)$ is the terminal facility corresponding to l. So by Lemma 6.5.4, $c_{i(j)l'} \leq 6\phi_{l'}$. This implies that $c_{i(j)j} \leq 7\alpha_j^{(1)}$. ∎

Putting the pieces together and taking $\rho_{ST} = 1.55$, we get the following.

Theorem 6.5.7 *Using the* 1.55*-approximation algorithm of [65], the algorithm above produces a solution of cost at most* $8.55 \cdot OPT$*.*

Proof : The total cost incurred is $\sum_{i\in F'} f_i + \sum_j c_{i(j)j} + \mathsf{cost}(T') + \mathsf{cost}(T'')$ and by our earlier discussion, $\mathsf{cost}(T'') \leq 2\sum_{j\in D'} c_{\sigma(j)j} + \rho_{ST} \cdot \mathcal{O}^*$. Using Lemma 6.5.6 now proves the result. ∎

Bounding the Integrality Gap

As in the previous section we can simulate the algorithm of [2, 26] with $\rho_{ST} = 2$ in Phase 2 to obtain a solution with cost at most $9 \cdot OPT$, thereby showing that the integrality gap of (ConFL-P) is at most 9.

We initialize our tree T to T'. As before, a minimal violated set (MVS) is a minimal set S such that $S \cap L' \neq \emptyset, v \notin S$ and $\delta(S) \cap T = \emptyset$. Initially these are just the components of T' not containing v. All dual variables are initially 0. We do not raise any β_{ij} in this phase. We shall raise the α_j value of clients in D' only. For a set S, define D_S to be $\bigcup_{l\in S\cap L'} D_l$. The rest of the procedure is identical to the procedure described in the previous section. This yields the tree $T = T' \cup T''$ connecting all the open facilities, where T'' denotes the set of Steiner edges added by this process. Let $\left(\alpha^{(2)}, 0, \theta^{(2)}\right)$ be the dual solution constructed.

Theorem 6.5.8 *The above algorithm produces a solution of cost at most* $9 \cdot OPT$*.*

Proof : As in Lemma 6.4.5, $\mathsf{cost}(T'') = 2\sum_j \alpha_j^{(2)}$, and by Lemma 6.4.6, $\left(\alpha', 0, \theta^{(2)}\right)$ is a feasible dual solution where $\alpha'_j = \max(\alpha_j^{(2)} - c_{\sigma(j)j}, 0)$. So $\mathsf{cost}(T'') \leq 2 \cdot OPT + 2\sum_{j \in D'} c_{\sigma(j)j}$ and $\mathsf{cost}(T) \leq 2 \cdot OPT + 2\sum_{j \in D'} c_{\sigma(j)j} + \mathsf{cost}(T')$. Adding this to $\sum_{i \in F'} f_i + \sum_j c_{i(j)j}$ and using Lemma 6.5.6, we get the claimed bound. ∎

6.6 Generalization to Edge Capacities

We can extend our results to a capacitated generalization of connected facility location where edges have capacities. Each edge has a *length* c_e. We are given two kinds of cables; one has a cost of σ per unit length and capacity u, the other has a cost of M per unit length and infinite capacity. We wish to open facilities and lay a network of cables so that clients are connected to open facilities using the first kind of cable. Furthermore, we want the facilities to be connected to each other by a Steiner tree using cables of the second type. We may install multiple copies of a cable along an edge, if necessary, to handle the total demand through the edge. So routing d units of demand through edge e now costs $\sigma\lceil \frac{d}{u} \rceil c_e$, whereas earlier the cost was simply $d \cdot c_e$. Assuming integer demands, the uncapacitated problem considered earlier is a special case obtained by setting $u = 1$, scaling edge costs by σ and M by $\frac{1}{\sigma}$. The facility location aspect of this problem where we only have cables of the first type and do not require that facilities be interconnected was considered in [62].

The rent-or-buy case with $\mathcal{F} = V$, $f_i = 0$ for all i now corresponds to a rent-or-buy problem where we can either buy unlimited capacity on an edge paying a large fixed cost of M per unit length, or rent capacity in steps of u units, paying a cost of σ per unit length for every u units installed.

We assume $\sigma \leq M$ (since otherwise the optimal solution is just a Steiner tree connecting the clients to v). We only consider the unit demand case. In the case of arbitrary demands, this approach yields somewhat worse guarantees. The details may be found in [75]. We use a theorem of Hassin, Ravi & Selman [37] (see also [62]) stated in a slightly different form.

Theorem 6.6.1 *Let Z be a Steiner tree on a set of terminals D rooted at v where each edge has capacity u. Let w_j be a weight associated with terminal $j \in D$. We can clump the terminals into subtrees $Z_1, \ldots, Z_k$ so that,*

(i) Each subtree except possibly Z_k has exactly u terminals and Z_k has at most u terminals.

(ii) If we route flow along edges of Z from the $u-1$ terminals in Z_i to the terminal in Z_i with minimum weight for each $i < k$, and route flow from the terminals in Z_k to v, then we get a flow that respects edge capacities.

We can get a $(\rho_{ConFL} + \rho_{ST})$-approximation algorithm for this problem by using a ρ_{ConFL}-approximation algorithm for ConFL and a ρ_{ST}-approximation algorithm for the Steiner tree problem.

C1. Obtain a ConFL instance by setting the edge costs to $c'_e = \frac{\sigma c_e}{u}$ and $M' = \frac{Mu}{\sigma}$. A solution to the original instance gives a solution to the ConFL instance of no greater cost — the Steiner edges cost the same and the cost of routing d units of demand through a facility location edge is $d \cdot \frac{\sigma c_e}{u} \leq \sigma\lceil\frac{d}{u}\rceil c_e$. We solve this relaxation approximately using the ρ_{ConFL}-approximation algorithm. Let $i(j)$ be the facility to which j is assigned and T be the Steiner tree on the open facilities.

C2. Obtain a Steiner tree instance by setting the edge costs to σc_e with the terminals being the demand points and vertex v. This is a relaxation, since a solution to the original instance connects all demand points to open facilities and all open facilities to v with each edge costing at least σc_e, be it a facility location edge or a Steiner tree edge (since $M \geq \sigma$). We solve this Steiner tree instance approximately. Let Z be the resulting tree.

C3. Now we combine the two near-optimal solutions to get a feasible solution of cost no greater than the sum of the costs of the two solutions. We use Theorem 6.6.1 with the tree Z and $w_j = c'_{i(j)j}$ for demand point j. Let $Z_1, \ldots, Z_k$ be the subtrees obtained. We first route demand in each subtree along edges of Z as specified in the theorem. For each subtree $Z_i, i < k$, the u units of demand collected at the client $j \in Z_i$ for which $c'_{i(j)j}$ is minimum is then sent to facility $i(j)$ along the path from j to $i(j)$.

The cost of routing demand along Z is at most the cost of Z in the Steiner tree instance since each edge of Z carries at most u units of demand. Routing demand along the path from $j \in Z_i$ to $i(j)$ costs $\sigma c_{i(j)j} \leq \sum_{k \in Z_i} \frac{\sigma c_{i(k)k}}{u} = \sum_{k \in Z_i} c'_{i(k)k}$. The only facilities we use are v and the facilities opened in the ConFL solution and these are connected by the tree T which has the same cost in both the original instance and the ConFL instance. So we get a feasible solution of cost at most $(\rho_{ConFL} + \rho_{ST}) \cdot OPT$. Taking $\rho_{ST} = 1.55$ [65] and using Theorems 6.4.4 and 6.5.7 we obtain the following theorem.

Theorem 6.6.2 *There is a* 10.1*-approximation algorithm for Connected Facility Location with edge capacities and unit demands. For the case* $\mathcal{F} = V$ *and* $f_i = 0$ *for all* i*, there is a* 6.1*-approximation algorithm.*

6.7 Extensions and Refinements

Arbitrary Demands. Suppose instead of unit demands, each client j has a demand of $d_j \geq 0$. The results of Section 6.4 and Section 6.5 extend to this case. A simple way to handle this is to make d_j copies of client j. But this only gives a pseudo-polynomial time algorithm. We can however simulate this reduction.

In Phase 1, we raise each α_j at a rate of d_j. The variables $\beta_{ij}, \theta_{S,j}$ responding to the increase in α_j, also increase at rate d_j. We modify the definition of tightness to reflect this by replacing α_j with α_j/d_j, i.e., we say that j is tight with i if $\alpha_j/d_j \geq c_{ij}$. Instead of the number of clients tight with a location l, we now consider the *total* *demand* that is tight with l, i.e., $\sum_{j:\alpha_j \geq d_j c_{ij}} d_j$, and in the general case we again consider only clients j bound to l. In the general case, the representative client for a terminal location l is now the client k bound to l with smallest $\alpha_j^{(1)}/d_j$ value, and we set $\phi_l = \max\bigl(\alpha_k^{(1)}/d_k, t_l\bigr)$. To bound the integrality gap, when we raise dual variables in Phase 2 we raise α_j and $\theta_{S,j}$ at a rate of $\frac{d_j}{\sum_{j \in D_S} d_j} \leq \frac{d_j}{M}$ so that θ_S increases at a rate of 1. The analogues of lemmas proved in Sections 6.4 and 6.5 are easily shown to be true and we get the same approximation ratios. The guarantees of Section 6.6, however suffer slightly.

The case $M = 1$. We can get significantly better results for this case. In Phase 1, we run the Jain-Vazirani primal-dual algorithm for uncapacitated facility location described in Section 2.2. Note that we never raise any dual variables $\theta_{S,j}^{(1)}$. Let $\bigl(\alpha^{(1)}, \beta^{(1)}, 0\bigr)$ be the dual solution constructed. Let F' be the set of opened facilities, $i(j)$ be the facility to which j is assigned, and $\mathcal{C}_{F'}$ be the set of clients j such that $\beta_{ij}^{(1)} > 0$ for some facility $i \in F'$. Recall that the Jain-Vazirani algorithm ensures that for every client j there is at most one facility $i \in F'$ such that $\beta_{ij}^{(1)} > 0$, and it assigns all clients with $\beta_{ij}^{(1)} > 0, i \in F'$ to i. Further, every client $j \notin \mathcal{C}_{F'}$ is assigned to an open facility that is at most $3\alpha_j^{(1)}$ distance away. For any i in F', $f_i = \sum_{j \in \mathcal{C}_{F'}: i(j)=i} \beta_{ij}^{(1)}$ and if $j \in \mathcal{C}_{F'}$ then $c_{i(j)j} + \beta_{i(j)j}^{(1)} = \alpha_j^{(1)}$.

For each $i \in F'$ we identify a client j connected to i such that $\beta_{ij}^{(1)} > 0$. Call this the *primary demand point* for i. We add edges on the path from i to j to the Steiner tree and contract these edges to form a supernode w_i. Also make v a supernode, if it is not already included in some supernode. In Phase 2, a Steiner tree is built on the supernodes using the primal-dual algorithm of [2, 26]. Only the primary demand points pay for the Steiner tree by increasing their α_j variables. Let $\bigl(\alpha^{(2)}, 0, \theta^{(2)}\bigr)$ be the dual solution, and $D' \subseteq \mathcal{C}_{F'}$ be the set of primary demand points.

Theorem 6.7.1 *The cost of the solution produced is at most* $4 \cdot OPT$.

Proof : By arguing as in Lemmas 6.4.5 and 6.4.6, we get that the cost of the tree on the supernodes is at most $2\sum_j \alpha_j^{(2)}$ and that $\bigl(\alpha^{(2)}, 0, \theta^{(2)}\bigr)$ is now a *feasible* dual solution. The total cost is bounded by $\sum_{i \in F'} f_i + \sum_j c_{i(j)j} + \sum_{j \in D'}(c_{i(j)j} + 2\alpha_j^{(2)}) \leq \sum_{j \in D'}(2\alpha_j^{(1)} + 2\alpha_j^{(2)}) + \sum_{j \notin D'} c_{i(j)j}$. For $j \in D'$ and any i, $2\alpha_j^{(1)} + 2\alpha_j^{(2)} \leq 4c_{ij} + 2\beta_{ij}^{(1)} + 2\sum_{S \subseteq V: i \in S, v \notin S} \theta_{S,j}^{(2)}$, and for $j \notin D'$, $c_{i(j)j} \leq 3\alpha_j^{(1)} \leq 3c_{ij} + 3\beta_{ij}^{(1)}$ for any i. So the cost is at most 4 times the value of a dual feasible solution, hence at most $4 \cdot OPT$. ■

This gives a 5.55-approximation algorithm for the edge capacitated version discussed in Section 6.6.

The Connected *k*-Median Problem. In Section 7.4 we consider a variant of connected facility location where we impose the additional requirement that at most k facilities may be opened. We use the primal-dual algorithm from Section 6.5 as a black box to obtain a constant-factor approximation ratio for this problem. The algorithm for the connected k-median problem can then be used as a subroutine to obtain results for variants and special cases of the problem involving edge capacities, unit/arbitrary demands, $M = 1$ vs. $M > 1$. The details may be found in [75].

Chapter 7

k-Median Problems

7.1 Introduction

In various facility location settings, in place of, or in addition to, the facility opening costs, there may be a bound imposed on the number of facilities that may be opened. For example, the actual facility cost might consist of a long-term running cost and a short-term opening cost, and we want to minimize the total long-term running cost of the facilities and the client assignment costs, subject to the constraint that the short-term opening cost is within a certain budget. If the short-term costs of the different facilities are more or less comparable, then this translates to a cardinality bound on the number of facilities that may be opened. So we could model the problem by setting the fixed cost of a facility to its long-term running cost, with the objective being to minimize the sum of the facility opening costs and client assignment costs subject to the additional constraint that at most k facilities are opened (i.e., the short-term cost does not exceed a budget). Now suppose that there are no facility opening costs in the above problem (but there still is a bound of k on the number of facilities we may open) and facilities may be opened at any location, then the problem may also be described as follows: given a set of points (clients) in a metric space, we want to choose k of them as medians (facilities) and assign each point to a median so as to minimize the total assignment cost, that is, the sum of the distances from each point to its assigned median. This is the classical *k-median* clustering problem. One may view each median as creating a cluster around it consisting of all the points that are nearest to it, and the goal is to find the k centers which yield the best clustering of the point set (under the k-median objective function). To avoid confusion, unless it is otherwise clear from the context, whenever we say "the k-median problem", we are referring to the k-median clustering problem where there are *no facility opening costs* and *facilities may be opened anywhere* and we call the k-median version of UFL where there may also be facility opening costs "the k-facility location problem".

In this chapter we consider the k-median versions of some facility location problems considered earlier and devise constant-factor approximation algorithms for these problems. The k-median version of a facility location problem Π is the problem where in addition to the constraints of the problem Π, there is an added constraint that specifies that at most k facilities may be opened.

The same high-level framework is used to obtain all these approximation guarantees: we will use an algorithm devised for the facility location problem Π, and the fact that this algorithm satisfies certain desired properties to obtain an approximation algorithm for the k-median version of problem Π. To see the connection between a facility location problem and its corresponding k-median version, consider the classical k-median clustering problem. Given an instance with a set $\mathcal{N}$ of points located in a metric space, consider the UFL instance where each point is both a facility and a client, i.e., $\mathcal{F} = \mathcal{D} = \mathcal{N}$, and every facility has an opening cost of λ, but there is no restriction on the number of facilities that may be opened. When $\lambda = 0$, any reasonable solution to the UFL instance would open a facility at each point thus opening $|\mathcal{D}|$ facilities; on the other extreme if λ is *very* large, then we would open just one facility and assign every point to this facility. The variable λ is thus a *Lagrangian multiplier* (or a dual variable) that penalizes the violation of the "hard" cardinality constraint limiting the number of open facilities, and the uncapacitated facility location problem with facility costs set to λ arises as the *Lagrangian relaxation* of the k-median problem. It seems natural, that by adjusting the value of λ one should be able to get a solution, using an algorithm for UFL, that opens k facilities, and that such a solution may be a good solution for the k-median problem. This idea, does in fact work, and we exploit it to derive approximation algorithms for various k-median problems.

7.1.1 Summary of Results

We illustrate the Lagrangian relaxation method described above by considering three specific examples. Jain & Vazirani [41] introduced this technique and used it to obtain an elegant 6-approximation algorithm for the k-facility location problem. We describe their algorithm in Section 7.2. In Sections 7.4 and 7.5 we look at the k-median versions of the connected facility location problem (Chapter 6) and the facility location problem with service installation costs (Chapter 5) respectively, and give constant-factor approximation algorithms for these problems. Each of these algorithms, uses as a subroutine, an algorithm that was devised for the corresponding facility location problem. The algorithm for the k-median version of UFL due to Jain & Vazirani uses the primal-dual 3-approximation algorithm described in Section 2.2. For the connected k-median problem we use the algorithm developed in Section 6.5, and for the k-median version of facility location with service installation costs, we use the primal-dual algorithm from Section 5.4.

7.1.2 Related Work

Although the classical k-median problem has been extensively studied in various disciplines as a clustering problem and various heuristics have been devised for it (e.g., the k-means algorithm), the first constant-factor approximation algorithm for this problem was given relatively recently by Charikar, Guha, Tardos & Shmoys [16] based on LP rounding. They also gave an algorithm for the k-facility location problem, i.e., the k-median version of UFL where facilities may have opening costs. Jain & Vazirani [41] gave a 6-approximation algorithm for the k-median problem based on their primal-dual algorithm for UFL. This was subsequently improved to 4 [15, 40].

All these results are LP-based results and also give upper bounds on the integrality gap of the k-median LP. Most recently, Archer, Rajagopalan & Shmoys [6] showed that the integrality gap of the k-median LP is at most 3, but their proof does not yield a polynomial time algorithm. The guarantees proved in these papers also carry over to the k-facility location problem. The current best approximation guarantee for the k-median problem is $(3+\epsilon)$ due to Arya, Garg, Khandekar, Meyerson, Munagala & Pandit [7] and is obtained by a local search procedure. Their algorithm can be adapted to get a slightly worse approximation guarantee for the k-facility location problem. To the best of our knowledge, the connected k-median problem and the k-median problem with service installation costs seem to be new problems that have not been considered earlier in the literature.

7.2 The *k*-Facility Location Problem

We now describe the algorithm of Jain & Vazirani for the classical k-facility location problem. The input to the problem is a set of facilities $\mathcal{F}$ and a set of clients $\mathcal{D}$ and a number k, and a solution consists of opening at most k facilities and assigning each client to an open facility. The goal is to minimize the total facility opening and client assignment costs. As usual we will assume that clients have unit demand. The LP relaxation of this problem and its dual are as follows:

$$\min \sum f_i y_i + \sum_{j,i} c_{ij} x_{ij} \qquad \text{(KFL-P)}$$

$$\begin{aligned}
\text{s.t.} \quad & \sum_i x_{ij} \geq 1 && \forall j \\
& x_{ij} \leq y_i && \forall i,j \\
& \sum_i y_i \leq k && && (1) \\
& x_{ij}, y_i \geq 0 && \forall i,j.
\end{aligned}$$

$$\max \sum_j \alpha_j - k\lambda \qquad \text{(KFL-D)}$$

$$\begin{aligned}
\text{s.t.} \quad & \alpha_j \leq c_{ij} + \beta_{ij} && \forall i,j \\
& \sum_j \beta_{ij} \leq f_i + \lambda && \forall i && (2) \\
& \alpha_j, \beta_{ij}, \lambda \geq 0 && \forall i,j.
\end{aligned}$$

Constraint (1) limits the number of facilities opened to k. Let OPT_k denote the common optimal value.

Let us first give some intuition about the primal and the dual problems. The dual problem has a variable λ corresponding to constraint (1) that penalizes the violation of this constraint. Suppose we drop constraint (1) from the primal and instead add the penalty term $\lambda(\sum_i y_i - k)$ to the primal objective function, so that the primal problem is to minimize $\sum_i f_i y_i + \sum_{j,i} c_{ij} x_{ij} + \lambda(\sum_i y_i - k)$ subject to the remaining primal constraints above. But this is essentially a UFL instance (with an additional $-k\lambda$ term which is constant for a fixed λ) with the cost of each facility i set to $f_i + \lambda$. Further for any value of λ, the value of this minimization problem provides a lower bound on OPT_k, since any solution to (KFL-P) yields a feasible solution to this UFL instance of no greater objective value (since $\sum_i y_i \leq k$). Therefore to get the best lower bound we can take the maximum value of this minimization problem over all values of λ, and this is precisely the dual problem (KFL-D). If we consider λ as fixed in (KFL-D) and only maximize over the α_j and β_{ij} variables, then (KFL-D) is simply the dual problem for the UFL instance we obtained above with the facility costs set to $f_i + \lambda$ (the extra $-k\lambda$ term in the objective function corresponds to the $-k\lambda$ term in the UFL minimization objective), and therefore by maximizing over λ in (KFL-D) we are aiming to get the best lower bound on the value of the primal program (KFL-P).

We use this connection with UFL in the following way. Suppose we fix the value of λ, and run the JV primal-dual algorithm from Section 2.2 on the UFL instance where the cost of each facility i is set to $f_i + \lambda$. Let $(\tilde{x}, \tilde{y})$ be the integer UFL primal solution and (α, β) be the UFL dual solution constructed by the algorithm. Notice that (α, β, λ) is a feasible solution to (KFL-D). Suppose in the primal solution, exactly k facilities are opened. Then $(\tilde{x}, \tilde{y})$ is a feasible integer solution to (KFL-P). Furthermore, from the guarantee we proved in Theorem 2.2.2 for the JV algorithm, we get that $3\sum_i (f_i + \lambda)\tilde{y}_i + \sum_{j,i} c_{ij}\tilde{x}_{ij} \leq 3\sum_j \alpha_j$, or equivalently,

$$3\sum_i f_i \tilde{y}_i + \sum_{j,i} c_{ij}\tilde{x}_{ij} \leq 3\Big(\sum_j \alpha_j - \lambda \sum_i \tilde{y}_i\Big) = 3\Big(\sum_j \alpha_j - k\lambda\Big) \leq 3 \cdot OPT_k,$$

where the last inequality follows since (α, β, λ) is a feasible solution to (KFL-D). The trick then is to *guess* the right value of λ so that when the facility costs are set to $f_i + \lambda$, the JV algorithm ends up opening k facilities.

Recall that in the JV algorithm, we decide which subset of tentatively open facilities to open by picking a maximal independent set, and when we described the algorithm in Section 2.2 we did not specify any particular way of picking the maximal independent set. We now fix the way in which we pick the maximal independent set: consider the tentatively open facilities in the order they were tentatively opened and pick a maximal independent set greedily, that is, add facility i to the current independent set if adding it preserves independence. Note that with this rule of picking the maximal independent set, the primal solution we construct depends on the *order in which events happen* in the dual ascent process, and hence on the way in which we break ties between events that happen at the same time in the dual ascent process; the dual solution constructed is however independent of the order in which ties are broken.

Suppose the the JV algorithm opens at most k facilities when $\lambda = 0$. Then, $(\alpha, \beta, 0)$ is a feasible solution to (KFL-D) of value $\sum_j \alpha_j$ and the primal solution obtained is a feasible k-facility location solution of cost at most $3\sum_j \alpha_j \leq 3 \cdot OPT_k$. So suppose that at $\lambda = 0$ the JV algorithm opens more than k facilities. When λ is very large, say, $\lambda = |\mathcal{D}| \max_{ij} c_{ij}$, the JV algorithm will open just one facility and assign every client to this facility. It seems reasonable to expect that there is an intermediate value of λ at which the JV algorithm opens exactly k facilities. If we could find such a primal solution, then as shown above, that would give us a solution of cost at most $3 \cdot OPT_k$. Unfortunately such a value of λ need not exist, that is, the number of facilities opened by the JV algorithm need not decrease in a continuous fashion as we increase λ from 0. However we will show that by doing a bisection search in the range $[0, |\mathcal{D}| \max_{ij} c_{ij}]$ and stopping when the search interval becomes sufficiently small, we can get in polynomial time two primal (integer) solutions, one opening $k_1 < k$ facilities and the other opening $k_2 > k$ facilities, such that these two distinct primal solutions correspond to a *single value of* λ, and may be obtained by running the JV algorithm with that value of λ and breaking ties between events in the dual ascent process appropriately.

Assume for now that we have these two primal solutions (x_1, y_1) and (x_2, y_2) opening $k_1 < k$ and $k_2 > k$ facilities respectively obtained at $\lambda = \lambda_0$ and that (α, β) is the common dual solution constructed by the JV algorithm. An important fact worth pointing out is that *we use the values α, β and λ_0 only in the analysis, and not in the algorithm.* Let (F_1, C_1) and (F_2, C_2) denote respectively the cost of the solutions (x_1, y_1) and (x_2, y_2) where F_i denotes the facility cost, and C_i denotes the assignment cost. Then,

$$3(F_1 + k_1\lambda_0) + C_1 \leq 3\sum_j \alpha_j, \qquad \text{and} \qquad 3(F_2 + k_2\lambda_0) + C_2 \leq 3\sum_j \alpha_j.$$

A *convex combination* of these two solutions yields a fractional solution (x, y) that opens exactly k facilities and in which every client is assigned to at most two facilities. Let a and b be such that $ak_1 + bk_2 = 1, a + b = 1$. So,

$$3(aF_1 + bF_2) + (aC_1 + bC_2) \leq 3\Big(\sum_j \alpha_j - k\lambda_0\Big) \leq 3 \cdot OPT_k. \tag{3}$$

We now round the fractional solution (x, y) using a rounding procedure described in [41] to get an integer solution that opens at most k facilities losing a factor of at most 2, and thus get a 6-approximation algorithm.

We call a facility opened in (x_1, y_1) a "small" facility, and a facility opened in (x_2, y_2) a "large" facility. For each small facility we look at the large facility nearest to it. Let N be this set of large facilities. If $|N| < k_1$, then we arbitrarily add $k_1 - |N|$ large facilities (that are not already in N) to N. With probability a we open all the small facilities and with probability $1 - a = b$ we open all the facilities in N. This opens exactly k_1 facilities. Next we randomly choose a set of $k - k_1$ large facilities not in N and open all of these. Note that each such facility is opened with probability $(k - k_1)/(k_2 - k_1) = b$. It is clear that we open exactly k facilities this way, and that the expected facility cost is at most $aF_1 + bF_2$.

To bound the assignment cost, consider a demand j and let i_1, i_2 be the facilities to which it is assigned in y_1, y_2 respectively. If $i_2 \in N$, then exactly one of i_1 and i_2 is open, and the expected assignment cost of j is $ac_{i_1j} + bc_{i_2j}$. Otherwise, let

i_3 be the facility nearest to i_1 in y_2, so $i_3 \in N$ and one of i_1, i_3 is opened. We assign j to i_2 if it is open and otherwise to i_1 or i_3, whichever is open. Since $c_{i_3 j} \leq c_{i_1 j} + c_{i_1 i_3} \leq c_{i_1 j} + c_{i_1 i_2} \leq 2c_{i_1 j} + c_{i_2 j}$ the expected assignment cost is at most, $bc_{i_2 j} + a(ac_{i_1 j} + bc_{i_3 j}) \leq bc_{i_2 j} + a\big((1+b)c_{i_1 j} + bc_{i_2 j}\big) \leq \max(1+a, 1+b)(ac_{i_1 j} + bc_{i_2 j})$. So the total assignment cost is at most $2(aC_1 + bC_2)$ and the expected total cost is at most, $(aF_1 + bF_2) + 2(aC_1 + bC_2) \leq 6 \cdot OPT_k$ using (3).

Theorem 7.2.1 *The above algorithm is a* 6*-approximation algorithm for the* k*-Facility Location problem.*

7.2.1 Obtaining the Solutions (x_1, y_1) and (x_2, y_2)

We briefly describe how to obtain the two solutions (x_1, y_1) and (x_2, y_2) with the required properties.

In the dual ascent process each pair (i, j) and facility i' corresponds to an event; the pair (i, j) corresponds to the event that at time t, $\alpha_j = t = c_{ij}$ and the facility i' corresponds to the event that at time t, i' gets paid for, i.e., $\sum_k \beta_{i'k} = \sum_k \max(0, \alpha_k - c_{i'k}) = f_{i'}$. Fix an ordering $\mathcal{O}$ of all such possible events, which will be used to break ties in the dual ascent process of the JV algorithm. By this we mean that, if (i, j) comes before facility i' in the ordering $\mathcal{O}$, and in the dual-ascent process if the events corresponding to (i, j) and facility i' both happen at time t, then we break ties in favor of event (i, j) and say that event (i, j) happened before event i'. For a given value of λ, let the sequence for λ denote the sequence of events that occur in the dual-ascent process, listed in the order in which they happen. We say that λ is a *critical point* if an infinitesimal change in λ results in a change in the sequence. We will argue that (1) if λ_0 is a critical point then both the sequence for λ_0 and the sequence for $\lambda_0 \pm \epsilon$ can be obtained *at* $\lambda = \lambda_0$ depending on how we break ties between events, and (2) two critical points are separated by at least $c = 2^{-(\text{poly}(n)+L)}$, where L is the number of bits to represent the largest distance. Given these two facts, suppose we terminate the bisection search when the search interval $[\lambda_2, \lambda_1]$ satisfies $\lambda_1 - \lambda_2 < c$

and $(x_1, y_1), (x_2, y_2)$ be the primal solutions at λ_1, λ_2 respectively that open $k_1 < k$ and $k_2 > k$ facilities respectively. By fact (2), we know that there is a single critical point $\lambda_0 \in [\lambda_1, \lambda_2]$, and by fact (1) there is a way of breaking ties between events so that we get both (x_1, y_1) and (x_2, y_2) as solutions at $\lambda = \lambda_0$. These two solutions, which can be found in polynomial time, satisfy all the required properties. Note that we do not explicitly need to find the value of λ_0 or the dual solution (α, β) at $\lambda = \lambda_0$.

We now argue briefly that facts (1) and (2) hold. The details may be found in the preliminary version of [41] (Section 3.2). To show fact (1) suppose that λ_0 is a critical point with associated sequence s and that an infinitesimal change results in a different sequence s'. Then it suffices to note that breaking ties according to the ordering $\mathcal{O}'$ which lists s' first followed by an arbitrary ordering of the events that do not appear in s', will result in the sequence s' at $\lambda = \lambda_0$. To show fact (2), suppose that we get sequence s' at $\lambda = \lambda_0 + \epsilon$ (the argument is similar if s' is obtained at $\lambda = \lambda_0 - \epsilon$). Then we can write $\lambda_0 = \inf\{\lambda : \lambda \text{ gives sequence } s'\}$. We will express λ_0 as the optimal solution to a polynomial size linear program, which will show that we can write λ_0 using at most $\log(1/c) = \text{poly}(n) + L$ bits. The linear program will have variables $t_1, t_2, \ldots$ representing the times at which the events in s' take place with $t_1 \leq t_2 \leq \ldots$. Since we know the entire sequence of events, we can express the values of α_j and β_{ij} at any time t_i in terms of the variables $t_1, \ldots, t_i$. For each t_i, we write three types of constraints which encode that (a) the event s'_i corresponding to t_i in s' must occur, (b) any event that comes before s'_i in the ordering $\mathcal{O}$, and after s'_i in sequence s' has not yet occurred, and (c) the dual constraints are satisfied.

7.3 A General Framework

We sketch a generic framework along the above lines, that we will use to obtain approximation algorithms for the k-median versions of other facility location problems. Let $\mathcal{A}$ be a primal-dual γ-approximation algorithm for a facility location problem Π. We require that $\mathcal{A}$ has the stronger guarantee that it returns a solution with facility cost F such that,

$$\gamma \cdot F + \text{remaining primal cost} \leq \gamma \cdot (\text{dual solution value}).$$

The k-median version of Π adds the constraint that at most k facilities be opened, to the primal problem, and modifies the dual problem accordingly. As in the k-facility location problem, for any value of λ, if we fix the facility costs to $f_i + \lambda$, then any feasible solution to the dual of the resulting Π-instance of value Dual_λ gives a feasible solution to the dual of the k-median version of Π of value $\mathsf{Dual}_\lambda - k\lambda$. So if we can find a value of λ such that $\mathcal{A}$ opens exactly k facilities when run on the instance with facility costs set to $f_i + \lambda$, then, since $\gamma \cdot (F + k\lambda) + \text{remaining primal cost} \leq \gamma \cdot \mathsf{Dual}_\lambda$, we get a feasible solution to the k-median version, of cost at most $\gamma \cdot (\mathsf{Dual}_\lambda - k\lambda) \leq \gamma \cdot OPT_k$. The generic algorithm is as follows:

G1. If at $\lambda = 0$, algorithm $\mathcal{A}$ returns a solution that opens at most k facilities, then we have a feasible k-median solution of cost at most $\gamma \cdot \mathsf{Dual}_0 \leq \gamma \cdot OPT_k$.

G2. Otherwise, we do a bisection search between $\lambda = 0$ and $\lambda = \lambda_{\max}$ to find two primal solutions P_1 and P_2 such that P_1 opens $k_1 < k$ facilities and P_2 opens $k_2 > k$ facilities, and *both* P_1 *and* P_2 may be obtained by running algorithm $\mathcal{A}$ with λ set to a common value λ_0 by breaking ties appropriately in the dual ascent process.

G3. A convex combination of P_1 and P_2 yields a fractional primal solution that opens exactly k facilities and is therefore a feasible k-median solution. The cost of this solution is at most $\gamma \cdot (\mathsf{Dual}_{\lambda_0} - k\lambda_0) \leq \gamma \cdot OPT_k$.

G4. We now round this fractional solution to get an integer solution while losing only a constant in the approximation guarantee. This gives an approximation algorithm for the k-median version of problem Π.

Here, the value of $\lambda_{\max}$ in step G2, and the rounding procedure in step G4 will depend on the particular problem that we are considering.

7.4 The Connected k-Median Problem

The Connected k-Median problem is the k-median version of the connected facility location (ConFL) problem. Recall that in ConFL we have a set of facilities $\mathcal{F}$, a set of clients $\mathcal{D}$, and a parameter $M \geq 1$, and our objective is to open facilities, assign each client to an open facility and connect the open facilities by a Steiner tree so as to minimize the total cost of opening facilities, assigning clients and connecting facilities. In the connected k-median problem we are allowed to open at most k facilities. Again, we restrict our attention to unit demands $d_j = 1$, but everything carries over to arbitrary demands.

Since we initially guess an open facility v to formulate the LP relaxation of ConFL, this adds the following inequality to the linear program (ConFL-P) for ConFL: $\sum_{i \neq v} y_i \leq k - 1$. This changes the objective function of the dual (ConFL-D) to max $\sum_j \alpha_j - \sum_j \beta_{vj} - k'\lambda$, where $k' = k - 1$. Constraint (7) in the dual LP gets replaced by $\sum_j \beta_{ij} \leq f_i + \lambda$. Let OPT_k be the common optimal value of the connected k-median primal and dual LPs. We use Phase 1 of the primal-dual algorithm developed in Section 6.5 for ConFL, as algorithm $\mathcal{A}$ in the generic scheme outlined in Section 7.3, and a ρ_{ST}-approximation algorithm for the Steiner tree problem (where $\rho_{ST} \leq 2$), to obtain a $(14 + \rho_{ST})$-approximation for the connected k-median problem. Whenever we say "the ConFL algorithm" we mean Phase 1 of the algorithm given in Section 6.5.

Let (F^*, C^*, S^*) be the cost of an optimal integer connected k-median solution, so $OPT_k \leq \mathcal{O}^* = F^* + C^* + S^*$. Suppose we fix λ, modify the facility opening costs to $f_i + \lambda$ for all $i \neq v$, and run the ConFL algorithm to get a (partial) primal solution (x, y, z), and a dual solution $\left(\alpha^{(1)}, \beta^{(1)}, \theta^{(1)}\right)$. Recall that the algorithm picks a subset L' of the terminal locations, and builds some components connecting each $l \in L'$ to its terminal facility via Steiner edges. D' is the set of clients that are *bound* to locations in L', and $\sigma(j)$ denotes the terminal location in L' associated with demand j. Let (F, C, T') be the cost of the resulting (partial) primal solution, where $F = \sum_i f_i y_i$ is the unmodified facility cost, C is the assignment cost, and T' is the cost of the partial

Steiner tree constructed on the open facilities. We will abuse notation and use F and T' to also denote the set of open facilities and the partial Steiner tree on the open facilities respectively. As argued in Section 6.5.1, the tree S^* can be extended to yield a Steiner tree on the components of T' of cost at most $S^* + C^* + \sum_{j \in D'} c_{\sigma(j)j}$. So the total cost of building an approximate Steiner tree on the open facilities is at most,

$$T' + \rho_{ST} \sum_{j \in D'} c_{\sigma(j)j} + \rho_{ST}(S^* + C^*). \tag{4}$$

Suppose the solution (x, y, z) opens exactly k' facilities, i.e., $\sum_{i \neq v} y_i = k'$. Then, since $\big(\alpha^{(1)}, \beta^{(1)}, \theta^{(1)}, \lambda\big)$ is a feasible solution to the dual of the connected k-median LP, using Lemma 6.5.6 we get that, $7(F + k'\lambda) + C + T' + 2\sum_{j \in D'} c_{\sigma(j)j} \leq 7\alpha_j^{(1)} \implies 7F + C + T' + 2\sum_{j \in D'} c_{\sigma(j)j} \leq 7\big(\sum_j \alpha_j^{(1)} - k'\lambda\big) \leq 7 \cdot OPT_k$, so by (4) we get that the total cost is at most $(7 + \rho_{ST}) \cdot \mathcal{O}^*$. We will always include the middle term in (4), or something that upper bounds it, in the cost of our partial solution which only has a partial tree on the open facilities. So if we show that the resulting cost is within some factor of OPT_k, then (4) shows that we can complete the Steiner tree on the open facilities and losing only an additive factor of $\rho_{ST} \cdot \mathcal{O}^*$. Using the framework developed in Section 7.3, we will obtain a partial solution that opens k facilities, and has net cost (where we include the additional term mentioned above) at most $14 \cdot OPT_k$. This will give a $(14 + \rho_{ST})$-approximation algorithm for the connected k-median problem.

If the algorithm opens at most k' facilities when $\lambda = 0$, then the net cost (including the term $2\sum_{j \in D'} c_{\sigma(j)j}$) is at most $7\sum_j \alpha_j^{(1)} \leq 7 \cdot OPT_k$ since $\big(\alpha^{(1)}, \beta^{(1)}, \theta^{(1)}, 0\big)$ is a feasible connected k-median dual solution. So we get a solution of cost at most $(7 + \rho_{ST}) \cdot \mathcal{O}^*$.

So suppose that at $\lambda = 0$ the algorithm opens more than k' facilities. When $\lambda \geq |\mathcal{D}| \max_j c_{vj}$, the algorithm will connect all demands to v and not open any other facility. So, by doing a bisection search in this range, we can find in polynomial time two (partial) primal solutions, one opening $k_1 < k'$ facilities and the other opening $k_2 > k'$ facilities, such that both the solutions may be obtained by running the ConFL algorithm with a single value $\lambda = \lambda_0$, depending on how we break ties between events in the dual-ascent process.

Let (x_1, y_1, z_1) and (x_2, y_2, z_2) be the two solutions obtained at $\lambda = \lambda_0$, and $\big(\alpha^{(1)}, \beta^{(1)}, \theta^{(1)}\big)$ be the common dual solution. Let (F_1, C_1, T'_1) and (F_2, C_2, T'_2) denote the cost of the solutions (x_1, y_1, z_1) and (x_2, y_2, z_2) respectively. A *convex combination* of the two solutions yields a fractional solution (x, y, z) that opens exactly k' facilities. Let $ak_1 + bk_2 = k', a + b = 1$. To avoid cumbersome notation, let A denote the quantity $2\sum_{j \in D'} c_{\sigma(j)j}$ in the solution (x_1, y_1, z_1) and let B denote the corresponding quantity in (x_2, y_2, z_2). Then,

$$7(aF_1 + bF_2) + (aC_1 + bC_2) + (aT'_1 + bT'_2) + aA + bB \\ \leq 7\Big(\sum_j \alpha_j^{(1)} - k'\lambda\Big) \leq 7 \cdot OPT_k. \quad (5)$$

We round (x, y, z) to get a solution that opens at most k facilities (including v) losing a factor of at most 2.

If $a \geq \frac{1}{2}$ we take the solution (x_1, y_1, z_1) and from (5) we get that $F_1 + C_1 + T'_1 + A \leq 14 \cdot OPT_k$.

Otherwise we open a subset of the facilities opened by (x_2, y_2, z_2) and get a solution of assignment cost at most $2(aC_1 + bC_2)$. Call a facility opened in (x_1, y_1, z_1) a *small* facility and a facility opened in (x_2, y_2, z_2) a *large* facility. For each small facility we consider the large facility closest to it. Let N be this set of large facilities. If $|N| < k_1$ we arbitrarily add large facilities to N till $|N| = k_1$. We open all the facilities in N. We also randomly pick a set of $k' - k_1$ large facilities not in N, and open these. Note that each such facility is opened with probability $(k' - k_1)/(k_2 - k_1) = b$. We also add edges of T'_2 corresponding to the open facilities.

For a demand j, let i_1, i_2 denote the small and large facilities to which it is assigned respectively. Let i_3 be the large facility nearest to i_1. Note that i_3 is always opened. We assign j to i_2 if it is open and to i_3 otherwise. Since $c_{i_3 j} \leq c_{i_1 j} + c_{i_1 i_3} \leq c_{i_1 j} + c_{i_1 i_2} \leq 2c_{i_1 j} + c_{i_2 j}$ and $a < b$, the expected assignment cost is at most, $bc_{i_2 j} + ac_{i_3 j} \leq 2(ac_{i_1 j} + bc_{i_2 j})$. So the total assignment cost is at most $2(aC_1 + bC_2)$. From (5),

$F_2 + 2(aC_1 + bC_2) + T_2' + B \leq 14 \cdot OPT_k$.

Completing the Steiner tree on the open facilities costs an additional $\rho_{ST}(S^* + C^*)$ factor, so the total cost is at most $(14 + \rho_{ST}) \cdot \mathcal{O}^*$.

Theorem 7.4.1 *Taking $\rho_{ST} = 1.55$, the above algorithm is a 15.55-approximation algorithm for the Connected k-Median problem.*

If we use the algorithm of [2, 26] with $\rho_{ST} = 2$, then we also get a bound the integrality gap.

Corollary 7.4.2 *The integrality gap of the Connected k-Median linear program is at most 16.*

7.5 The *k*-Median Problem with Service Installation Costs

We now consider the k-median version of facility location with service installation costs (FLSIC) introduced in Chapter 5. Recall that in this problem, we have a set of facilities $\mathcal{F}$, a set of clients $\mathcal{D}$, and a set of services $\mathcal{S}$. Each client requests a specific service in $\mathcal{S}$, and has to be assigned to an open facility on which that service is installed. Incurring a service l on facility i incurs a service installation cost of f_i^l, and we have to decide which facilities to open, which services to install on each open facility, and how to assign the clients to the open facilities, so as to minimize the total facility opening, service installation and client assignment costs. In the k-median version, at most k facilities may be opened.

The k-median version of FLSIC generalizes the k-facility location problem, where there is only one service type, and also is interesting from a clustering perspective. Most clustering objective functions insist that each data point be assigned to a single cluster. In the classical k-median problem, each point has to be assigned to a single median or center, and the cluster quality is measured by looking at the deviation or distance of each data point from its assigned center. If the goal of such a clustering is to get a good, compact representation of the data so as to infer trends and patterns in the data, then insisting that a data point be assigned to one cluster only, might be too restrictive in some settings. For example, a customer transaction on an online shopping site is a data object with multiple attributes. Each attribute could represent a different category of items bought like books, clothing, electronics, etc., and a good summary of the data should contain a clustering for each of these categories. So an object would lie in multiple clusters — a books cluster based on the genre of books bought, a clothing cluster specifying the type of clothes, and so on. The k-median version of FLSIC where there are no facility opening costs and one can open a facility at any location, can be used to model a clustering problem where a data point may be assigned to multiple clusters: given some points with multiple attributes (services) located in a metric space, we want to choose k of these as centers/medians, allot attributes to each center paying a cost per attribute allotted, and assign each attribute of every point to a center to which that attribute is allotted. The cost of the clustering (inversely proportional to its quality) is the total number of attributes allotted plus the sum of the distances from each point-attribute to its assigned center.

The k-median problem with service installation costs (KSIC) adds the constraint $\sum_i y_i \leq k$ to the linear program (FLS-P). The objective function of the dual (FLS-D) gets modified to max $\sum_j \alpha_j - k\lambda$ and constraint (2) changes to $\sum_j \beta_{ij} \leq f_i + \lambda$. Let (KP) and (KD) be the modified primal and dual programs and OPT_k be the common optimal value. We use the primal-dual algorithm from Section 5.4 to approximate KSIC to within a factor of $\frac{5(\sqrt{13}+1)}{2} \approx 11.52$ of the optimal when the installation cost f_i^l depends only on the service type l and not on i.

Again, following the outline in Section 7.3, we will try out different values of λ and for each value of λ, run the primal-dual algorithm with the facility costs modified to $f_i + \lambda$. Suppose the algorithm returns a primal solution of cost (O, I, C) that opens k facilities for some value of λ, and a dual solution (α, β, θ). Here O, I, C denote respectively the facility opening cost with the original costs f_i, the service installation cost, and the client assignment cost. By a now familiar argument, using Corollary 5.4.11, this shows that we have a solution of cost at most $O + I + C <$

$5O + I + C \le 5\big(\sum_j \alpha_j - k\lambda\big) \le 5 \cdot OPT_k$. By the same argument, if the algorithm opens at most k facilities when $\lambda = 0$, then the cost of this solution is at most $5 \cdot OPT_k$.

Recall that the primal-dual algorithm in Section 5.4 requires an ordering $\mathcal{O}$ of the facilities such that if i comes before i' in this ordering then for any service l, $f_i^l \le f_{i'}^l$. Since the service cost f_i^l does not depend on i, any ordering $\mathcal{O}$ can be used to order the facilities. We will consider the ordering $\mathcal{O}$ where the tentatively opened facilities come first in the order in which they were tentatively opened, followed by the remaining facilities in an arbitrary order. Note that since $\mathcal{O}$ is specified by the order in which events happen in the dual ascent process, the facilities that we open in step II of the algorithm, and hence the primal solution constructed, depends on the order in which events happen, and in particular, on how ties get broken between events that happen at the same time. The dual solution constructed however, is independent of the tie-breaking rule used as in the JV algorithm.

Suppose the algorithm opens more than k facilities at $\lambda = 0$. If $\lambda \ge |\mathcal{D}| \max_{ij} c_{ij} + |\mathcal{S}| \max_{il} f_i^l$, the algorithm will open just one facility, install all services on that facility, and assign all demands to it. We perform a bisection search to find two primal solutions, one opening $k_1 < k$ facilities and the other opening $k_2 > k$ facilities, both corresponding to a single dual solution obtained at $\lambda = \lambda_0$. Let (x_1, y_1) and (x_2, y_2) be these two primal solutions with costs (O_1, I_1, C_1), (O_2, I_2, C_2) respectively, and (α, β, θ) be the common dual solution. Let $(x, y) = a(x_1, y_1) + b(x_2, y_2)$ be the convex combination with a and b such that $ak_1 + bk_2 = k, a + b = 1$. We have

$$5(aO_1 + bO_2) + (aI_1 + bI_2) + (aC_1 + bC_2) \le 5\Big(\sum_j \alpha_j - k\lambda\Big) \le 5 \cdot OPT_k. \qquad (6)$$

We show how to round (x, y). If $a \ge \frac{\sqrt{13}-1}{6}$, then we take the solution (x_1, y_1) incurring a cost of at most $\frac{5}{a} \cdot OPT_k = \frac{5(\sqrt{13}+1)}{2} \cdot OPT_k$.

Otherwise, we use a rounding procedure similar to the LP rounding algorithm. Call a facility opened in (x_1, y_1) a *small* facility, and a facility opened in (x_2, y_2) a *large* facility. For simplicity we assume that these two solutions do not share a common open facility; we treat such a facility as two distinct facilities. For a demand

j, let $i_1(j), i_2(j)$ be the small and large facilities to which j is assigned, and $F_j = \{i_1(j), i_2(j)\}$. We first form some clusters as in the algorithm in Section 5.5 but using a different center selection rule. The algorithm is as follows.

K1. For every service type l, we consider the clients in G_l and cluster the facilities on which service l is installed. Pick $j \in G_l$ with smallest $c_{i_1(j)j} + c_{i_2(j)j}$ value and form a cluster around j consisting of the facilities in F_j. We make j the *representative* of every client $k \in G_l$ (including j) that is served (fractionally) by some facility in F_j, remove each such client from G_l, and recurse on the remaining set of clients until G_l becomes empty. This gives a set of cluster centers D_l for each service l. For a client $k \notin D_l$ let $\sigma(k)$ denote its representative cluster center in D_l.

K2. Let $D = \bigcup_l D_l$. We cannot open a facility in every cluster since different clusters could share the same fractional facility weight (y_i) if the cluster centers request different services. Say that $j, k \in D$ are dependent if $F_j \cap F_k \neq \phi$. Consider clients in D in increasing order of $c_{i_1(j)j} + c_{i_2(j)j}$ and greedily pick a maximal independent subset D'. For every client $k \in D \setminus D'$, there is some $j \in D'$ that was picked before k such that j and k are dependent. Call j the neighbor of k and denote it by $\mathsf{nbr}(k)$. For convenience, we set $\mathsf{nbr}(j) = j$.

K3. We now match each small facility with a large facility. For each cluster centered at $j \in D'$ we match $i_1(j)$ with $i_2(j)$. The remaining small facilities are matched arbitrarily with distinct unmatched large facilities. With probability a we open all the small facilities, and with probability b we open all the matched large facilities. We open k_1 facilities this way and ensure that each cluster centered at $j \in D'$ contains an open facility.

K4. Next, we randomly choose a set of $k - k_1$ unmatched large facilities and open all of these. Note that each unmatched large facility is opened with probability $(k - k_1)/(k_2 - k_1) = b$.

K5. For each open facility we install all services that are installed on it in the fractional solution (x, y).

K6. For every $j \in D \setminus D'$ if neither $i_1(j)$ nor $i_2(j)$ is opened, we install service $g(j)$ on the facility opened from $F_{\mathsf{nbr}(j)}$.

K7. Demand j is assigned to the nearest open facility at which service $g(j)$ is installed.

7.5.1 Analysis

Lemma 7.5.1 *The expected cost of opening facilities is at most $aO_1 + bO_2$. The expected cost of installing services is at most $(1 + ab)(aI_1 + bI_2)$.*

Proof : Each small facility is opened with probability a, and each large facility is opened with probability b, so the expected facility opening cost is $aO_1 + bO_2$. The expected cost of installing services in step K5 is $\sum_i \Pr[i \text{ is opened }] \sum_{l:y_i^l>0} f_i^l$. Since (x_1, y_1), (x_2, y_2) are integer solutions and we assume that no facility is opened in both of these solutions, if $y_i^l > 0$ then $y_i^l = y_i$, and if i is a small facility then $y_i^l = y_i = a$, otherwise $y_i^l = y_i = b$. So the cost of installing facilities in step K5 is

$$a \cdot \sum_{\substack{\text{small} \\ \text{facility } i}} \sum_{l:y_i^l>0} f_i^l + b \cdot \sum_{\substack{\text{large} \\ \text{facility } i}} \sum_{l:y_i^l>0} f_i^l = aI_1 + bI_2.$$

In step K6, the probability that we install service $g(j)$ due to client j, is the probability that none of the facilities in F_j is open, which is at most ab (it is 0 if $i_2(j)$ is matched and ab otherwise). So the expected cost of installing services in step K6 is at most $ab \sum_{j\in D\setminus D'} f^{g(j)} \leq ab(aI_1 + bI_2)$, since any two clients in D_l have disjoint clusters. ■

Lemma 7.5.2 *The expected service cost of any client j is at most* $\max(1 + 3a, 1 + b)(ac_{i_1(j)j} + bc_{i_2(j)j})$.

Proof : Let $i = i_1(j), i' = i_2(j)$ and $C_j = ac_{i_1(j)j} + bc_{i_2(j)j}$. If i' is a matched large facility, exactly one of i and i' is open and the service cost is bounded by C_j. This takes care of the case when $j \in D'$. Otherwise if i' is not matched, there are 2 cases to consider.

Case 1: $j \in D \setminus D'$. Let $k = \mathsf{nbr}(j) \in D'$. Then $F_j \cap F_k = \{i = i_1(j) = i_1(k)\}$. Either i or $i_2(k)$ must be open; we assign j to i' if it is open, otherwise to i if it is open and otherwise to $i_2(k)$. Note that service $g(j)$ is installed on the facility to which j is assigned. The expected cost under this possibly suboptimal assignment is $bc_{i'j} + a(ac_{ij} + bc_{i_2(k)j})$ which is at most

$$bc_{i'j} + a\big(ac_{ij} + b(c_{i_2(k)k} + c_{ik} + c_{ij})\big) \le bc_{i'j} + a\big(c_{ij} + b(c_{ij} + c_{i'j})\big) \le \max(1+a, 1+b)C_j,$$

where the first inequality follows since we picked k before j in step K2.

Case 2: $j \notin D$. Let $j' = \sigma(j)$. If $j' \in D'$, then as in Case 1, we can argue that the assignment cost of j is at most $\max(1+a, 1+b)C_j$ since service $g(j) = g(j')$ is installed on either $i_1(j')$ or $i_2(j')$, and $c_{i_1(j')j'} + c_{i_2(j')j'} \le c_{ij} + c_{i'j}$. So let $j' \in D \setminus D'$ and $k = \mathsf{nbr}(j)$. We know that

$$c_{i_1(k)k} + c_{i_2(k)k} \le c_{i_1(j')j'} + c_{i_2(j')j'} \le c_{ij} + c_{i'j}, \tag{7}$$

where the first inequality follows since $k = \mathsf{nbr}(j')$ and the second since $j' = \sigma(j)$. Let $A = F_j \cap F_{j'} \neq \phi$, and $B = F_{j'} \cap F_k \neq \phi$. There are three sub-cases.

(a) $A = \{i\}$. We consider assigning j first to i', then to i, then to facility $i_2(j')$, and lastly if none of these facilities is open, to the open facility in F_k. The expected cost is at most $bc_{i'j} + a^2 c_{ij} + ab(pc_{i_2(j')j} + qd)$, where $p = \Pr[i_2(j') \text{ is open}|i, i'$ are not open$]$, $q = 1 - p$ and d is the expected distance to the facility opened from F_k conditioned on the event that i, i' and $i_2(j')$ are not open. Note that p, q and d will depend on whether $i_2(j')$ is matched or not. If $i_2(j')$ is matched, then $p = 1$ and using (7), we can substitute $c_{i_2(j')j} \le c_{ij} + c_{i_1(j')j'} + c_{i_2(j')j'} \le 2c_{ij} + c_{i'j}$. If $i_2(j')$ is not matched, then we have $B = \{i = i_1(j) = i_1(k)\}$. We again substitute for $c_{i_2(j')j}$ and bound $d = c_{i_2(k)j}$ by $c_{ij} + c_{i_1(k)k} + c_{i_2(k)k} \le 2c_{ij} + c_{i'j}$. In either case, we get that the expected cost is bounded by $\max(1+a, 1+b)C_j$.

(b) $A = \{i'\}$. We assign j to i' if it is open, otherwise to i if it is open, and otherwise to the open facility $i_2(k)$ in F_k. Here we know that $B = \{i_1(j') = i_1(k)\}$ (since $i_2(j') = i'$ is not matched) and if i is not open, then facility $i_1(k)$ is not open. The expected assignment cost is $bc_{i'j} + a^2 c_{ij} + abc_{i_2(k)j}$. We can bound $c_{i_2(k)j}$ by

$$c_{i'j} + c_{i_2(j')j'} + c_{i_1(j')j'} + c_{i_1(k)k} + c_{i_2(k)k} \leq 2c_{ij} + 3c_{i'j}.$$

Substituting, we get that the expected cost is at most $\max(1 + 3a, 1 + b)C_j$.

(c) $A = \{i, i'\}$. Then it must be that $B = \{i = i_1(j') = i_1(k)\}$ since i' is not matched. We consider assigning j first to i', then to i, and then to $i_2(k)$. The cost is bounded is as above, except that we now have $c_{i_2(k)j} \leq 2c_{ij} + c_{i'j}$, so the cost is at most $\max(1 + a, 1 + b)C_j$.

So in every case, the assignment cost of j is bounded by $\max(1+3a, 1+b)(ac_{i_1(j)j} + bc_{i_2(j)j})$. ■

Theorem 7.5.3 *There is an* 11.52*-approximation algorithm for the* k*-Median problem with Service Installation Costs.*

Proof : If $a \geq \frac{\sqrt{13}-1}{6}$, then we take solution (x_1, y_1) incurring a cost of at most $\frac{5}{a} \cdot OPT_k = \frac{5(\sqrt{13}+1)}{2} \cdot OPT_k$. Otherwise by Lemma 7.5.1 and Lemma 7.5.2, we get that the expected total cost of the solution returned by the rounding procedure is at most is at most $(aO_1 + bO_2) + (1 + ab)(aI_1 + bI_2) + \max(1 + 3a, 1 + b)(aC_1 + bC_2)$ which is at most $\frac{5(\sqrt{13}+1)}{2} \cdot OPT_k$ by (6) and since $0 \leq a \leq \frac{\sqrt{13}-1}{6}$. Thus we get an $\frac{5(\sqrt{13}+1)}{2} \approx 11.52$-approximation algorithm. ■

Bibliography

[1] A. Ageev and M. Sviridenko. Pipage rounding: a new method of constructing algorithms with proven performance guarantee. *Journal of Combinatorial Optimization*. To appear.

[2] A. Agrawal, P. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized Steiner problem on networks. *SIAM Journal on Computing*, 24(3):440–456, 1995.

[3] M. Akgül. *Topics in relaxation and ellipsoidal methods.* Pitman Advanced Publishing Program, London, 1997.

[4] N. Aksoy and S. Erenguc. Multi-item inventory models with coordinated replenishments: a survey. *International Journal of Operations & Production Management*, 8:63–73, 1988.

[5] M. Andrews and L. Zhang. The access network design problem. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, pages 40–49, 1998.

[6] A. Archer, R. Rajagopalan and D. Shmoys. Lagrangian relaxation for the k-median problem: new insights and continuity properties. In *Proceedings of the 11th Annual European Symposium on Algorithms*, pages 31–42, 2003

[7] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for k-median and facility location problems. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pages 21–29, 2001.

[8] I. Baev and R. Rajaraman. Approximation algorithms for data placement in arbitrary networks. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 661–670, 2001.

[9] M.L. Balinski. Integer programming: methods, uses, computation. *Management Science*, 12(3):253–313, 1965.

[10] Y. Bartal, M. Charikar and D. Raz. Approximating min-sum k-clustering in metric spaces. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pages 11–20, 2001.

[11] E. M. L. Beale. On minimizing a convex function subject to linear inequalities. *Journal of the Royal Statistical Society, Series B*, 17:173–184; discussion 194–203, 1955.

[12] D. Bertsimas and M. Sim. Robust discrete optimization and network flows. Mathematical Programming, Series B, 98:49–71, 2003.

[13] J. R. Birge and F. V. Louveaux. *Introduction to Stochastic Programming*. Springer-Verlag, New York, 1997.

[14] J. Borwein and A. S. Lewis. *Convex Analysis and Nonlinear Optimization*. Springer-Verlag, New York, 2000.

[15] M. Charikar and S. Guha. Improved combinatorial algorithms for the facility location and k-median problems. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 378–388, 1999.

[16] M. Charikar, S. Guha, É. Tardos, and D. B. Shmoys. A constant-factor approximation algorithm for the k-median problem. *Journal of Computer and System Sciences*, 65(1):129–149, 2002.

[17] M. Charikar, S. Khuller, D. Mount, and G. Narasimhan. Algorithms for facility location problems with outliers. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 642–660, 2001.

[18] F. Chudak and D. Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. *SIAM Journal on Computing*, 33(1):1–25, 2003.

[19] V. Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4:233–235, 1979.

[20] COSP. Stochastic programming community home page. World Wide Web, `http://stoprog.org`, 2002.

[21] G. B. Dantzig. Linear programming under uncertainty. *Management Science*, 1:197–206, 1955.

[22] S. Dye, L. Stougie, and A. Tomasgard. The stochastic single node service provision problem. COSOR-Memorandum 99-13, Dept. of Mathematics and Computer Sc., Eindhoven, Tech. Univ., Eindhoven, 1999. Also in *Stochastic Programming E-Print Series*, February 2002.

[23] M. Dyer, R. Kannan, and L. Stougie. A simple randomised algorithm for convex optimisation. SPOR-Report 2002-05, Dept. of Mathematics and Computer Science, Eindhoven Technical University, Eindhoven, 2002.

[24] M. Dyer and L. Stougie. Computational complexity of stochastic programming problems. SPOR-Report 2003-09, Dept. of Mathematics and Computer Science, Eindhoven Technical University, Eindhoven, 2003.

[25] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45:634–652, 1998.

[26] M. X. Goemans and D. P. Williamson. The primal-dual method for approximation algorithms and its application to network design problems. In D. S. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*, chapter 4, pages 144–191. PWS Publishing Company, 1997.

[27] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization.* Springer-Verlag, New York, 1988.

[28] S. Guha and S. Khuller. Connected facility location problems. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 40:179–190, 1997.

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

[30] S. Guha, A. Meyerson, and K. Munagala. A constant factor approximation for the single sink edge installation problems. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pages 383–388, 2001.

[31] S. Guha, A. Meyerson, and K. Munagala. Hierarchical placement and network design problems. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, pages 603–612, 2000.

[32] A. Gupta, J. Kleinberg, A. Kumar, R. Rastogi, and B. Yener. Provisioning a virtual private network: A network design problem for multicommodity flow. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pages 389–398, 2001.

[33] A. Gupta, A. Kumar, M. Pál, and T. Roughgarden. Approximation via cost sharing: a simple approximation algorithm for the multicommodity rent-or-buy problem. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 605–615, 2003.

[34] A. Gupta, A. Kumar, and T. Roughgarden. Simple and better approximation algorithms for network design. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 365–372, 2003.

[35] A. Gupta, M. Pál, R. Ravi, and A. Sinha. Boosted sampling: approximation algorithms for stochastic optimization. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 417–426, 2004.

[36] G. H. Hardy, J. E. Littlewood, and G. Pólya. *Inequalities*. Cambridge University Press, 1952.

[37] R. Hassin, R. Ravi, and F. S. Selman. Approximation algorithms for a capacitated network design problem. In *Proceedings of the 3rd International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, pages 167–176, 2000.

[38] D. S. Hochbaum. Heuristics for the fixed cost median problem. *Mathematical Programming*, 22(2):148–162, 1982.

[39] N. Immorlica, D. Karger, M. Minkoff, and V. Mirrokni. On the costs and benefits of procrastination: approximation algorithms for stochastic combinatorial optimization problems. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 684–693, 2004.

[40] K. Jain, M. Mahdian, E. Markakis, A. Saberi, and V. Vazirani. Greedy facility location algorithms analyzed using dual-fitting with factor-revealing LP. *Journal of the ACM* 50(6):795–824, 2003.

[41] K. Jain and V.V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation. *Journal of the ACM* 48(2):274–296, 2001. Preliminary version in *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 2–13, 1999.

[42] P. Kall and S. Wallace. *Stochastic Programming*. Wiley, Chichester, 1994.

[43] D. R. Karger and M. Minkoff. Building Steiner trees with incomplete global knowledge. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, pages 613–623, 2000.

[44] S. Khuller and A. Zhu. The general Steiner tree-star problem. *Information Processing Letters*, 84(4):215–220, 2002.

[45] Tae Ung Kim, Timothy J. Lowe, Arie Tamir, and James E. Ward. On the location of a tree-shaped facility. *Networks*, 28(3):167–175, 1996.

[46] A. J. Kleywegt, A. Shapiro, and T. Homem-De-Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal of Optimization*, 12:479–502, 2001.

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

[48] C. Krick, H. Räcke, and M. Westermann. Approximation algorithms for data management in networks. In *Proceedings of the 13th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 237–246, 2001.

[49] A. Kumar, A. Gupta, and T. Roughgarden. A constant-factor approximation algorithm for the multicommodity rent-or-buy problem. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, pages 333–342, 2002.

[50] A. Kumar, R. Rastogi, A. Silberschatz, and B. Yener. Algorithms for provisioning virtual private networks in the hose model. In *Proceedings of the Annual ACM Conference of the Special Interest Group on Data Communication*, pages 135–146, 2001.

[51] M. Labbé, G. Laporte, I. Rodrígues Martín, and J. J. Salazar González. The median cycle problem. Technical Report 2001/12, Department of Operations Research and Multicriteria Decision Aid at Université Libre de Bruxelles, 2001.

[52] Y. Lee, S. Y. Chiu, and J. Ryan. A branch and cut algorithm for a Steiner tree-star problem. *INFORMS Journal on Computing*, 8(3):194–201, 1996.

[53] R. Levi, R. Roundy, and D. Shmoys. Primal-dual algorithms for deterministic inventory problems. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 353–362, 2004.

[54] J. H. Lin and J. S. Vitter. ϵ-approximations with minimum packing constraint violation. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 771–782, 1992.

[55] J. Linderoth, A. Shapiro, and R. K. Wright. The empirical behavior of sampling methods for stochastic programming. *Annals of Operations Research*. To appear.

[56] M. Mahdian, Y. Ye, and J. Zhang. Improved approximation algorithms for metric facility location. In *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, pages 229–242, 2002.

[57] R.R. Mettu and C.G. Plaxton. The online median problem. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, pages 339–348, 2000.

[58] P. Mirchandani and R. Francis, editors. *Discrete Location Theory*. John Wiley and Sons, Inc., New York, 1990.

[59] P. Raghavan and C.D. Thompson. Randomized rounding. *Combinatorica*, 7:365–374, 1987.

[60] R. Ravi and F. S. Selman. Approximation algorithms for the traveling purchaser problem and its variants in network design. In *Proceedings of the 7th Annual European Symposium on Algorithms*, pages 29–40, 1999.

[61] R. Ravi and A. Sinha. Hedging uncertainty: approximation algorithms for stochastic optimization problems. In *Proceedings of the 10th International Conference on Integer Programming and Combinatorial Optimization*, pages 101–115, 2004.

[62] R. Ravi and A. Sinha. Integrated logistics: Approximation algorithms combining facility location and network design. In *Proceedings of the 9th International Conference on Integer Programming and Combinatorial Optimization*, pages 212–229, 2002.

[63] R. Ravi and A. Sinha. Multicommodity facility location. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 335–342, 2004.

[64] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 475–484, 1997.

[65] G. Robins and A. Zelikovsky. Improved steiner tree approximation in graphs. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 770–779, 2000.

[66] A. Ruszczynski and A. Shapiro, editors. *Stochastic Programming*, volume 10 of *Handbooks in Operations Research and Management Science*, North-Holland, Amsterdam, 2003.

[67] A. Shapiro. Monte Carlo sampling methods. In A. Ruszczynski and A. Shapiro, editors, *Stochastic Programming*, volume 10 of *Handbooks in Operations Research and Management Science*, North-Holland, Amsterdam, 2003.

[68] D. B. Shmoys. Approximation algorithms for facility location problems. In *Proceedings of the 3rd International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, pages 27–33, 2000.

[69] D. B. Shmoys and C. Swamy. Stochastic optimization is (almost) as easy as deterministic optimization. To appear in *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, 2004.

[70] D. B. Shmoys, C. Swamy, and R. Levi. Facility location with service installation costs. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1081–1090, 2004.

[71] D. B. Shmoys, É. Tardos, and K. I. Aardal. Approximation algorithms for facility location problems. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 265–274, 1997.

[72] L. Stougie and M. H. van der Vlerk. Approximation in stochastic integer programming. SOM Research Report 03A14, University of Groningen, 2003. Also in *Stochastic Programming E-Print Series*, 2003-12.

[73] M. Sviridenko. An improved approximation algorithm for the metric uncapacitated facility location problem. In *Proceedings of the 9th International Conference on Integer Programming and Combinatorial Optimization*, pages 240–257, 2002.

[74] C. Swamy. A note on the data placement problem. Unpublished manuscript, 2003.

[75] C. Swamy and A. Kumar. Primal-dual algorithms for connected facility location problems. *Algorithmica*. To appear. Preliminary version in *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, pages 256–269, 2002.

[76] K. Talwar. The single-sink buy-at-bulk LP has constant integrality gap. In *Proceedings of the 9th International Conference on Integer Programming and Combinatorial Optimization*, pages 475–486, 2002.

[77] B. Verweij, S. Ahmed, A. J. Kleywegt, G. L. Nemhauser, and A. Shapiro. The sample average approximation method applied to stochastic routing problems: a computational study. *Computational Optimization and Applications*, 24:289–333, 2003.

[78] M. H. van der Vlerk. Stochastic programming bibliography. World Wide Web, `http://mally.eco.rug.nl/spbib.html`, 1996–2004.

[79] R. J.-B. Wets. Stochastic programming. In G. L. Nemhauser, A. H. G. Rinnooy Kan and M. J. Todd, editors, *Optimization*, volume 1 of *Handbooks in Operations Research and Management Science*, pages 573–629, North-Holland, Amsterdam, 1989.

[80] D. P. Williamson. The primal-dual method for approximation algorithms. *Mathematical Programming, Series B*, 91(3):447–478, 2002.