Building Networks in the Face of Uncertainty

by

Shubham Gupta

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

Waterloo, Ontario, Canada, 2011

Abstract

The subject of this thesis is to study approximation algorithms for some network design problems in face of uncertainty. We consider two widely studied models of handling uncertainties - Robust Optimization and Stochastic Optimization.

We study a robust version of the well studied Uncapacitated Facility Location Problem (UFLP). In this version, once the set of facilities to be opened is decided, an adversary may close at most β facilities. The clients must then be assigned to the remaining open facilities. The performance of a solution is measured by the worst possible set of facilities that the adversary may close. We introduce a novel LP for the problem, and provide an LP rounding algorithm when all facilities have same opening costs.

We also study the 2-stage Stochastic version of the Steiner Tree Problem. In this version, the set of terminals to be covered is not known in advance. Instead, a probability distribution over the possible sets of terminals is known. One is allowed to build a *partial solution* in the first stage a low cost, and when the exact scenario to be covered becomes known in the second stage, one is allowed to extend the solution by building a *recourse network*, albeit at higher cost. The aim is to construct a solution of low cost in expectation. We provide an LP rounding algorithm for this problem that beats the current best known LP rounding based approximation algorithm.

Acknowledgements

I would like to express my heartfelt thanks and gratitude to my advisor, Dr. Chaitanya Swamy, for his continuous moral and intellectual support. He has been a source of motivation for me during these two years, and our weekly meetings have been a very intellectually stimulating experience for me. His wisdom and enthusiasm in his work has been inspirational for me. I am greatly thankful to Dr. Joseph Cheriyan for his constant support and guidance during the course of my stay here.

I am thankful to the readers of my thesis - Dr. Jochen Könemann and Dr. Joseph Cheriyan, for their careful review of my thesis, and for pointing out the corrections. In particular, I am very thankful to my advisor Dr. Swamy for his careful proofreading and very detailed comments about the thesis.

Next, I would like to thank my fellow graduate students at C & O, with whom I had many stimulating discussions. I would also like to thank my new friends in Waterloo for making my stay here enjoyable and my old friends for many years of friendship and support. In particular, I am grateful to my office-mate, Xili Zhang, and roommates Akhilesh, Jalaj and Sarvagya for providing help, advice and support during easy and not so easy times allike.

Last but not the least, I would like to express my love and gratitude towards my parents for their love and support throughout the years. This thesis is dedicated to them.

Table of Contents

List of Figures

Chapter 1

Introduction

Uncertainty is an integral part of most practical problems, and optimization problems are no different. Uncertainty may arise in optimization problems for various reasons, such as unpredictable information revealed in the future or inherent fluctuations caused by noise. [38]. The need to model uncertainty in a mathematical framework arises frequently in many areas such as transportation models, financial instruments and network design. Some examples where decisions need to be taken in face of incomplete information include problems in portfolio selection, the task of opening facilities to cater to some clients whose exact locations are unknown initially, managing operation of water reservoirs that must be able to reasonably distribute water for drinking purposes, irrigation, etc, but the exact demands for each task and the location of such requirements may not be known precisely in the beginning, and so on.

A common theme in such models is the need to make decisions without having their full effect known in advance. Not only the uncertainty makes the task challenging in terms of developing clean algorithms, but introduction of uncertainty may make an otherwise easy problem hard to solve, in lieu of complexity theory. This means that while a deterministic problem may be solvable in polynomial time, its counterpart, with some notion of uncertainty, may be harder in terms of complexity (e.g. it may become *NP*-hard).

There are various options when it comes to measuring the *goodness* of a solution to a problem with uncertainties. *Robust Optimization* and *Stochastic Optimization* are two widely accepted notions of modeling a problem with uncertainties. Though both approaches cater to the basic need of dealing with uncertainties, the way they achieve this differs. While stochastic optimization models the uncertainties as a probability distribution over a possible set of scenarios (which capture the uncertainty), robust optimization

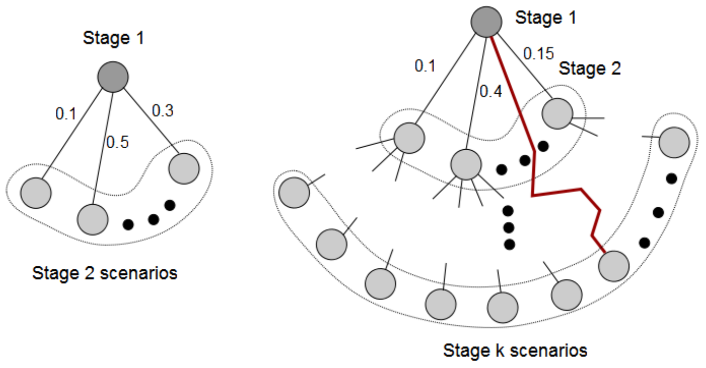

Figure 1.1: 2-stage and multistage stochastic optimization with recourse

develops a solution that would work for any possible scenario that occurs as a result of uncertainty [9].

1.1 Stochastic Optimization

Stochastic optimization provides a means to handle uncertainty by modeling it by a probability distribution over possible realizations of the actual data, called scenarios. [38]. The origins of stochastic optimization date back to the 1950's in works of Dantzig [15] and Beale [4].

A very widely used model of Stochastic optimization is the *multistage stochastic optimization with recourse* . In a 2-*stage stochastic optimization with recourse* model, the actual scenario that need to be serviced is not known in advance. Instead, a probability distribution over the possible set of scenarios is known in the *first stage*, based on which, one is allowed to *construct* a partial solution. The actual scenario is revealed in stage 2,. Further *recourse solution* must now be built to satisfy the requirements of the revealed scenario. The aim is to construct a solution of low expected total cost incurred. Note that, however, the prices of the components in this stage are higher as compared to the prices in the first stage. Hence there is a tradeoff between constructing a solution in first stage at lower cost but without complete information about the requirements, and waiting for the requirements to be revealed but having to build a solution at higher costs. In a general k stage process, the prices of the components increase with every stage, and a new information is received at every stage which gives a probability distribution over a restricted set of scenarios. One is allowed to buy components at each stage (at the prices of the stage), taking decisions based on the new knowledge available. Once the actual scenario gets revealed in the final stage, the solution built so far has to be extended to satisfy the revealed requirements.

Specification of the probability distribution is an important issue here. The set of scenarios can be finite or infinite. While it is not possible to list out a probability distribution on infinite scenarios explicitly unless expressed concisely, even the finite scenario model can be tricky. If the number of scenarios are not polynomial in other input parameters, then it may not be feasible to list out the distribution explicitly. In a *black box model*, the user is allowed to make polynomial amount of queries to a *black box*, and each such query returns a scenario drawn according to the probability distribution. These samples are then used to frame an approximation to the actual probability distribution. Another variant specifies the probability with which each component occurs in the scenario that is presented to the user in the second stage. This means that the probability of a scenario occurring is just the product of the probabilities of occurrence of the components in the scenario and the probabilities of non-occurrence of the components not in the scenario (assuming independence).

1.2 Robust Optimization

Robust optimization handles uncertainty by constructing a solution that works well for the *worst* possible scenario that may materialize. Unlike that in stochastic optimization, the aim is not to construct a solution which performs well in expectation, but a solution that performs well in the worst possible case. As noted in [9], the origins of robust optimization lies in the field of *robust control*, where the uncertainty is in form of perturbations in the parameters of the problem. Many optimization problems can be highly sensitive to such perturbations, leaving the computed solution highly infeasible, suboptimal or both. The idea of robustness is to develop solutions resilient to such sensitivity.

The uncertainty in such situations is not stochastic, but rather deterministic and set-based. One intends to immunize the solution against any realization of the uncertainty in a given set, instead of making it immune in some probabilistic sense to stochastic uncertainty [9]. Recently, study of robust optimization received interest from the theoretical community in an effort to understand computational tractability issues, among others. Works of Ben-Tal and Nemirovski (e.g., [6], [7], [8],[5]) and others like [16], [32] explore robust optimization versions of many problems in varied areas like portfolio optimization and SDP.

While we can define a multistage stochastic optimization problem, robust optimization does not normally happen in multiple stages. All information needed to compute a solution is known beforehand. The only thing that is not known is the actual scenario that will materialize. This allows for a two stage process. Given the set of possible scenarios, one is allowed to construct a *partial solution* to the problem at low costs. When the actual scenario is revealed, the costs of the components involved are also inflated. Thus, one is required to build a *recourse* solution to satisfy the scenario, but at a higher cost. The total cost is the sum of the costs of solutions constructed in first and second stages. If Ω is the set of scenarios, then the aim is (for a minimization problem):

$$\min\left(\text{(cost of first-stage partial solution)} + \max_{A\in\Omega} \text{ (cost of second stage recourse solution)}\right)$$

where costs may potentially be higher in the second stage. This translates to a single stage process if the costs in the second stage are not inflated. In this case, the problem of robust optimization becomes a min-max problem as there is no incentive of performing an action in the first stage. The aim of the optimization problem can then thus be viewed as construction of a mechanism (algorithm) $\mathcal{A}$, which takes the deterministic parts of the problem and a particular scenario from the set of all possible scenarios as input, and returns a solution to the problem with that scenario as output. For a minimization problem, the aim is:

$$\min\left(\max_{A\in\Omega}\mathcal{A}\left(\text{deterministic parts of problem}, A\right)\right)$$

1.3 Approximation Algorithms - Uncertainty in Network Design Problems

Many deterministic network design problems are well studied and understood in terms of approximation algorithms. It is natural to extend the problems to include uncertainty, and develop approximation algorithms for these. Recently, much work has been done in developing such models to include relevant notions of uncertainty, and constructing solutions which are *good* in terms of some defined notion of *goodness*. In light of the discussion above, one can develop both stochastic and robust optimization versions of such network design problems. Let us consider some such common problems informally, and their corresponding robust optimization and stochastic optimization versions.

1.3.1 Set Cover

The input to the deterministic problem is a set system $(U, \mathcal{F})$, where U is the universe of elements, and $\mathcal{F}$ is a family of sets from U. There is a (non-negative) weight associated with each set in the family $\mathcal{F}$. Given a set of special elements $K \subseteq U$ called *terminals*, the aim is to come up with a collection of sets from the family $\mathcal{F}$ of minimum weight, such that all the terminals are *covered* by this collection. It is *hard* to approximate this problem to a factor better than log of the number of terminals, under reasonable assumptions [17].

Introduction of uncertainty: Here, the uncertainty lies in the exact set of terminals that need to be covered. A scenario is a particular set of terminals that may need to be covered, but it is not known precisely which scenario will occur from the set of all possible scenarios Ω (Ω is supplied as an input to the problem). The robust optimization version of the problem seeks to find a mechanism, which, given a scenario S_i, returns a set cover C_i covering S_i, and the cost of the solution is judged by the cost of covering the most notorious scenario, i.e.

$$\min \left(\max_{S_i \in \Omega} (\text{ total weight of sets in } C_i \text{ }) \right)$$

In a 2-stage stochastic optimization version of the problem, the sets are costlier in second stage as compared to the first stage. In addition to Ω, a probability distribution over the scenarios in Ω is also supplied as input to the problem. The aim is to include some sets in the cover in first stage at lower weights (E_0), and to extend it by an additional collection of sets E_i (at higher weights) to satisfy the scenario S_i in the second stage. The intention is to minimize the expected cost of the solution:

$$\min \left((\text{total weight of sets in } E_0 \text{ }) + \mathbb{E}_{S_i \in \Omega} (\text{ total weight of sets in } E_i \text{ }) \right)$$

The exact approximation guarantees that can be obtained for these problems depend on the way the set Ω is made accessible, and also the probability distribution in case of the stochastic version. One is referred to [35] and [22] for further reading on this and related problems. In this thesis, we study the next two problems we describe here.

1.3.2 Robust Facility Location Problem

In the deterministic version of the *uncapacitated facility location* problem (UFLP), the input consists of a set of clients that need to satisfy their demands. Their demands are satisfied by facilities, and the input specifies a set of potential locations $\mathcal{F}$ where the facilities can be opened, and the cost of opening a facility at each location. There is an underlying metric which specifies the distance between clients and facilities, which also serves as the cost of serving a unit demand at the client location by the facility concerned. The aim is to open a set of facilities, and assign the clients to the open facilities, so that the total cost of opening facilities and serving of client demands by the open facilities is minimized.

Introduction of uncertainty: Here, the uncertainty is supplied by an adversary, that closes at most β facilities from the set of facilities F_0 the user decides to open. Hence, a scenario S_i is a set of at most β facility locations. The collection of all scenarios Ω is all possible subsets of the facility locations $\mathcal{F}$ of size at most β. The aim is to design an algorithm that opens a set of facilities, and specifies the assignment σ_i of clients to open facilities for every scenario S_i, so as to achieve:

$$\min\left((\text{cost of } F_0) + \max_{S_i \in \Omega}(\text{cost of assignment } \sigma_i \text{ of clients to facilities in } F_0 \setminus S_i)\right)$$

In this thesis, we study this robust version for UFLP and other variants of the facility location problems.

1.3.3 Stochastic Steiner Tree Problem

The very well studied deterministic version of the Steiner tree problem is as follows: We have a graph $G = (V, E)$, and a cost function that assigns non-negative costs to each edge of the graph. Additionally, a subset of vertices K (terminals) is provided as an input. The aim is to find a subgraph of the original graph of minimum cost, such that this subgraph is a tree that spans the terminals K.

Introduction of uncertainty: It is not known beforehand which set of terminals one needs to cover, i.e. the uncertainty lies in the specification of set K. A scenario is a set of terminals that need to be covered. In the 2-stage stochastic version, the collection of all possible sets of terminals (i.e. set of scenarios Ω) and a probability distribution over these terminal sets is provided as an input. One is allowed to construct a partial network E_0 in the first stage when only the set Ω and the probability distribution are known. Note that E_0 need not be a tree. In the second stage, the exact set of terminals $S_i \in \Omega$ to cover becomes known, and one must buy include additional set of edges E_i, such that $E_0 \cup E_i$ is indeed a Steiner tree on S_i. Care must be taken, however, that edges are costlier in second stage as compared to the first stage. The objective is:

$$\min\left((\text{cost of } E_0) + \mathbb{E}_{S_i \in \Omega}(\text{cost of } E_i)\right)$$

In this thesis, we study this stochastic version of the Steiner tree problem.

1.4 Outline of the Thesis

We provide an LP-rounding algorithm for the robust version of UFLP, that follows closely rounding algorithms for the deterministic versions of the capacitated FLP. In Chapter 2, we provide an overview of the rounding algorithms for the capacitated and uncapacitated deterministic versions of FLP, an understanding of which is essential for understanding the rounding algorithm for the robust version. In Chapter 3, we describe our rounding algorithm for the robust version of the uncapacitated facility location problem. In Chapter 4, we extend this notion of robustness to the k-median facility location problem, and explore promising directions towards solving this version. Finally, we present an improved rounding algorithm for stochastic Steiner tree problem in Chapter 5.

Chapter 2

LP Rounding Algorithms for Facility Location Problems

In the next chapter, we describe a rounding algorithm for the robust version of the uncapacitated facility location problem ($\beta RUFLP$). As we shall see later, $\beta RUFLP$ not only is an extension of the uncapacitated facility location problem ($UFLP$), but also has flavors from the capacitated facility location problem ($CFLP$). Our rounding algorithm for $\beta RUFLP$ derives inspiration from the rounding algorithms for $UFLP$ and $CFLP$. We, therefore, first present the rounding algorithms for $UFLP$ and $CFLP$ in this chapter. The algorithms we discuss in this chapter are the ones presented by Shmoys et al in [36].

2.1 Uncapacitated FLP

In this section, we discuss the *uncapacitated facility location problem.* We are given a set of clients $\mathcal{D}$, and a set of facilities $\mathcal{F}$. Every facility $i \in \mathcal{F}$ has a non-negative opening cost f_i associated with it. Every client $j \in \mathcal{D}$ has a non-negative demand d_j which need to be served by one or more open facilities. The cost of serving a unit demand of client j by facility i is denoted by c_{ij}. The aim is to serve all the demands at minimum cost. We discuss the metric version of the problem, in which the cost function c_{kl} is defined for $k, l \in \mathcal{F} \cup \mathcal{D}$, and it follows the triangular inequality.

Hochbaum [26] presented a greedy algorithm with $O(log\, n)$ approximation guarantee. Shmoys, Tardos, and Aardal [36] used the techniques of Lin and Vitter [30] to give the first constant-approximation algorithm. Guha and Khuller in [20] showed that no ρ approximation algorithm for metric UFL exists for any $\rho < 1.463$, unless $NP \subset DTIME(n^{O(log\, log\, n)})$. Byrka [10] gave a 1.5-approximation algorithm, which is the best approximation factor for the problem so far.

Throughout our discussion, we will use i to index the facilities in $\mathcal{F}$ and j to index the clients in $\mathcal{D}$. For the sake of ease of presentation, we will also assume unit demand at every client(i.e. $d_j = 1$). The following integer program captures this version of the facility location problem:

$$\begin{aligned} \text{minimize:} \quad & \sum_{i,j} c_{ij} \cdot x_{ij} + \sum_i f_i \cdot y_i \\ \text{subject to:} \quad & \forall j, \quad \sum_i x_{ij} \geq 1 && (2.1) \\ & \forall i, j, \quad x_{ij} \leq y_i && (2.2) \\ & \forall i, j, \quad x_{ij}, y_i \in \{0, 1\} && (2.3) \end{aligned}$$

A natural linear relaxation of the above integer program relaxes the integrality constraint to the following:

$$\forall i, j, \qquad x_{ij} \geq 0 \qquad \text{and} \qquad y_i \geq 0$$

Since there are no capacity constraints, we need not enforce that $y_i \leq 1$ in the constraint above. We will henceforth refer to this LP as $UFL - LP$.

2.1.1 LP Rounding

The rounding algorithm proceeds in two stages. The first stage is *filtering* and the second stage is *clustering and rounding*. A filtering step ensures that a client is served (fractionally) by only nearby facilities. The main obstacle to rounding the $UFL - LP$ solution are the clients which are served by many fractionally open facilities to a small extent. The clustering step clusters all the facilities close to a suitably chosen client, such that this cluster contains large facility weight (in this case, the clusters contain enough weight to pay for opening a facility integrally). The clusters are created, so that every client is *close* to some suitable set of clusters, and is able to send a significant part of their demand to these clusters.

Thus the algorithm results in a set of clusters, each of which can open a facility without paying any extra cost than what is paid for the cluster by the $UFL - LP$ solution, and all the clients are sending enough flow to some of these clusters that are *close* enough to the client. Once we have such a solution with integrally open facilities and fractional assignment costs, we are done, because assigning a client j to a facility $i_j = \underset{i:x_{ij}>0}{\operatorname{argmin}}\, c_{ij}$ yields an integral assignment with a connection cost of at most the fractional connection cost.

2.1.2 Rounding Algorithm

Filtering

The rounding algorithm starts by *filtering*, first introduced by Lin and Vitter in [30]. The filtering step modifies the solution so that a client is served only by *nearby* facilities, without increasing the cost of the solution by much. Let (x^*, y^*) be an optimal solution to the $UFL - LP$. The contribution of a client j towards the optimal LP solution is denoted by $\bar{C}_j$ and is defined by:

$$\bar{C}_j = \sum_i x^*_{ij} \cdot c_{ij} \tag{2.4}$$

Lemma (2.1.1) shows that a *good* fraction of a client j is served by nearby facilities. Mathematically, let $N_j(\alpha) = \{i : c_{ij} \leq \frac{1}{1-\alpha} \cdot \bar{C}_j\}$, for $\alpha \in (0, 1)$, then $\sum_{i \in N_j(\alpha)} x_{ij} \geq \alpha$, i.e. an α fraction of client j is served by facilities *close enough* to j.

Let $Cost(x, y)$ denote the value of the $UFL - LP$ objective for the solution (x, y). For a fixed value of α (to be decided later) and due to the Lemma (2.1.1), one can easily modify (x^*, y^*) to obtain another solution $(\bar{x}, \bar{y})$, where the following hold:

$$\begin{aligned}
&\forall i, \quad \bar{y}_i = \min(1, y^*_i/\alpha) \\
&\forall j, \quad \bar{x}_{ij} = 0 \quad \text{if } i \notin N_j(\alpha) \\
&\forall i, j, \quad \bar{x}_{ij} \leq \frac{1}{\alpha} \cdot x^*_{ij} \\
&\forall j, \quad \sum_i \bar{x}_{ij} = 1 \\
&Cost(\bar{x}, \bar{y}) \leq \frac{1}{\alpha} \cdot Cost(x^*, y^*)
\end{aligned}$$

and thus the new solution ensures that if $\bar{x}_{ij} > 0$ then i is *near* j, i.e. $i \in N_j(\alpha)$.

Clustering Facilities and Rounding

We now modify the solution $(\bar{x}, \bar{y})$ to obtain a more structured solution $(\hat{x}, \hat{y})$. If a client j is served fractionally by some integrally open facility i, then one can simply assign the client to this facility integrally. Due to the filtering step, we know that $c_{ij} \leq \frac{1}{1-\alpha} \cdot \bar{C}_j$, and hence the new connection cost of this client is bounded in terms of its connection cost in the optimal LP solution, which is what we desire. Hence we need to cluster facilities of only those clients that are not served by any integrally open facility.

Initialize $\hat{D} = \{j : \forall i,\ \bar{x}_{ij} > 0 \Rightarrow \bar{y}_i < 1\}$, i.e., set of all clients not served by any integrally open facility. Also let $\hat{F} = \{i : \bar{y}_i \neq 1\}$, i.e., set of facilities not yet opened integrally. Initialize all the $\hat{x}$-values to 0 and set $\hat{y}_i = 1$ if $\bar{y}_i = 1$, otherwise set $\hat{y}_i$ to 0.

Clustering Facilities: While $\hat{D} \neq \phi$, repeat the following steps :

1. Pick $j \in \hat{D}$ with minimum $\bar{C}_j$ value.
2. Let $S_j = \{i \in \hat{F} : \bar{x}_{ij} > 0\}$ and $D_j = \{j' \in \hat{D} : \exists i \in S_j \text{ s.t. } \bar{x}_{ij'} > 0\}$. By definition itself, $j \in D_j$. S_j is said to be the *cluster* and j is the corresponding *cluster center.*
3. Do: $\hat{D} \leftarrow \hat{D} \setminus D_j$ and $\hat{F} \leftarrow \hat{F} \setminus S_j$

Opening Facilities: For every cluster S_j, let o_j be the cheapest facility in S_j. Set $\hat{y}_{o_j} = 1$ and $\hat{y}_i = 0\ \forall i \in S_j \setminus \{o_j\}$, i.e. open the cheapest facility in the cluster.

Assigning Clients: Assign every client to the nearest open facility, i.e. for a client j', let $i_{j'} = \operatorname*{argmin}_{i:\hat{y}_i > 0} c_{ij'}$. Set $\hat{x}_{ij'} = 1$ for $i = i_{j'}$ and 0 otherwise.

2.1.3 Analysis

Let us first state and prove the lemma that enabled our filtering step to be feasible.

Lemma 2.1.1. *Let $\alpha \in (0, 1)$, and let $N_j = \{i : c_{ij} \leq \frac{1}{1-\alpha} \cdot \bar{C}_j\}$, then $\sum_{i \in N_j} y_i \geq \alpha$.*

Proof. Let $\sum_{i \notin N_j} x^*(i,j) > 1 - \alpha$. Then,

$$\bar{C}_j \;=\; \sum_i x^*(i,j) \cdot c_{ij} \;>\; \sum_{i \notin N_j} x^*(i,j) \cdot c_{ij} \;\geq\; \frac{1}{1-\alpha} \cdot \bar{C}_j \cdot \sum_{i \notin N_j} x^*(i,j) \;\geq\; \bar{C}_j$$

which is a contradiction. Thus, it follows that

$$\sum_{i \in N_j} y_i^* \;\geq\; \sum_{i \in N_j} x^*(i,j) \;\geq\; 1 - \sum_{i \notin N_j} x^*(i,j) \;\geq\; \alpha$$

where the first inequality follows form the LP constraint (2.2) and the second inequality follows from the constraint (2.1) of the $UFL - LP$. □

For $\alpha = \frac{1}{2}$, the above lemma asserts that in a ball of radius $2 \cdot \bar{C}_j$ centered at client j, the amount of facility weight opened by the solution is at least a $\frac{1}{2}$. The set N_j is the set of facilities *near* to j. Thus the filtering step as presented in the algorithm is feasible.

Facility Opening Cost: Let $j \in \hat{D}$ be a cluster center, and S_j be the corresponding cluster. At any stage of the algorithm, $\hat{D}$ is the set of those clients which are served entirely by fractionally open facilities. Thus, we have,

$$\sum_{i \in S_j} y_i \;\geq\; \sum_{i \in S_j} x_{ij} \;=\; 1$$

Note that:

$$\sum_{i \in S_j} f_i \cdot \bar{y}_i \;\geq\; f_{o_j} \cdot \sum_{i \in S_j} \bar{y}_i \;\geq\; f_{o_j} \;=\; \sum_{i \in S_j} f_i \cdot \hat{y}_i$$

Thus, in a particular cluster, cost of opening the cheapest facility is paid off completely by the facility opening cost of the cluster S_j by the solution $(\bar{x}, \bar{y})$. Since these clusters are mutually disjoint sets, we have:

$$\sum_i \hat{y}_i \cdot f_i \quad \leq \quad \sum_i \bar{y}_i \cdot f_i \tag{2.5}$$

Connection Cost: Let $\mathcal{J}$ be the set of all cluster centers. Due to the filtering step, all facilities in a cluster S_j are *close* to each other. We have:

$$\forall i, i' \in S_j, \qquad c_{ii'} \quad \leq \quad c_{ij} + c_{ji'} \quad \leq \quad \frac{2}{1-\alpha}\bar{C}_j. \tag{2.6}$$

The following Lemma shows that every client is close to some open facility, and thus the cost of connecting every client to its closest open facility is not much.

Lemma 2.1.2. *For every client $j \in \mathcal{D}$, there is an open facility i (i.e. $\hat{y}_i = 1$), with $c_{ij} \leq \frac{3}{1-\alpha}\bar{C}_j$.*

Proof. If $j \in \mathcal{J}$, then every facility in S_j is *close* to j. In particular, $o_j \in S_j$ is such that $\hat{y}_{o_j} = 1$ and $c_{o_j j} \leq \frac{1}{1-\alpha} \cdot \bar{C}_j$. If $j \notin \mathcal{J}$, then there is a cluster center j^*, such that $\bar{C}_{j^*} \leq \bar{C}_j$, and that the cluster S_{j^*} serves j (fractionally), i.e. $\exists i \in S_{j^*}$, such that, $x_{ij} > 0$. Because of filtering, we know that $c_{ij} \leq \frac{1}{1-\alpha} \cdot \bar{C}_j$ and from (2.6), we have $c_{io_{j^*}} \leq \frac{2}{1-\alpha}\bar{C}_{j^*} \leq \frac{2}{1-\alpha}\bar{C}_j$. Thus using triangular inequality, we have that $c_{jo_{j^*}} \leq \frac{3}{1-\alpha}\bar{C}_j$. $\square$

The following bound on the connection cost is thus immediate.

Corollary 2.1.3. *The connection cost, when every client is served by the nearest open facility, is bounded by $\frac{3}{1-\alpha}\sum_j \bar{C}_j$*

Total Cost: Thus, we have:

$$\begin{aligned} Cost(\hat{x}, \hat{y}) = \sum_i f_i \cdot \hat{y}_i + \sum_i \sum_j \hat{x}_{ij} \cdot c_{ij} &\leq \sum_i \bar{y}_i \cdot f_i + \frac{3}{1-\alpha} \cdot \sum_j \bar{C}_j \\ &\leq \frac{1}{\alpha} \cdot \sum_i y_i^* \cdot f_i + \frac{3}{1-\alpha} \cdot \sum_j \bar{C}_j \end{aligned} \tag{2.7}$$

Taking $\alpha = \frac{1}{4}$, we have,

$$Cost(\hat{x}, \hat{y}) \leq 4 \cdot \left(\sum_i y_i^* \cdot f_i + \sum_j \bar{C}_j \right) = 4 \cdot Cost(x^*, y^*) \tag{2.8}$$

Thus yielding a 4-approximation algorithm. Shmoys, Tardos and Aardal [36] use a randomized filtering step, choosing α randomly from some interval, which improves this approximation ratio to 3.16.

2.2 Capacitated FLP

In this section, we discuss the *capacitated facility location problem.* In this problem, we are given a set of facilities, $\mathcal{F}$ and a set of clients $\mathcal{D}$. Each client $j \in \mathcal{D}$ has a demand d_j that needs to be served by one or more open facilities. Opening a facility $i \in \mathcal{F}$ incurs a cost f_i. It costs c_{ij} to serve 1 unit of demand of a client j by facility i. Furthermore, no facility may service more than u units of demand. The aim is to service all clients at the lowest cost possible. We are going to assume that for any $k, l \in \mathcal{F} \cup \mathcal{D}$, c_{kl} is defined and obeys the triangular inequality [14].

Capacitated facility location is studied in two varieties - with *hard capacities* and with *soft capacities.* The facilities are said to have hard capacities if only one facility is allowed to be open at a facility location. In the soft capacities version, however, a facility i with opening cost f_i and capacity u_i pays $\lceil \frac{C}{u_i} \rceil \cdot f_i$ for serving C clients, or in other words, $\lceil \frac{C}{u_i} \rceil$ facilities are opened at the location i . Further effort goes in designing algorithms that ensure that the number of such facilities opened at a location is bounded.

Approximation algorithms for capacitated facility location problem with hard capacities are all based on local search. Korupolu, Plaxton and Rajaraman gave the first constant approximation algorithm for the case of *uniform capacities* [34] (i.e. every facility has the same capacity). Their analysis was subsequently improved by Chudak and Williamson [14] which was further improved to yield a 3-approximation algorithm by Garg et al [1]. For the facility location problem with non-uniform hard capacities, Pal, Tardos and Wexler [33] gave the first constant factor approximation algorithm. The current best known approximation factor for this version is 5.83 by Zhang, Chen and Ye [39].

Jain and Vazirani [27], gave a 4-approximation algorithm for the facility location problem with non-uniform soft capacities. The current best known algorithm for this variation is a 2-approximation algorithm by Mahdian, Ye and Zhang [31]. Recently, there has been some work on the *lower bounded facility location problem*, where every open facility has to serve at least a certain number B of clients. The first constant approximation algorithm for this version was given by Svitkina [37], which was improved by Ahmadian and Swamy [3] to yield a 82.6-approximation algorithm.

We will use i to index the facilities in $\mathcal{F}$, and j to index the clients in $\mathcal{D}$. The following integer program captures the capacitated facility location problem, with uniform capacities:

$$\begin{aligned}
\text{minimize:} \quad & \sum_{i,j} c_{ij} \cdot x_{ij} \cdot d_j + \sum_i f_i \cdot y_i \\
\text{subject to:} \quad & \forall j, \quad \sum_i x_{ij} \geq 1 && (2.9) \\
& \forall i,j, \quad x_{ij} \leq y_i && (2.10) \\
& \forall i, \quad \sum_j x_{ij} \cdot d_j \leq u \cdot y_i && (2.11) \\
& \forall i,j, \quad x_{ij}, y_i \in \{0,1\} && (2.12)
\end{aligned}$$

The natural linear relaxation of the above program relaxes the integrality constraints to the following:

$$\forall i,j, \qquad x_{ij} \geq 0 \qquad \text{and} \qquad 0 \leq y_i \leq 1$$

Henceforth, we refer to the resulting LP as $CFL - LP$. The capacitated version of facility location problem comes in two versions: the unsplittable version, where the demand located at a client must be served by a single facility, and the splittable version, where the demand present at a client is allowed to be split among open facilities. In the splittable version, if all capacities are integral, then standard network flow theory implies that it is wlog to assume that the demand is split integrally among the open facilities. With unit demands (i.e. $d_j = 1 \; \forall j$), the two versions are same. For simplicity of presentation, we will consider this special case, where each client has a unit demand. However, the algorithm can be easily adapted to work for the more general case of splittable non-unit demands. Thus the objective function becomes:

$$\text{minimize:} \quad \sum_{i,j} c_{ij} \cdot x_{ij} + \sum_i f_i \cdot y_i$$

Unfortunately, this LP-relaxation has an unbounded integrality gap, as demonstrated by the following example presented in [36]: Consider an instance with $u + 1$ locations with unit demand and at distance 0 from each other, with $f_1 = 0$ and $f_2 = f_3 = \ldots = f_{u+1} = 1$. Any integral solution has to open at least one facility of unit cost, because there are $u + 1$ demands, thus incurring a cost of at least 1. The optimum fractional solution has a cost of $\frac{1}{u+1}$: Set $y_1 = 1$, $y_2 = \frac{1}{u+1}$ and $y_3 = \ldots = y_{u+1} = 0$, $x_{1j} = \frac{u}{u+1}$ and $x_{2j} = \frac{1}{u+1}$, $\forall j \in \{1, 2, \ldots, u+1\}$. However, if one allows multiple facilities to be opened at a location, the above example is no longer a problem, which is equivalent to saying that the capacities are allowed to be violated by a bounded factor.

2.2.1 LP Rounding

As in the uncapacitated case, the rounding algorithm proceeds in two stages. The first stage is *filtering* and the second stage is *clustering and rounding.* The filtering step is exactly the same as in the uncapacitated case, and it ensures that the facilities serving a client are *close* to the client. Like in the uncapacitated case, the problem in rounding the solution to $CFL - LP$ are those clients which are served to a large extent by facilities opened fractionally by the fractional solution. But since the facilities have capacity limits, a more sophisticated clustering algorithm is required here. This clustering step clusters the facilities together into disjoint clusters, where each cluster has enough facility weight (at least half) so that facilities can be opened in these clusters without paying a much greater cost than what the cluster pays towards the facility opening cost in the fractional solution. Since a facility can serve only a limited amount of demand, we need to open multiple facilities in a given cluster, instead of just one. Also, we can no longer ensure that every client is served fully by the facilities opened, but we can ensure that every client is served to a large extent (at least half) by these clusters. This is reasonable enough, because one can then suitably scale the solution to satisfy the clients fully, by violating the capacity of the facilities by some bounded amount.

2.2.2 Rounding Algorithm

Filtering

As in the uncapacitated case, we can modify (x^*, y^*) to obtain a more structured solution $(\bar{x}, \bar{y})$, where the following holds:

$$\begin{aligned}
&\forall i, \quad \bar{y}_i = y_i^*/\alpha \\
&\forall j, \quad \bar{x}_{ij} = 0 \quad \text{if } i \notin N_j(\alpha) \\
&\forall i, j, \quad \bar{x}_{ij} \leq \frac{1}{\alpha} \cdot x_{ij}^* \\
&\forall j, \quad \sum_i \bar{x}_{ij} = 1
\end{aligned}$$

where $N_j(\alpha) = \{i : c_{ij} \leq \frac{1}{1-\alpha} \cdot \bar{C}_j\}$, for $\alpha \in (0, 1)$. Thus, the new solution ensures that if $\bar{x}_{ij} > 0$ then i is *near* j, i.e. $i \in N_j(\alpha)$. Also, $Cost(\bar{x}, \bar{y}) \leq \frac{1}{\alpha} \cdot Cost(x^*, y^*)$. Note that $\bar{y}_i$ may be greater than 1, which is required, because otherwise the capacity constraint may be violated. As discussed earlier, since the algorithm is allowed to open multiple facilities at a location, $\bar{y}_i > 1$ does not pose as a problem, but allows us do precisely that.

Clustering Facilities and Rounding

As discussed earlier, one cannot ensure here that a client is served fully by the clusters. What one can ensure, however, is that the clients are served to a large extent by the clusters. Thus, we keep a client in the set $\hat{D}$ till it is not served to an extent of half by the clusters. $\mathcal{J}$ is the set of all cluster centers. The modified clustering algorithm is as follows:

Clustering Facilities: Initialize $\hat{D} = \mathcal{D}$, and $\mathcal{J} = \phi$. While $\hat{D} \neq \phi$, repeat the following steps :

1. Pick $j \in \hat{D}$ with minimum $\bar{C}_j$ value. Set $\mathcal{J} \leftarrow \mathcal{J} \cup \{j\}$.
2. Let $S_j = \{i \in \hat{F} : \bar{x}_{ij} > 0\}$. S_j is said to be a *cluster* and j is the corresponding *cluster center*. Let D_j be the collection of all those clients in $\hat{D}$ which are served to an extent of at least half by the clusters, i.e. $D_j = \{j' \in \hat{D} : \sum_{j'' \in \mathcal{J}} \sum_{i \in S_{j''}} \bar{x}_{ij'} \geq \frac{1}{2}\}$. By definition itself, $j \in D_j$.
3. Do: $\hat{D} \leftarrow \hat{D} \setminus D_j$ and $\hat{F} \leftarrow \hat{F} \setminus S_j$

Opening Facilities: For every cluster S_j, let O_j be the cheapest $\left\lceil \sum_{i \in S_j} y_i \right\rceil$ facilities in S_j. Set $\hat{y}_i = 1\ \forall i \in O_j$ and $\forall j \in \mathcal{J}$. Set $\hat{y}_i = 0\ \forall i \in \mathcal{F} \setminus \bigcup_{j \in \mathcal{J}} O_j$.

Assigning Clients: Solve a transportation problem, where every client j' sends $\sum_{i \in S_j} \bar{x}_{ij'}$ units of demand to the open facilities O_j of only those clusters S_j which have $\bar{C}_j \leq \bar{C}_{j'}$. An open facility i accepts at most u units of demand. The solution of this transportation problem defines the $\hat{x}$ values. Lemma 2.2.1 shows that such a transportation problem is feasible.

2.2.3 Analysis

We first show that the transportation problem set up in the last step of the algorithm is a feasible one.

Lemma 2.2.1. *Let $\mathcal{T}$ be the following transportation problem: the clients $j' \in \mathcal{D}$ act as suppliers, the set of facilities $\bigcup_j O_j$ opened by the algorithm act as consumers. A supplier $j' \in \mathcal{D}$ has a supply of $\sum_{i \in S_j} x_{ij'}$ for the clusters S_j which have $\bar{C}_j \leq \bar{C}_{j'}$. Each consumer $i \in O_j$ has a capacity u. $\mathcal{T}$ is a feasible transportation problem, i.e. the total supply for a cluster is at most the total capacity of the cluster.*

Proof. For a cluster S_j with the cluster center j (i.e. $j \in \mathcal{J}$), we have:

$$\begin{aligned} \text{total capacity} &= u \cdot |O_j| \;=\; u \cdot \left\lceil \sum_{i \in S_j} y_i \right\rceil \\ \text{total supply} &= \sum_{j' \in \mathcal{D}} \sum_{i \in S_j} \bar{x}_{ij'} \;\leq\; \sum_{i \in S_j} u \cdot \bar{y}_i \;\leq\; u \cdot \left\lceil \sum_{i \in S_j} \bar{y}_i \right\rceil \end{aligned}$$

where the first inequality follows from (2.11). Thus, the total supply for a cluster is at most the total capacity of the cluster and hence the transportation problem is well defined and can be solved in polynomial time. The solution to this transportation problem defines the $\hat{x}_{ij}$ values. □

Lemma 2.2.2. *A client $j' \in \mathcal{D}$ is assigned to an extent of at least half among the clusters, i.e., $\sum_{j \in \mathcal{J}} \sum_{i \in S_j} \hat{x}_{ij} \geq \frac{1}{2}$*

Proof. If $j' \in \mathcal{J}$, then by definition of $\hat{D}$, we have that $\sum_{i \in S_{j'}} \bar{x}_{ij'} \geq \frac{1}{2}$. Thus,

$$\sum_{j \in \mathcal{J}} \sum_{i \in S_j} \hat{x}_{ij'} = \sum_{j \in \mathcal{J} : \bar{C}_j \leq \bar{C}_{j'}} \sum_{i \in S_j} \hat{x}_{ij'} = \sum_{j \in \mathcal{J} : \bar{C}_j \leq \bar{C}_{j'}} \sum_{i \in S_j} \bar{x}_{ij'} \geq \sum_{i \in S_{j'}} \bar{x}_{ij'} \geq \frac{1}{2}$$

Otherwise, $j' \notin \mathcal{J}$. Since j is not added as a cluster center, thus it is being served to an extent of at least half by the clusters created by the time j' was eligible to be considered as a cluster center. All the clusters created so far had cluster centers j such that $\bar{C}_j \leq \bar{C}_{j'}$.

Thus, we have,

$$\sum_{j\in\mathcal{J}}\sum_{i\in S_j}\hat{x}_{ij'} = \sum_{j\in\mathcal{J}:\bar{C}_j\le\bar{C}_{j'}}\sum_{i\in S_j}\hat{x}_{ij'} = \sum_{j\in\mathcal{J}:\bar{C}_j\le\bar{C}_{j'}}\sum_{i\in S_j}\bar{x}_{ij'} \ge \frac{1}{2}$$

This completes the proof. □

An important consequence of the way the $\hat{x}$-values are defined is that a client j' is served by a cluster in the solution $(\hat{x},\hat{y})$ only if it was also served by this cluster in the solution $(\bar{x},\bar{y})$. This observation shall be helpful for us in bounding the connection cost of the solution $(\hat{x},\hat{y})$. Also note that if $\hat{x}_{ij'} > 0$ for some $i \in S_j$, $j \in \mathcal{J}$, then $\bar{C}_j \le \bar{C}_{j'}$.

Facility Opening Cost: We know that

$$\sum_{i\in S_j}\bar{y}_i \ge \sum_{i\in S_j}\bar{x}_{ij} \ge \frac{1}{2}$$

The first inequality follows from the constraint (2.10), and the second inequality follows from the definition of $\hat{D}$. Also,

$$\sum_{i\in S_j}\bar{y}_i \;\ge\; \frac{1}{2} \;\Rightarrow\; \lceil\sum_{i\in S_j}\bar{y}_i\rceil \;\le\; 2\cdot\sum_{i\in S_j}\bar{y}_i$$

Thus, one can easily verify (it follows from using Lemma (3.4.2)) that the cost of opening cheapest $\lceil\sum_{i\in S_j}\bar{y}_i\rceil$ facilities in S_j is upper bounded by $2\cdot\sum_{i\in S_j} f_i\cdot\bar{y}_i$. Thus, we have,

$$\sum_{i\in S_j}\hat{y}_i\cdot f_i \;\le\; 2\cdot\sum_{i\in S_j}\bar{y}_i\cdot f_i.$$

Since the clusters are mutually disjoint and their union may not cover the entire set of facilities $\mathcal{F}$, we have:

$$\sum_{i}\hat{y}_i\cdot f_i \;=\; \sum_{j\in\mathcal{J}}\sum_{i\in S_j}\hat{y}_i\cdot f_i \;\le\; 2\cdot\sum_{j\in\mathcal{J}}\sum_{i}\bar{y}_i\cdot f_i \;\le\; 2\cdot\sum_{i}\bar{y}_i\cdot f_i \qquad (2.13)$$

Connection Cost: Due to the filtering step, all facilities in S_j are *close* to each other.

We have:

$$\forall i, i' \in S_j, \qquad c_{ii'} \quad \leq \quad c_{ij} + c_{ji'} \quad \leq \quad \frac{2}{1-\alpha}\bar{C}_j. \tag{2.14}$$

The maximum distance between any client $j' \in \mathcal{D}$ and a facility $i \in O_j$ (where j is a cluster center) that serves it fractionally (i.e. $\hat{x}_{ij'} > 0$), can be bound in terms of $\bar{C}_{j'}$. This is possible because, $\exists i'' \in S_j$, such that, $\bar{x}_{i''j'} > 0$ and thus due to the filtering step, $c_{i''j'} \leq \frac{1}{1-\alpha}\bar{C}_{j'}$, and thus, we have:

$$\forall i \in S_j \qquad c_{ij'} \quad \leq \quad c_{ii''} + c_{i''j'} \quad \leq \quad \frac{2}{1-\alpha}\bar{C}_j + \frac{1}{1-\alpha}\bar{C}_{j'} \quad \leq \quad \frac{3}{1-\alpha}\bar{C}_{j'} \tag{2.15}$$

where the first inequality follows from the metric property, the second inequality follows from (2.14), and the third property follows because of the way the transportation problem was set up.

Therefore, at the end of the rounding algorithm, every client is satisfied to an extent of at least a $\frac{1}{2}$ by integrally open facilities, and the following property holds:

$$\forall i, j, \qquad \hat{x}_{ij} > 0 \quad \Rightarrow \quad c_{ij} \ \leq \ \frac{3}{1-\alpha} \cdot \bar{C}_j \tag{2.16}$$

Thus, as discussed earlier, we are now done, because we can now open twice as many facilities at every location, (set $\hat{y}_i \leftarrow 2 \cdot \hat{y}_i, \ \forall i$) and suitably modifying the $\hat{x}$-values, increasing them to at most twice their original value, and satisfying $\sum_i \hat{x}_{ij} = 1 \ \forall j$. Since we do not introduce any new non-zero x-value in this process, the property (2.16) still holds, and the equation (3.27) changes to:

$$\sum_i \hat{y}_i \cdot f_i \quad \leq \quad 4 \cdot \sum_i \bar{y}_i \cdot f_i \tag{2.17}$$

The property (2.16) ensures that the connection cost of the solution $(\hat{x}, \hat{y})$ is not very high. The connection cost is:

$$\sum_i \sum_j \hat{x}_{ij} \cdot c_{ij} \leq \frac{3}{1-\alpha} \cdot \sum_j \bar{C}_j \cdot \sum_i \hat{x}_{ij} = \frac{3}{1-\alpha} \cdot \sum_j \bar{C}_j \tag{2.18}$$

The equality holds because $\sum_i \hat{x}_{ij} = 1, \ \forall j$.

Total Cost: The solution $(\hat{x}, \hat{y})$ has integral $\hat{y}$-values and fractional $\hat{x}$-values. The $\hat{x}$-

values can be thought of as flow from demands to open facilities, and since the demand at each location is integral, an integral flow exists between the demands and the open facilities of the same cost as the fractional flow, and this integral flow can be found in polynomial time. Thus the cost of the integral solution is :

$$\begin{aligned} Cost(\hat{x}, \hat{y}) = \sum_i f_i \cdot \hat{y}_i + \sum_i \sum_j \hat{x}_{ij} \cdot c_{ij} &\leq 4 \cdot \sum_i \bar{y}_i \cdot f_i + \frac{3}{1-\alpha} \cdot \sum_j \bar{C}_j \\ &\leq \frac{4}{\alpha} \cdot \sum_i y_i^* \cdot f_i + \frac{3}{1-\alpha} \cdot \sum_j \bar{C}_j \end{aligned} \tag{2.19}$$

Taking $\alpha = \frac{4}{7}$, we have,

$$Cost(\hat{x}, \hat{y}) \leq 7 \cdot \left(\sum_i y_i^* \cdot f_i + \sum_j \bar{C}_j \right) = 7 \cdot Cost(x^*, y^*) \tag{2.20}$$

And thus the rounding algorithm is a 7-approximation algorithm.

Chapter 3

Robust Facility Location Problem

The classical facility location problems like uncapacitated facility location(UFL), capacitated facility location(CFL), as well as many other standard facility location models assume that facilities are completely reliable and not prone to failures. However, there are various settings where this is not true, and in order to deliver effective solutions, we need to design solutions that are resilient to failures. Various ways of incorporationg such resilience have been considered in the literature.

Jain and Vazirani introduced the fault tolerant version of facility location problems in [28]. In this version, a client j is to be served by r_j open facilities, its shipping cost being some weighted linear combination of the costs of shipping demand from all the facilities to which it is assigned. This introduces a kind of robustness in the system, so that each client is not connected to just one facility, but several facilities, and thus if some facilities go down, the client is possibly still served.

Chechik and Peleg introduced a new kind of robustness in the facility location problem framework in their recent work [13]. In this variant, once we decide which facilities to open, an adversary closes at most β of them. The clients are then connected to the closest facility from the remaining open facilities. The client pays only for the cost of shipping its demand from the surviving facility that actually supplied its demand.

More formally, the *β-robust uncapacitated facility location problem* ($\beta RUFLP$) is defined as follows: We are given a set of clients $\mathcal{D}$, and a set of facilities $\mathcal{F}$. Every facility $i \in \mathcal{F}$ has a non-negative opening cost f_i associated with it. Every client $j \in \mathcal{D}$ has a non-negative demand d_j which need to be served by one or more open facilities. The cost of serving a unit demand of client j by facility i is denoted by c_{ij}. However, when the set of facilities F to be opened is chosen, an adversary can close up to β of the open facilities, and the clients are assigned to the remaining open facilities. The aim is to search for a set of facilities F, that minimizes the sum of costs of opening the facilities in F and the costs of assigning the demand of each node to the open facilities that did not fail, for any failure of up to β facilities. We discuss the metric version of the problem, in which the cost function c_{kl} is defined for $k, l \in \mathcal{F} \cup \mathcal{D}$, and it follows the triangle inequality.

In Chapter 4, we study the extension of this definition of robustness to the k-median facility location problem, yielding the *β-robust k-median facility location problem*. In this version, we are again looking for a set of facilities F that minimizes the sum of the cost of the opening facilities and the cost of serving clients by the open facilities not closed by the adversary, subject to the additional constraint that the set F cannot have more than k facilities.

Chechik and Peleg give a 6.5 approximation for $\beta RUFLP$ problem with at most 1 failure, and a $1.5 + 7.5\beta$-approximation with at most $\beta > 1$ failures. Byrka, Ghodsi and Srinivasan [11] gave an LP-rounding based $(\beta + 5 + 4/\beta)$-approximation algorithm for this problem. We study this version in this chapter, and present a different rounding algorithm than what is presented in [11]. Our aim in studying $\beta RUFLP$ with this rounding algorithm was to get some insights into the structure of the problem and apply these ideas to the β-robust k-median facility location problem, and thus we did not attempt to optimize the approximation factors. We assume that all the facility opening costs are same (i.e. $\forall i, f_i = f$) and obtain a 13.93-approximation algorithm for the special case of $\beta = 1$.

As Chelek and Peleg note in [13], their version is closely related to the 2-stage stochastic model. The stochastic version proceeds in two decision stages. There is a probability distribution over possible scenarios (a scenario, in facility location problems, specifies the clients and their demands). In the first stage, some facilities may be purchased. This is followed by some scenario, and a second decision stage is entered. One is now allowed to open more facilities in this stage, but the facility costs may now be much higher than what they were in the first stage. In contrast to this, in the robust version, the facilities must be opened and paid for in advance, and these advance decisions must be adequate under al possible future scenarios.

3.1 Linear Programming Formulation

Byrka, Ghodsi and Srinivasan consider the following Integer Program for this problem in [11]:

$$
\begin{aligned}
\text{minimize:} \quad & \sum_{i\in\mathcal{F}} f_i \cdot y_i + \max_A \sum_i \sum_j c_{ij} \cdot x(A,i,j) & \\
\text{subject to:} \quad & \forall A, j, \quad \sum_i x(A,i,j) \geq 1, & (3.1)\\
& \forall i, A, j, \quad x(A,i,j) \leq y_i, & (3.2)\\
& \forall A, i \in A, j, \quad x(A,i,j) = 0 & (3.3)\\
& \forall i, A, j, \quad x(A,i,j),\ y_i \in \{0,1\}. & (3.4)
\end{aligned}
$$

Here, A denotes a set of facilities of cardinality β. $x(A,i,j) = 1$ if the client j is served by facility $i \notin A$ when the set of facilities closed is A, and is 0 otherwise. The constraints ensure that a facility serves a client only if it is open. Also, if a facility $i \in A$, then it cannot serve any clients. The integrality constraints in the above program can be relaxed to the following:

$$\forall i, A, j, \qquad x(A,i,j) \geq 0 \qquad \text{and} \qquad y_i \geq 0$$

Note that the above formulation is not exactly a linear program, because the presence of the max term in the objective makes it a non-linear function. But this can be easily taken care of by introducing a new variable B, and modifying the objective function to following:

$$\text{minimize:} \quad \sum_{i\in\mathcal{F}} f_i \cdot y_i \ + \ B$$

and introducing the following additional constraints:

$$\forall A, \quad \sum_i \sum_j c_{ij} \cdot x(A,i,j) \ \leq \ B$$

Note that here can *guess* the value of B (see Section 3.5.1). As B is no longer a variable but a known constant, the B term is no longer required in the objective function. Also, we can strengthen this LP, by adding the following additional constraints when the value of B is known:

$$\forall A, i, \quad \sum_j c_{ij} \cdot x(A,i,j) \ \leq \ B \cdot y_i$$

This trick of making the objective function linear and guessing the max term can be applied to every such LP formulation presented in this thesis.

This *LP* is henceforth referred to as $RUFLP - LP1$. A natural extension of this LP to the *β-robust k-median facility location problem* would be to introduce the following constraint in $RUFLP - LP1$:

$$\sum_i y_i \quad \leq \quad k$$

However, Lemma 4.1.1, we show that the *LP* so obtained has an unbounded integrality gap, and thus cannot be rounded. This motivated us to strengthen the above linear program. We will introduce the strengthened form of this linear program for the $\beta RUFLP$ (referred to as $RUFLP - LP2$) here, and work with it for the rest of the chapter. Lemma 3.2.1 shows that $RUFLP - LP2$ is stronger than $RUFLP - LP1$, in that every solution to $RUFLP - LP2$ can be converted to a solution to $RUFLP - LP1$ of the same cost, and hence the optimal value of $RUFLP - LP1$ is at most the optimal value of $RUFLP - LP2$, and thus $RUFLP - LP2$ provides a better lower bound on the optimum. Byrka, Ghodsi and Srinivasan, in [11] also give an example that shows that the integrality gap of $RUFLP - LP1$ is at least $\beta + 1 + \varepsilon$. Later in the chapter, we shall also show that this example breaks down for $RUFLP - LP2$.

3.2 A Different Linear Program

Let us consider the case when $\beta = 1$, i.e. when the adversary closes at most 1 facility. The objective function is:

$$\min_{F \subseteq \mathcal{F}} \left(c(F) + \max_{i \in \mathcal{F}} \sum_j c\left(j, F \setminus \{i\}\right) \right) \tag{3.5}$$

Here $c(F)$ is the cost of opening the facilities in F, and $c(j, S)$ denotes the cost of connecting client j to the closest facility in set S. The integer program we consider has $x(i, i', j)$ and y_i as variables where $i' \neq i$; $i, i' \in \mathcal{F}$ and $j \in \mathcal{D}$. $x(i, i', j) = 1$ iff client j is served by facility i and is served by facility i' in the event that the adversary closes facility i. We can think of this as assigning a client j to a pair ii' of facilities, where we call i as the **primary** facility of j and i' as the **secondary** facility of j.

Note that these variables are very different from the variables $x(A, i, j)$ of the earlier IP. The constraints of the earlier IP are such that the facilities in A need not be open for the variable $x(A, i, j) > 0$. The constraint (3.1), in fact, ensures that whatever the set A be, immaterial of whether facilities of A are opened (even fractionally) by the solution or not, there is some i, such that $x(A, i, j) > 0$. In the variables we use here, however, the variable $x(i, i', j)$ can be non-zero only if both the facilities i and i' are open facilities.

The objective function (3.5) can be re-written as follows:

$$\min_{F \subseteq \mathcal{F}} \left(c(F) + \sum_j c(j, F) + \max_{i \in \mathcal{F}} \sum_j \left(c\left(j, F \setminus \{i\}\right) - c\left(j, F\right) \right) \right) \tag{3.6}$$

In terms of our variables, we have:

$$c\left(j, F \setminus \{i\}\right) = \sum_{i' \neq i} x(i, i', j) \cdot c_{i'j} + \sum_{i_1 \neq i} \sum_{i_2 \neq i_1} x(i_1, i_2, j) \cdot c_{i_1 j} \tag{3.7}$$

$$c\left(j, F\right) = \sum_{i_1} \sum_{i_2 \neq i_1} x(i_1, i_2, j) \cdot c_{i_1 j} \tag{3.8}$$

And thus:

$$c\left(j, F \setminus \{i\}\right) - c\left(j, F\right) = \sum_{i' \neq i} x(i, i', j) \cdot (c_{i'j} - c_{ij}) \tag{3.9}$$

Thus the objective function is:

$$\min \left(\sum_i f_i \cdot y_i + \sum_j \sum_i \sum_{i' \neq i} x(i, i', j) \cdot c_{ij} + \max_i \sum_j \sum_{i' \neq i} x(i, i', j) \cdot (c_{i'j} - c_{ij}) \right) \tag{3.10}$$

The following integer program captures the problem:

$$\text{minimize:} \quad \left(\sum_i f_i \cdot y_i + \sum_j \sum_i \sum_{i' \neq i} x(i, i', j) \cdot c_{ij} + \max_i \sum_j \sum_{i' \neq i} x(i, i', j) \cdot (c_{i'j} - c_{ij}) \right)$$

$$\text{subject to:} \quad \forall j, \quad \sum_i \sum_{i' \neq i} x(i, i', j) \geq 1, \tag{3.11}$$

$$\forall i, j, \quad \sum_{i' \neq i} x(i, i', j) + x(i', i, j) \leq y_i, \tag{3.12}$$

$$\forall i, i', j, \quad x(i, i', j),\ y_i \in \{0, 1\}. \tag{3.13}$$

A linear relaxation of this integer program modifies the last constraint to the following :

$$\forall i, i', j, \qquad x(i, i', j) \geq 0 \qquad \text{and} \qquad 0 \leq y_i \leq 1$$

We call the resulting linear program as $RUFLP - LP2$. Constraint (3.11) ensures that every client is assigned to some facility pair. Constraint (3.12) asserts that a facility can not be used as a primary or secondary facility for a particular client j, unless it is open. Moreover, this constraint also says that a facility can only act as either a primary or a secondary facility for a particular client, but not both, in an integral solution. It is essentially this constraint that makes the $RUFLP - LP2$ different from $RUFLP - LP1$, and it is this constraint that captures the robustness of the problem. An integral solution to an instance of $\beta RUFLP$ opens at least 2 facilities. The constraint (3.12) ensures that this is true even for the fractional solution, which is not ensured in $RUFLP - LP1$:

$$\sum_i y_i \geq \sum_i \sum_{i' \neq i} (x(i, i', j) + x(i', i, j)) = 2 \cdot \sum_i \sum_{i' \neq i} x(i, i', j) \geq 2$$

The following lemma shows that this linear program is indeed stronger than the original one.

Lemma 3.2.1. *Any feasible solution (x, y) to $RUFLP - LP2$ of objective function value $Cost(x, y)$ can be converted into a feasible solution $(\bar{x}, \bar{y})$ to $RUFLP - LP1$ having an objective function value $Cost(\bar{x}, \bar{y})$, such that $Cost(x, y) = Cost(\bar{x}, \bar{y})$.*

Proof. We obtain the solution $(\bar{x}, \bar{y})$ from (x, y) as follows:

$$\begin{aligned} \forall i, \quad \bar{y}_i &= y_i \\ \forall i_1, i_2 \neq i_1, j, \quad \bar{x}(\{i_1\}, i_2, j) &= x(i_1, i_2, j) + \sum_{i_3 \neq i_2} x(i_2, i_3, j) \\ \forall i_1, j, \quad \bar{x}(\{i_1\}, i_1, j) &= 0 \end{aligned}$$

We need to show that this solution $(\bar{x}, \bar{y})$ is a feasible solution to *RUFLP* $-$ *LP1*. The constraint (3.1) is:

$$\forall\, i_1, j, \quad \sum_{i_2} x(\{i_1\}, i_2, j) = \sum_{i_2 \neq i_1} \left(x(i_1, i_2, j) + \sum_{i_3 \neq i_2} x(i_2, i_3, j) \right) = \sum_{i_2} \sum_{i_3 \neq i_2} x(i_2, i_3, j) \geq 1$$

where the last inequality follows from the constraint (3.11). The constraint (3.2) is:

$$\begin{aligned} \forall\, i_1, i_2 \neq i_1, j, \quad x(\{i_1\}, i_2, j) &= x(i_1, i_2, j) + \sum_{i_3 \neq i_2} x(i_2, i_3, j) \\ &\leq \sum_{i_3 \neq i_2} \left(x(i_3, i_2, j) + x(i_2, i_3, j) \right) \leq y_i \end{aligned}$$

By construction, constraint (3.3) is satisfied too. The integrality constraints and their linear relaxations also hold. Thus the solution $(\bar{x}, \bar{y})$ is feasible for *RUFLP* $-$ *LP1*. The cost of this solution is:

$$\begin{aligned} Cost(\bar{x}, \bar{y}) &= \sum_{i \in \mathcal{F}} f_i \cdot \bar{y}_i + \max_{i_1} \sum_{i_2} \sum_{j} \bar{x}(\{i_1\}, i_2, j) \cdot c_{i_2 j} \\ &= \sum_{i \in \mathcal{F}} f_i \cdot y_i + \max_{i_1} \sum_{i_2 \neq i_1} \sum_{j} \left(x(i_1, i_2, j) + \sum_{i_3 \neq i_2} x(i_2, i_3, j) \right) \cdot c_{i_2 j} \\ &= \sum_{i \in \mathcal{F}} f_i \cdot y_i + \max_{i_1} \sum_{j} \left(\sum_{i_2 \neq i_1} x(i_1, i_2, j) \cdot c_{i_2 j} + \sum_{i_2 \neq i_1} \sum_{i_3 \neq i_2} x(i_2, i_3, j) \cdot c_{i_2 j} \right) \\ &= Cost(x, y) \end{aligned}$$

where the last equality follows from (3.5) and (3.7). □

We shall later show that a similar result holds for the case $\beta > 1$ also, i.e. the *RUFLP* $-$ *LP2* extended for the general case of $\beta \geq 1$ is stronger than *RUFLP* $-$ *LP1*.

This immediately implies that any approximation guarantees obtained by using the former linear program, also apply for this linear program. In particular, we immediately have a rounding algorithm for $RUFLP - LP2$ having an approximation ratio of $(\beta + 5 + 4/\beta)$, using the algorithm of Byrka et al in [11]:

Corollary 3.2.2. *There is an LP-rounding based $(\beta+5+4/\beta)$-approximation algorithm for the $\beta RUFLP$ problem which rounds a fractional solution of $RUFLP - LP2$ to an integral solution.*

In an attempt to gain some insights into rounding our linear program $RUFLP - LP2$, we design a different rounding algorithm for this LP, albeit one that works only for uniform facility costs. In the rest of this chapter, we present this rounding algorithm. For the purposes of this new rounding algorithm, it will be more convenient to consider the following slightly modified objective function, which is an upper bound on the objective function (3.10):

$$\min \left(\sum_i f_i \cdot y_i \; + \sum_j \sum_i \sum_{i' \neq i} x(i,i',j) \cdot c_{ij} + \; \max_i \sum_j \sum_{i' \neq i} x(i,i',j) \cdot c_{i'i} \right) \tag{3.14}$$

We also need to show that the objective function in (3.14) is not very much larger than the objective function (3.10). We show this later in Section 3.5.2. We call the components of the objective function (3.14) as *Opening* cost, *Connection* cost and *Relocation* cost, respectively. We can *guess* the maximum relocation cost B, and thus we can remove the max term from the objective function and incorporate this information as a constraint, which reduces the objective function to:

$$\min \left(\sum_i f_i \cdot y_i \; + \sum_j \sum_i \sum_{i' \neq i} x(i,i',j) \cdot c_{ij} \right) \tag{3.15}$$

Since the relocation cost B is now a parameter to the LP, we will refer to the above LP as $RFLP(B)$ (Robust Facility Location Problem with relocation cost as B). Thus the cost of a solution to the problem is $B +$ the value of the objective function. Also, our algorithm works for the case where all facility opening costs are same (i.e. $f_i = f \; \forall \; i$). Thus, the following integer program captures the problem for which we devise a rounding algorithm

$$\begin{aligned}
\text{minimize:}\quad & \sum_i f \cdot y_i + \sum_i \left(\sum_{i' \neq i} \sum_j x(i,i',j) \cdot c_{ij} \right) \\
\text{subject to:}\quad & \forall j, \quad \sum_i \sum_{i' \neq i} x(i,i',j) \geq 1, && (3.16) \\
& \forall i,j, \quad \sum_{i' \neq i} x(i,i',j) + x(i',i,j) \leq y_i, && (3.17) \\
& \forall i, \quad \sum_{i' \neq i} \sum_j x(i,i',j) \cdot c(i,i') \leq B \cdot y_i, && (3.18) \\
& \forall i,i',j, \quad x(i,i',j),\ y_i \in \{0,1\}. && (3.19)
\end{aligned}$$

Constraint (3.18) ensures that the relocation cost of an open facility i, i.e. cost of reassigning its clients in the event i is closed, is bounded by B (the maximum relocation cost). Recall that we are now assuming that we know B, and that B is not a variable (and thus constraint (3.18) is a linear constraint). We show how to find the value of B in Section 3.5.1. Also, constraint 3.5.1 is similar to the *capacity* constraint in *CFLP*. Though our problem is uncapacitated, this constraint introduces flavor of *CFLP*, and thus we will refer to this constraint as *capacity* constraint from now on.

3.3 Rounding the LP solution

Let (x^*, y^*) be an optimal solution to the above LP. We modify the solution in stages and obtain an integer solution at the end. Let us denote the contribution of a client j towards the value of the solution (x^*, y^*), by $\bar{C}_j$, defined as

$$\bar{C}_j = \sum_i \sum_{i' \neq i} x^*(i,i',j) \cdot c_{ij}$$

Let $x^*(i,j) = \sum_{i' \neq i} x^*(i,i',j)$, i.e. $x^*(i,j)$ is the weight of c_{ij} in the weighted sum $\bar{C}_j$. As for the *UFLP* and *CFLP*, Lemma 2.1.1 shows that the facilities serving a client j partially, are such that a large chunk of j is served by facilities not too far away, i.e., for $\alpha \in (0,1)$, and $N_j = \{i : c_{ij} \leq \frac{1}{1-\alpha} \cdot \bar{C}_j\}$, we have, $\sum_{i \in N_j} y_i \geq \alpha$.

3.3.1 Filtering

Due to Lemma 2.1.1, we can modify the solution (x^*, y^*) of the LP to obtain a more structured solution $(\bar{x}, \bar{y})$, just as we did for *UFLP* and *CFLP*. In particular, the solution $(\bar{x}, \bar{y})$ satisfies the following properties, for a fixed α (whose value will be decided later):

$$\forall i, \quad \bar{y}_i = \min(1, y_i^*/\alpha) \tag{3.20}$$

$$\forall i, i', j, \quad \bar{x}(i, i', j) = 0 \quad \text{if } i \notin N_j(\alpha) \tag{3.21}$$

$$\forall i, i', j, \quad \bar{x}(i, i', j) \leq \frac{1}{\alpha} \cdot x(i, i', j) \tag{3.22}$$

$$\forall j, \quad \sum_i \sum_{i' \neq i} \bar{x}(i, i', j) = 1 \tag{3.23}$$

Since the constraint (3.20) asserts that $\bar{y}_i \leq 1$, the *capacity constraint* for $RFLP(B)$ may no longer be satisfied by the solution $(\bar{x}, \bar{y})$. Instead, the following modification of the constraint is satisfied:

$$\forall i \quad \sum_{i' \neq i} \sum_j \bar{x}(i, i', j) \cdot c(i, i') \leq \frac{1}{\alpha} \cdot B \cdot \bar{y}_i, \tag{3.24}$$

Thus, this solution is feasible for $RFLP(B')$, where, $B' = \frac{1}{\alpha} \cdot B$. Also,

$$Cost(\bar{x}, \bar{y}) \leq \frac{1}{\alpha} \cdot Cost(x^*, y^*) \tag{3.25}$$

3.3.2 Clustering and Rounding

The clustering and rounding algorithm we use here follows closely the clustering and rounding algorithm for *CFLP*, as described in Section 2.2.2. In the robust version, a client j is allotted to a pair (i, i') instead of just a single facility, where i is the *primary* facility and i' is the *secondary* facility of j. Our algorithm clusters the primary facilities, and opens suitable secondary facilities so that their opening cost can be charged to the cost of opening the primary facilities, which can be bounded in a fashion similar to that in *CFLP*.

As in Section 2.2.2, let us use the solution $(\bar{x}, \bar{y})$ to obtain a new solution $(\hat{x}, \hat{y})$. In our rounding algorithm, we are going to open pairs of facilities integrally as opposed to opening single facilities. If every client j is served to an extent of at least a $\frac{1}{2}$ by the facility pairs opened integrally, then we are done, because we can double our guess of the maximum relocation cost B' and suitably modify the $\hat{x}$ values without violating any constraints and increasing the relocation and connection costs of the solution by at most a factor of 2.

The clustering algorithm groups the *primary* facilities to form clusters having significant ($\geq \frac{1}{2}$) *primary* facility weight open. Let $\hat{D}$ be the set of clients not served to an extent of at least $\frac{1}{2}$ by the facility pairs opened integrally and $\hat{F}$ be the set of potential *primary* facilities not opened integrally. The algorithm terminates when $\hat{D}$ becomes empty.

Clustering Facilities: Initialize $\hat{D} = \mathcal{D}$, and $\mathcal{J} = \phi$. Also initialize all the $\hat{y}_i$ values to be 0. While $\hat{D} \neq \phi$, repeat the following steps :

1. Pick $j \in \hat{D}$ with minimum $\bar{C}_j$ value. Set $\mathcal{J} \leftarrow \mathcal{J} \cup \{j\}$.
2. Let $S_j = \{i \in \hat{F} : \sum_{i \neq i'} \bar{x}(i, i', j) > 0\}$. S_j is said to be a *cluster* and j is the corresponding *cluster center*. Let D_j be the collection of all those clients in $\hat{D}$ which are served to an extent of at least half by the clusters, i.e. $D_j = \{j' \in \hat{D} : \sum_{j'' \in \mathcal{J}} \sum_{i \in S_{j''}} \sum_{i' \neq i} \bar{x}(i, i', j') \geq \frac{1}{2}\}$. By definition itself, $j \in D_j$.
3. Do: $\hat{D} \leftarrow \hat{D} \setminus D_j$ and $\hat{F} \leftarrow \hat{F} \setminus S_j$

Opening Facilities: For a given facility i, let $s(i)$ denote the facility in $\mathcal{F}$ closest to i. Let $P(S) = \{(i, s(i)) : i \in S\}$, and let the cost associated with a pair $(i, s(i))$ be the distance between i and $s(i)$, i.e., $c_{i,s(i)}$. For every cluster S_j, let O_j be the cheapest $\left\lceil \sum_{i \in S_j} y_i \right\rceil$ pairs in $P(S_j)$. Set $\hat{y}_i = 1\ \forall i$ such that $(i, i') \in O_j$ or $(i', i) \in O_j$ for some $j \in \mathcal{J}$.

Assigning Clients: Solve a transportation problem, where every client j' sends $\sum_{i \in S_j} \sum_{i \neq i'} \bar{x}(i, i', j')$ units of demand to the open facility pairs O_j of only those clusters S_j which have $\bar{C}_j \leq \bar{C}_{j'}$. An open facility pair $(i, s(i))$ accepts at most $\frac{B'}{c_{i,s(i)}}$ units of demand. The solution of this transportation problem defines the $\hat{x}$ values. Lemma 3.4.1 shows that such a transportation problem is feasible.

3.4 Analysis

The Lemma below is similar to the Lemma 2.2.1 in the case of *CFLP*, and proves that the transportation problem so created in the final step of the algorithm is feasible.

Lemma 3.4.1. *Let $\mathcal{T}$ be the following transportation problem: the clients $j' \in \mathcal{D}$ act as suppliers, the set of facility pairs in $\bigcup_{j \in \mathcal{J}} O_j$ opened by the algorithm act as consumers. A supplier $j' \in \mathcal{D}$ has a supply of $\sum_{i \in S_j} \sum_{i' \neq i} x(i, i', j')$ for the clusters S_j which have $\bar{C}_j \leq \bar{C}_{j'}$. Each consumer $(i, s(i)) \in O_j$ has a capacity $\frac{B'}{c_{i,s(i)}}$. $\mathcal{T}$ is a feasible transportation problem, i.e., the total supply for a cluster is at most the total capacity of the cluster.*

Proof. For a cluster S_j with the cluster center j (i.e. $j \in \mathcal{J}$), we have:

$$\begin{aligned}
\text{total capacity} &= \sum_{(i,s(i)) \in O_j} \frac{B'}{c_{i,s(i)}} \\
\text{total supply} &= \sum_{j' \in \mathcal{D} \,:\, \bar{C}_j \leq \bar{C}_{j'}} \sum_{i \in S_j} \sum_{i' \neq i} \bar{x}(i, i', j')
\end{aligned}$$

By constraint (3.18), for a facility $i \in S_j$, we have:

$$\begin{aligned}
B' \cdot \bar{y}_i &\geq \sum_{j' \in \mathcal{D}} \sum_{i' \neq i} \bar{x}(i, i', j') \cdot c_{ii'} \\
&\geq \sum_{j' \in \mathcal{D} \,:\, \bar{C}_j \leq \bar{C}_{j'}} \sum_{i' \neq i} \bar{x}(i, i', j') \cdot c_{ii'} \geq \sum_{j' \in \mathcal{D} \,:\, \bar{C}_j \leq \bar{C}_{j'}} \sum_{i' \neq i} \bar{x}(i, i', j') \cdot c_{i,s(i)}
\end{aligned}$$

Thus the total demand served by a facility $i \in S_j$ is bounded as follows:

$$\text{Demand served by } i \in S_j \;=\; \sum_{j' \in \mathcal{D} \,:\, \bar{C}_j \leq \bar{C}_{j'}} \sum_{i' \neq i} \bar{x}(i, i', j') \;\leq\; \frac{B' \cdot \bar{y}_i}{c_{i,s(i)}}$$

And thus, we have:

$$\text{Total Demand} \;=\sum_{i \in S_j} \sum_{j' \in \mathcal{D} \,:\, \bar{C}_j \leq \bar{C}_{j'}} \sum_{i' \neq i} \bar{x}(i, i', j') \;\leq\; \sum_{i \in S_j} \frac{B' \cdot \bar{y}_i}{c_{i,s(i)}}$$

We just need to show that $\sum_{i \in S_j} \frac{B' \cdot \bar{y}_i}{c_{i,s(i)}} \leq \sum_{(i,s(i)) \in O_j} \frac{B'}{c_{i,s(i)}}$ to show that the transportation problem is a feasible problem. For this, we just need a small mathematical result stated in Lemma 3.4.2. Let $(i_1, s(i_1))$, $(i_2, s(i_2))$, $\ldots$, $(i_k, s(i_{k_0}))$ be an ordering of pairs in O in increasing order of their costs, where $k_0 = |O|$. Assign $\alpha_k = \frac{B'}{c_{(i_k, s(i_k))}}$ and $y_k = \bar{y}_k$.

Using Lemma (3.4.2), we have:

$$\sum_{i \in S_j} \frac{B' \cdot \bar{y}_i}{c_{i,s(i)}} \leq \sum_{(i,s(i)) \in O_j} \frac{B'}{c_{i,s(i)}} \tag{3.26}$$

This shows that the transportation problem is a feasible problem, and its solution is well-defined. The solution to this transportation problem defines the value of $\hat{x}$ variables. $\square$

Lemma 3.4.2. *Let $\alpha_1, \alpha_2, \dots \alpha_n$ be such that $\alpha_k \in \mathbb{R}$ and $\alpha_1 \geq \alpha_2 \geq \dots \geq \alpha_n$ and let $y_k \in [0,1]$, $k \in \{1, 2, \dots, n\}$. Let $k_0 = \lceil \sum_k y_k \rceil$. Then, $\sum_k \alpha_k \cdot y_k \leq \sum_{k \leq k_0} \alpha_k$.*

The next Lemma is analogous to the Lemma 2.2.2, and the proof follows exactly on the same lines.

Lemma 3.4.3. *A client $j' \in \mathcal{D}$ is assigned to an extent of at least half among the clusters, i.e., $\sum_{j \in \mathcal{J}} \sum_{i \in S_j} \sum_{i' \neq i} \hat{x}(i, i', j') \geq \frac{1}{2}$*

Proof. If $j' \in \mathcal{J}$, then by definition of $\hat{D}$, we have that $\sum_{i \in S_{j'}} \sum_{i' \neq i} \hat{x}(i, i', j') \geq \frac{1}{2}$. Thus,

$$\begin{aligned}\sum_{j \in \mathcal{J}} \sum_{i \in S_j} \sum_{i' \neq i} \hat{x}(i, i', j') &= \sum_{j \in \mathcal{J} : \bar{C}_j \leq \bar{C}_{j'}} \sum_{i \in S_j} \sum_{i' \neq i} \hat{x}(i, i', j') \\ &= \sum_{j \in \mathcal{J} : \bar{C}_j \leq \bar{C}_{j'}} \sum_{i \in S_j} \sum_{i' \neq i} \bar{x}(i, i', j') \geq \sum_{i \in S_{j'}} \sum_{i' \neq i} \bar{x}(i, i', j') \geq \frac{1}{2}\end{aligned}$$

Otherwise, $j' \notin \mathcal{J}$. Since j is not added as a cluster center, thus it is being served to an extent of at least half by the clusters created by the time j' was eligible to be considered as a cluster center. All the clusters created so far had cluster centers j such that $\bar{C}_j \leq \bar{C}_{j'}$. Thus, we have,

$$\sum_{j \in \mathcal{J}} \sum_{i \in S_j} \sum_{i' \neq i} \hat{x}(i, i', j') = \sum_{j \in \mathcal{J} : \bar{C}_j \leq \bar{C}_{j'}} \sum_{i \in S_j} \sum_{i' \neq i} \hat{x}(i, i', j') = \sum_{j \in \mathcal{J} : \bar{C}_j \leq \bar{C}_{j'}} \sum_{i \in S_j} \sum_{i' \neq i} \bar{x}(i, i', j') \geq \frac{1}{2}$$

This completes the proof. $\square$

Facility Opening Cost: The algorithm considers every facility at most once, when it lies in a set S_j, for some j. Once considered, the entire set is removed from $\hat{F}$ and hence is not considered again. In a cluster S_j, the algorithm opens $\lceil \sum_{i \in S_j} \bar{y}_i \rceil$ primary facilities, and during this iteration, the algorithm needs these many secondary facilities. Over all the iterations, the number of secondary facilities is at most the number of primary facilities (since a facility can both be a primary facility and a secondary facility for different clients, and also different primaries can have the same secondaries).

Note that a facility is given status of a primary facility only once, but a facility can be declared as a secondary facility in multiple iterations, and also for multiple primary facilities in the same iteration. Since in our assumptions we assumed that opening cost of all facilities is same, hence we can charge the opening cost of secondary facilities to the opening cost of primary facilities.

As $j \in \hat{D}$, we have:

$$\frac{1}{2} \leq \sum_{i \in S_j} \sum_{i' \neq i} x(i, i', j) \ \leq \ \sum_{i \in S_j} y_i$$

Thus S_j has a primary facility weight of at least a half. Thus we have that

$$\lceil \sum_{i \in S_j} y_i \rceil \ \leq \ 2 \cdot \sum_{i \in S_j} y_i$$

Thus, in a particular iteration, cost of opening primary facilities is :

$$f \cdot \lceil \sum_{i \in S_j} \bar{y}_i \rceil \ \leq \ 2 \cdot f \cdot \sum_{i \in S_j} \bar{y}_i.$$

Since the clusters are mutually disjoint and their union may not cover the entire set of facilities $\mathcal{F}$, we have:

$$\sum_i \hat{y}_i \cdot f \ = \ \sum_{j \in \mathcal{J}} \sum_{i \in S_j} \hat{y}_i \cdot f \ \leq \ 4 \cdot \sum_{j \in \mathcal{J}} \sum_i \bar{y}_i \cdot f \ \leq \ 4 \cdot \sum_i \bar{y}_i \cdot f \ \leq \ \frac{4}{\alpha} \cdot \sum_i y_i^* \cdot f \qquad (3.27)$$

Connection Cost: This analysis follows closely the analysis of connection cost in *CFLP*. Due to the filtering step, all facilities in S_j are *close* to each other. We have:

$$\forall i, i' \in S_j, \qquad c_{ii'} \quad \leq \quad c_{ij} + c_{ji'} \quad \leq \quad \frac{2}{1-\alpha} \bar{C}_j. \qquad (3.28)$$

The maximum distance between any client $j' \in \mathcal{D}$ and a facility i such that $(i, s(i)) \in O_j$ (where j is a cluster center) that serves it fractionally (i.e. $\exists i' \neq i$ such that $\hat{x}(i, i', j') > 0$), can be bound in terms of $\bar{C}_{j'}$. This is possible because, $\exists i'' \in S_j$ and $i''' \neq i''$, such that, $\bar{x}(i'', i''', j') > 0$ and thus due to the filtering step, $c_{i''j'} \leq \frac{1}{1-\alpha}\bar{C}_{j'}$, and thus, we have:

$$\forall i \in S_j \qquad c_{ij'} \quad \leq \quad c_{ii''} + c_{i''j'} \quad \leq \quad \frac{2}{1-\alpha}\bar{C}_j + \frac{1}{1-\alpha}\bar{C}_{j'} \quad \leq \quad \frac{3}{1-\alpha}\bar{C}_{j'} \tag{3.29}$$

where the first inequality follows from the metric property, the second inequality follows from (3.28), and the third property follows because of the way the transportation problem was set up.

Therefore, at the end of the rounding algorithm, every client is satisfied to an extent of at least a $\frac{1}{2}$ by integrally open facilities, and the following property holds:

$$\forall i, i' \neq i, j, \qquad \hat{x}(i, i', j) > 0 \quad \Rightarrow \quad c_{ij} \ \leq \ \frac{3}{1-\alpha} \cdot \bar{C}_j \tag{3.30}$$

Thus, as discussed earlier, we are now done, because we can now double our guess of the maximum relocation cost, (set $B' \leftarrow 2 \cdot B'$) and suitably modifying the $\hat{x}$-values, increasing them to at most twice their original value, and satisfying $\sum_i \sum_{i' \neq i} \hat{x}(i, i', j) = 1 \ \forall j$. Since we do not introduce any new non-zero x-value in this process, the property (3.30) still holds, and $(\hat{x}, \hat{y})$ is a feasible solution to $RFLP(B')$ which is $RFLP(\frac{2}{\alpha} \cdot B)$.

The property (3.30) ensures that the connection cost of the solution $(\hat{x}, \hat{y})$ is not very high. The connection cost is:

$$\sum_i \sum_{i \neq i'} \sum_j \hat{x}(i, i', j) \cdot c_{ij} \leq \frac{3}{1-\alpha} \cdot \sum_j \bar{C}_j \cdot \sum_i \sum_{i' \neq i} \hat{x}(i, i', j) = \frac{3}{1-\alpha} \cdot \sum_j \bar{C}_j \tag{3.31}$$

Total Cost: The solution $(\hat{x}, \hat{y})$ has integral $\hat{y}$-values and fractional $\hat{x}$-values. The cost of the optimal LP solution (x^*, y^*) is:

$$Cost(x^*, y^*) \ = \ \sum_i f \cdot y_i \ + \ \sum_j \bar{C}_j \ + \ B \tag{3.32}$$

The cost of our solution $(\hat{x}, \hat{y})$ is:

$$Cost(\hat{x}, \hat{y}) \ \leq \ \frac{4}{\alpha} \cdot \sum_i f \cdot y_i^* \ + \ \frac{3}{1-\alpha} \cdot \sum_j \bar{C}_j \ + \ \frac{2}{\alpha} \cdot B \tag{3.33}$$

Obtaining an integral solution: The $\hat{x}$-values can be thought of as a flow between demands acting as sources and facility pairs acting as sinks. Each demand is sending out a flow of 1, and each facility pair (i_1, i_2) accepts a flow of at most $\frac{2}{\alpha} \cdot B$. Since the flow coming out of every source is integral, thus we know that there is an integral flow between the sources and the sinks that respects the capacities as mentioned above, and that this flow can be found in polynomial time.

3.5 Some Missing Details

There are a few things that remain to be explained.

3.5.1 Choosing B

We try all possible values of B in geometric increments of $(1+\varepsilon)$, for some $\varepsilon \in (0,1)$. Let $B_i = (1+\varepsilon)^i$. Let $IP(B)$ and $LP(B)$ denote the value for the integer and linear programs respectively for this value of B. Let

$$\begin{aligned} B^* &= \operatorname*{argmin}_{B}(IP(B) \ + \ B) \\ B_j &= \operatorname*{argmin}_{i=0,\ldots}(LP(B_i) + B_i) \end{aligned}$$

Let B_k be such that $B^* \le B_k \le (1+\varepsilon)\cdot B^*$. A λ-approximation rounding algorithm for $RFLP(B_j)$, produces an integral solution of cost at most

$$\begin{aligned} &\lambda \cdot (LP(B_j) + B_j) \\ \le\ &\lambda \cdot (LP(B_k) + B_k) \\ \le\ &\lambda \cdot (1+\varepsilon) \cdot (LP(B^*) + B^*) \\ \le\ &\lambda \cdot (1+\varepsilon) \cdot (IP(B^*) + B^*) \end{aligned}$$

and thus yields a $\lambda \cdot (1+\varepsilon)$-approximation algorithm for the problem.

3.5.2 Objective function

In section (3.2), we said that the objective functions (3.10) and (3.14) are not very far apart from each other. Let (x, y) be a feasible solution w.r.t. to the constraints. We need to show that the value of the two objective functions is not very far apart. Let us state the two objective functions here again.

Objective 1:

$$\min \left(\sum_i f_i \cdot y_i + \sum_j \sum_i \sum_{i' \neq i} x(i, i', j) \cdot c_{ij} + \max_i \sum_j \sum_{i' \neq i} x(i, i', j) \cdot (c_{i'j} - c_{ij}) \right) \tag{3.34}$$

Objective 2:

$$\min \left(\sum_i f \cdot y_i + \sum_j \sum_i \sum_{i' \neq i} x(i, i', j) \cdot c_{ij} + \max_i \sum_j \sum_{i' \neq i} x(i, i', j) \cdot c_{i'i} \right) \tag{3.35}$$

Let $F_i(x, y)$, $C_i(x, y)$ and $R_i(x, y)$ represent the facility cost, connection cost and relocation cost of the objective function i for the solution (x, y). Certainly they have the same facility opening costs ($F_1(x, y) = F_2(x, y)$) and same connection costs ($C_1(x, y) = C_2(x, y)$). Their relocation costs differ. Let us assume for the time being that:

$$x(i, i', j) > 0 \Rightarrow c_{ij} \leq c_{i'j} \tag{3.36}$$

Note that (3.36) holds for an (some) integer optimal solution. This is because, assigning j to a nearer open facility makes sense as it may lead to reduction in the cost of solution. The fractional solution, however, may not hold property (3.36). One can convert any fractional solution to a solution that satisfies this property, loosing a factor of 2 in the process. Or, since there is an optimal solution that satisfies this property, one can include constraints of following form in the LP:

$$x(i, i', j) = 0 \quad \text{if} \quad c_{ij} > c_{i'j} \tag{3.37}$$

These constraints do not interfere with our rounding algorithm, and works perfectly when we have a solution to this modified LP. This helps us, because now, we can say:

$$\begin{aligned} R_2(x,y) &= \max_i \left(\sum_j \sum_{i' \neq i} x(i,i',j) \cdot c_{i'i} \right) \leq \max_i \left(\sum_j \sum_{i' \neq i} x(i,i',j) \cdot (c_{ij} + c_{i'j}) \right) \\ &\leq 2 \cdot \max_i \left(\sum_j \sum_{i' \neq i} x(i,i',j) \cdot c_{i'j} \right) \leq 2 \cdot (C_1(x,y) + R_1(x,y)) \end{aligned} \tag{3.38}$$

The last inequality holds because we know from (3.7) that:

$$\begin{aligned} (C_1(x,y) + R_1(x,y)) &= \max_i \left(\sum_j \left(\sum_{i' \neq i} x(i,i',j) \cdot c_{i'j} + \sum_{i_1 \neq i} \sum_{i_2 \neq i_1} x(i_1,i_2,j) \cdot c_{i_1 j} \right) \right) \\ &\geq \max_i \left(\sum_j \sum_{i' \neq i} x(i,i',j) \cdot c_{i'j} \right) \end{aligned}$$

Thus, we have,

$$\begin{aligned} F_1(x,y) + C_1(x,y) + R_1(x,y) &\leq F_2(x,y) + C_2(x,y) + R_2(x,y) \\ &\leq F_1(x,y) + 3 \cdot C_1(x,y) + 2 \cdot R_1(x,y) \end{aligned} \tag{3.39}$$

3.5.3 Approximation Ratio

Let (x^*, y^*) be a solution that minimizes the function (3.34). Since this is a feasible solution to the LP we consider in this chapter (with objective function (3.35)), one can very well start the rounding algorithm with this solution (in which case, terms like $\bar{C}_j$ is defined w.r.t. this solution). We know from (3.33), that we have an integral solution $(\hat{x}, \hat{y})$, such that:

$$\begin{aligned} Cost(\hat{x}, \hat{y}) &\leq \frac{4}{\alpha} \cdot F_2(x^*, y^*) + \frac{3}{1-\alpha} \cdot C_2(x^*, y^*) + \frac{2}{\alpha} \cdot R_2(x^*, y^*) \\ &\leq \frac{4}{\alpha} \cdot F_1(x^*, y^*) + \left(\frac{3}{1-\alpha} + \frac{4}{\alpha} \right) \cdot C_1(x^*, y^*) + \frac{4}{\alpha} \cdot R_1(x^*, y^*) \end{aligned} \tag{3.40}$$

Taking $\alpha = 0.5$, we get a 14 factor approximation. Taking α to be $4 - 2\sqrt{3}$, we get a slightly better approximation ratio of 13.93. This gives a $13.93 \cdot (1 + \varepsilon)^2$-approximation algorithm.

3.6 Extension to $\beta > 1$

The following linear program captures the extension of *RUFLP* $-$ *LP2* (henceforth referred to as *RUFLP-LP2(β)*) to the case when $\beta > 1$. In this linear program, $A \subseteq \mathcal{F}$ and B is a sequence of elements from $\mathcal{F}$. A represents the set of facilities closed by the adversary, and thus $|A| \leq \beta$. B is the extension of the *pairs* we had for the $\beta = 1$ case. B is a set of $\beta + 1$ facilities. In the integer program, $x(B, j) = 1$ means that among all the open facilities, B is the set of the $\beta + 1$ facilities closest to j. Also, $d(j, S)$ represents the distance of j from the closest facility to j in S, i.e. $d(j, S) = \min\{c_{ij} : i \in S\}$. In the linear program below, these constraints that specify the sets A and B are omitted, and are assumed to be clear from the description above.

$$\begin{aligned}
\text{minimize:} \quad & \sum_i f_i \cdot y_i + \max_A \sum_B \sum_j x(B, j) \cdot d(j, B \setminus A) \\
\text{subject to:} \quad & \forall j, \quad \sum_B x(B, j) \geq 1, && (3.41) \\
& \forall i, j, \quad \sum_{B\,:\,i \in B} x(B, j) \leq y_i, && (3.42) \\
& \forall B, j, \quad x(B, j), y_i \geq 0. && (3.43)
\end{aligned}$$

Note that *RUFLP* $-$ *LP2* is a special case of the above LP. As mentioned earlier, once the set of facilities F to open are known, one of the optimal solution is such that a variable $x(i, i', j) = 1$ if and only if i and i' are the two open facilities closest to the client j. For general $\beta \geq 1$, a client j is assigned to one of the $\beta + 1$ open facilities closest to j, no matter what set A of facilities is closed by the adversary.

One can prove a result similar to the Lemma 3.2.1, which shows *RUFLP-LP2(β)* is stronger than *RUFLP* $-$ *LP1*, by using a transformation similar to the one shown in the proof of that lemma.

Lemma 3.6.1. *Any feasible solution (x, y) to RUFLP-LP2(β) of objective function value Cost(x, y) can be converted into a feasible solution $(\bar{x}, \bar{y})$ to RUFLP $-$ LP1 having an objective function value Cost$(\bar{x}, \bar{y})$, such that Cost$(x, y) =$ Cost$(\bar{x}, \bar{y})$.*

Proof. The construction here is a generalization of the construction in the proof of Lemma 3.2.1. Intuitively, an integral solution of *RUFLP* $-$ *LP1* will set a variable $\bar{x}(A, i, j) = 1$

(where $|A| = \beta$) if the client j is assigned to the facility i in the event the set of facilities closed are the facilities in set A. We need to extract those variable of our new LP which are set to 1 in similar circumstances.

Recall that variables in the latter LP are of the form $x(B, j)$, where $|B| = \beta + 1$, and this is set to 1 if B is the set of those $\beta + 1$ open facilities which are closest to the client j. This ensures that no matter what β facilities the adversary closes, there is a facility in the set B, which is the closest facility to j among all open facilities. Thus, to be able to express $\bar{x}(A, i, j)$ in terms of the $x(B, j)$ variables, one must consider those x variables for which the corresponding set B is such that $i \in B$ and, moreover, i is the facility in B which is closest to j when the facilities in A are closed by the adversary. Formally, let $i \notin A$ and $\mathcal{B}(A, i, j) = \{B \subseteq \mathcal{F} : \; |B| = \beta + 1, \; i \in B$ and $\forall i' \in B$, we have, $c_{i'j} < c_{ij} \Rightarrow i' \in A\}$, i.e., the $\mathcal{B}(A, i, j)$ is the collection of all the sets B such that i is the facility in $B \setminus A$ closest to client j. If, for such a B, the variable $x(B, j)$ is 1, then intuitively, $\bar{x}(A, i, j)$ should also be set as 1.

Thus, we obtain the solution $(\bar{x}, \bar{y})$ from (x, y) as follows:

$$\begin{array}{rcl} \forall i, \quad \bar{y}_i & = & y_i \\ \forall A, i \notin A, j, \quad \bar{x}(A, i, j) & = & \displaystyle\sum_{B \in \mathcal{B}(A,i,j)} x(B, j) \\ \forall A, i \in A, j, \quad \bar{x}(A, i, j) & = & 0 \end{array}$$

We need to show that this solution $(\bar{x}, \bar{y})$ is a feasible solution to *RUFLP − LP1*. Let $\mathcal{B} = \{B \subseteq \mathcal{F}' : |B| = \beta + 1\}$. The constraint (3.1) is:

$$\forall \; A, j, \quad \sum_i \bar{x}(A, i, j) \; = \sum_{i \notin A} \; \sum_{B \in \mathcal{B}(A,i,j)} x(B, j) \; = \; \sum_{B \in \mathcal{B}'} x(B, j) \; \geq \; 1$$

where the last inequality follows from the constraint (3.41). We need to argue about the validity of the last equality above, i.e. we need to argue that $\sum_{i \notin A} \sum_{B \in \mathcal{B}(A,i,j)} x(B, j) = \sum_{B \in \mathcal{B}'} x(B, j)$. To show this, we argue that for any given A, and $\forall B \in \mathcal{B}'$, the term $x(B, j)$ appears in the first summation exactly once. Let $\{i_1, i_2, \cdots, i_{\beta+1}\}$ be an ordering of the facilities in B in order of increasing distance from j, and let k be such that $\{i_1, \cdots, i_{k-1}\} \subseteq A$ and $i_k \notin A$. Such a k always exists because B is strictly larger than A. By our definition, $B \in \mathcal{B}(A, i_k, j)$, and now it is easy to argue that $B \notin \mathcal{B}(A, i, j)$ for $i \neq i_k$ (for facilities equidistant from a client, one can break ties arbitrarily, yielding a strict order on facilities with respect to a client). Thus, it follows that $\sum_{i \notin A} \sum_{B \in \mathcal{B}(A,i,j)} x(B, j) = \sum_{B \in \mathcal{B}'} x(B, j)$.

The constraint (3.2) is also satisfied because:

$$\forall\, i, A, j, \quad \bar{x}(A,i,j) \;=\; \sum_{B \in \mathcal{B}(A,i,j)} x(B,j) \;\leq\; \sum_{B \in \mathcal{B}' :\, i \in B} x(B,j) \;\leq\; y_i$$

We are using (3.42) here, together with the simple fact that if $B \in \mathcal{B}(A,i,j)$, then $i \in B$. By construction, constraint (3.3) is satisfied too. The integrality constraints and their linear relaxations also hold. Thus the solution $(\bar{x}, \bar{y})$ is feasible for *RUFLP* $-$ *LP1*. The cost of this solution is:

$$\begin{aligned}
Cost(\bar{x},\bar{y}) &= \sum_{i \in \mathcal{F}} f_i \cdot \bar{y}_i \quad + \quad \max_A \sum_j \sum_i \bar{x}(A,i,j) \cdot c_{ij} \\
&= \sum_{i \in \mathcal{F}} f_i \cdot y_i \quad + \quad \max_A \sum_j \sum_{i \notin A} \bar{x}(A,i,j) \cdot c_{ij} \\
&= \sum_{i \in \mathcal{F}} f_i \cdot y_i \quad + \quad \max_A \sum_j \sum_{i \notin A} \sum_{B \in \mathcal{B}(A,i,j)} x(B,j) \cdot c_{ij} \\
&= \sum_{i \in \mathcal{F}} f_i \cdot y_i \quad + \quad \max_A \sum_j \sum_{B \in \mathcal{B}'} x(B,j) \cdot d(j, B \setminus A) \\
&= Cost(x,y)
\end{aligned}$$

This argument follows through because if $B \in \mathcal{B}(A,i,j)$, then i is the closest facility to j in the set $B \setminus A$, i.e. $c_{ij} = d(j, B \setminus A)$. □

3.6.1 Integrality Gap Example

Byrka, Ghodsi and Srinivasan presented an example instance in [11] which shows that *RUFLP* $-$ *LP1* has an integrality gap of at least $\beta + 1 - \varepsilon$. The example is as follows: There is a single client and n identical facilities, each having a facility cost of 1 and all co-located with the client (i.e. all connection costs are 0). The *RUFLP* $-$ *LP1* can get away by opening each facility fractionally to an extent of $\frac{1}{n-\beta}$, so that the total cost of opening opening facilities $= \frac{n}{n-\beta} \to 1$ as $n \to \infty$. The integral solution, however, has to open $\beta + 1$ facilities, and thus incurs a facility opening cost of $\beta + 1$.

This example, however, fails to be a bad example for the *RUFLP-LP2(β)*. This is because constraint (3.42) prevents any facility from being opened fractionally to a small extent. Any solution (x,y) has a facility opening cost of $\sum_i y_i$, which is at least $\sum_i \sum_{B \,:\, i \in B} x(B,j)$ (j being the single client in the instance) by constraint (3.42). The term $x(B, j)$, for a particular set B, appears $k + 1$ times in the sum above. Thus,

$$\sum_{i} \sum_{B\,:\,i \in B} x(B, j) \;=\; (k+1) \cdot \sum_{B} x(B, j) \;\geq\; (k+1)$$

The last inequality follows from the constraint (3.41). This proves that the facility cost of any solution to the *RUFLP-LP2(β)* is at least $k + 1$, which is in fact same as the facility opening cost of the optimal integral solution.

In Section 3.1, we discussed that we can strengthen the $RUFLP - LP1$ by adding following constraints : $\forall A, i, \quad \sum_j c_{ij} \cdot x(A, i, j) \;\leq\; B \cdot y_i$. The example presented above is still a bad example for even this strengthened linear program because the connection costs in this example are all zero.

Chapter 4

Robust k-median Facility Location Problem

The kind of robustness introduced by Chechik and Peleg in [13] can be extended to other versions of the facility location problems also. In this chapter we consider the robust version of the k-median facility location problem. In this version, the solution is not allowed to open more than k facilities, and of these facilities, the adversary closes at most β, where $\beta < k$.

More formally, the *β-robust k-median facility location problem* ($RFLP(\beta, k)$) is defined as follows: We are given a set of clients $\mathcal{D}$, and a set of facilities $\mathcal{F}$. Every facility $i \in \mathcal{F}$ has a non-negative opening cost f_i associated with it. Every client $j \in \mathcal{D}$ has a non-negative demand d_j which need to be served by one or more open facilities. The cost of serving a unit demand of client j by facility i is denoted by c_{ij}. Given a set F of facilities to be opened, an adversary can close up to β of the open facilities, and the clients are then assigned to the remaining open facilities. The aim is to search for a set of facilities F of size at most k, that minimizes the sum of costs of opening the facilities in F and the costs of assigning the demand of each node to the open facilities that did not fail, for any failure of up to β facilities.

We discuss the metric version of the problem, in which the cost function c_{kl} is defined for $k, l \in \mathcal{F} \cup \mathcal{D}$, and it follows the triangle inequality.

4.1 LP Formulation

In this section, we consider two possible linear programming formulations of this problem. The first one is a natural extension of *RUFLP-LP1* (from [11]) for the k-median version. The second is a natural extension of the *RUFLP-LP2(β)* for this problem. We will present an example to show that the first *LP* has an unbounded integrality gap, and that this example breaks down for the other linear program.

The natural extension of the integer program corresponding to *RUFLP-LP1* for the *β-robust k-median facility location problem* is the following integer program.

$$\text{minimize:} \quad \sum_{i \in \mathcal{F}} f_i \cdot y_i + \max_A \sum_i \sum_j c_{ij} \cdot x(A,i,j)$$

$$\text{subject to:} \quad \forall A, j, \quad \sum_i x(A,i,j) \geq 1, \tag{4.1}$$

$$\forall i, A, j, \quad x(A,i,j) \leq y_i, \tag{4.2}$$

$$\forall A, i \in A, j, \quad x(A,i,j) = 0 \tag{4.3}$$

$$\sum_i y_i \leq k \tag{4.4}$$

$$\forall i, A, j, \quad x(A,i,j),\ y_i \in \{0,1\}. \tag{4.5}$$

Here, $x(A,i,j) = 1$ in an integral solution, if the client j is assigned to facility i when the adversary closes the open facilities in A, where set A has size at most β. The constraints are same as that of *RUFLP-LP1*, with an additional constraint (4.4) which restricts the number of open facilities to at most k. The integrality constraints in the above program can be relaxed to the following:

$$\forall i, A, j, \qquad x(A,i,j) \geq 0 \qquad \text{and} \qquad y_i \geq 0. \tag{4.6}$$

We will henceforth refer to this *LP* as $RFLP(\beta,k) - LP1$. Though the objective function of this linear program is not exactly linear, this can be taken care of assuming that we know the value of $\max_A \sum_i \sum_j c_{ij} \cdot x(A,i,j)$ as B, and including the following constraint in $RFLP(\beta,k) - LP1$:

$$\forall A \qquad \sum_i \sum_j c_{ij} \cdot x(A,i,j) \quad \leq \quad B$$

This removes the max term from the objective function, and thus making it a linear

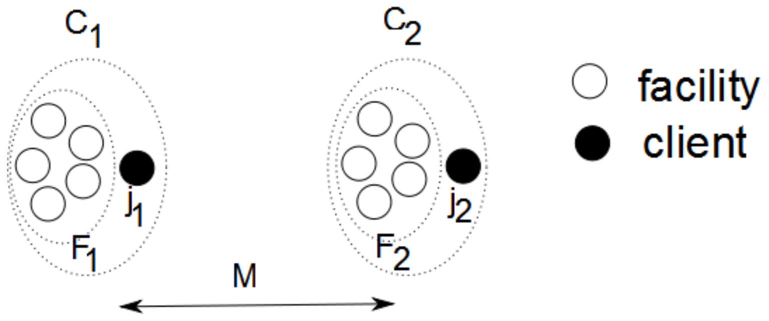

Figure 4.1: An example illustrating unbounded integrality gap for $RFLP(\beta, k) - LP1$

function. It is easy to guess the value of B, by using techniques of Section 3.5.1. Lemma 4.1.1 shows that the above LP has an unbounded integrality gap.

Lemma 4.1.1. *For any β and k, such that $\lfloor \frac{k}{2} \rfloor - 1 < \beta < k - 1$, the linear program $RFLP(\beta, k) - LP1$ has an unbounded integrality gap.*

Proof. Consider the following family of instances (see Fig. 4.1): There are two clusters at a distance M apart. Each cluster has 1 client, and k facilities. Thus in total, there are $2 \cdot k$ facilities, and 2 clients. The opening cost of each facility is 0 (i.e. $f_i = 0 \;\forall\, i$). Following is an optimal solution to $RFLP(\beta, k) - LP1$ of cost 0 :

$$\begin{aligned} \forall i, \qquad y_i &= \frac{1}{2} \\ \forall A, i, j, \qquad x(A, i, j) &= \frac{1}{2} \quad \text{if } c_{ij} = 0 \text{ and } i \notin A \\ &= 0 \quad \text{if } c_{ij} = M \text{ or } i \in A \end{aligned}$$

The solution clearly satisfies constraints (4.2) and (4.3). The total number of facilities opened is $2 \cdot k \cdot \frac{1}{2} = k$, and thus the solution satisfies constraint (4.4). When the adversary closes β facilities, there are at least 2 facility locations that remain open at each cluster and thus the solution satisfies constraint (4.1). Here, each cluster has, in total, at least 1 facility open. Each client is served by the facilities in its own cluster, no matter what facilities are closed by the adversary. Hence the solution above is a feasible solution and thus indeed optimum.

An integral solution can avoid the extra connection of cost M by opening enough facilities in each cluster such that irrespective of the facilities closed by the adversary, each cluster has at least one open facility. This is only possible if the integral solution opens at least $\beta+1$ facilities in each cluster, which is not possible, because $2\cdot(\beta+1) > 2\cdot(\frac{k}{2}-1+1) = k$. Hence any integral solution opens less than $\beta+1$ facilities in at least one of the clusters, and thus incurs a cost of M. $\square$

The natural extension of the integer program corresponding to *RUFLP-LP2(β)* is the following integer program:

$$\begin{aligned}
\text{minimize:}\quad & \sum_i f_i \cdot y_i + \max_A \sum_B \sum_j x(B,j)\cdot d(j, B\setminus A) & \\
\text{subject to:}\quad & \forall j, \quad \sum_B x(B,j) \geq 1, & (4.7)\\
& \forall i,j, \quad \sum_{B\,:\,i\in B} x(B,j) \leq y_i, & (4.8)\\
& \sum_i y_i \leq k & (4.9)\\
& \forall B,j, \quad x(B,j), y_i \in \{0,1\}. & (4.10)
\end{aligned}$$

The integrality constaints in the above program can be relaxed to the following:

$$\forall i,B,j, \qquad x(B,j) \;\geq\; 0 \qquad \text{and} \qquad y_i \;\geq\; 0. \qquad (4.11)$$

We will henceforth refer to the resulting linear program as $RFLP(\beta,k)-LP2$. The variables hold the same meanings as earlier. The only addition here is the constraint that limits the number of open facilities to at most k. Though the objective function is not linear, it can be easily made linear by guessing the value of the max term, and introducing it as a new constraint (in the same way as discussed for $RFLP(\beta,k)-LP1$).

We now show that the example produced in the proof of Lemma 4.1.1 is not a bad example for $RFLP(\beta,k)-LP2$.

Let (x,y) be a feasible solution to $RFLP(\beta,k)-LP2$. Let C_1 and C_2 be the two clusters with clients j_1 and j_2 and facilitiy sets F_1 and F_2 respectively.

Since $\sum_i y_i \leq k$, thus, one of the clusters has at most $\frac{k}{2}$ facility weight opened by the solution (x,y). Without loss of generality, let C_1 be such that $\sum_{i\in F_1} y_i \leq \frac{k}{2}$. Then, we have,

$$\frac{k}{2} \geq \sum_{i \in F_1} y_i \geq \sum_{i \in F_1} \sum_{B\,:\,i \in B} x(B, j_1) \geq \sum_{i \in F_1} \sum_{B \subseteq F_1\,:\,i \in B} x(B, j_1) = (\beta + 1) \cdot \sum_{B \subseteq F_1} x(B, j_1)$$

And thus, we have,

$$\sum_{B : B \cap F_2 \neq \phi} x(B, j_1) \;\geq\; 1 \;-\; \frac{k}{2 \cdot (\beta + 1)} \;>\; 0$$

where the last inequality follows from the assumption that $\left\lfloor \frac{k}{2} \right\rfloor - 1 < \beta$. This shows that the cost of the fractional solution to *RFLP*(β, k) − *LP2* is not 0. We still need to show that the cost of such a solution is not very small as compared to the optimal integral solution. We need to come up with a scenario A, whose removal results in a high relocation cost. Since $|B| = \beta + 1$, thus $|B \cap F_1| \leq \beta \Rightarrow B \cap F_2 \neq \phi$. There are $\binom{k}{\beta}$ different subsets of F_1 of size β. We have,

$$\sum_{S \subset F_1,\, |S| = \beta} \sum_{B : B \cap F_1 \subseteq S} x(B, j_1) \;\geq\; \sum_{B : B \cap F_2 \neq \phi} x(B, j_1) \;\geq\; 1 \;-\; \frac{k}{2 \cdot (\beta + 1)}$$

which means there is a set $S_0 \subset F_1$, with $|S_0| = \beta$, such that

$$\sum_{B : B \cap F_1 \subseteq S_0} x(B, j_1) \;\geq\; \frac{1}{\binom{k}{\beta}} \cdot \left(1 \;-\; \frac{k}{2 \cdot (\beta + 1)}\right)$$

which shows that the cost of the optimal fractional solution is at least $\frac{1}{\binom{k}{\beta}} \left(1 \;-\; \frac{k}{2 \cdot (\beta+1)}\right) M$. For the special case of $\beta = k - 2$, the cost of the optimal fractional solution is at least $O(\frac{M}{k^3})$.

4.1.1 Rounding Algorithm

We tried to round the linear program *RFLP*(β, k) − *LP2* to obtain an integral solution not very far from the optimal fractional solution. One of the ways to approach this is to attempt a rounding procedure similar to that of classical k-median problem as presented by [12]. As such, normal filtering cannot be applied to the k-median problem, unless one is willing to sacrifice the requirement that no more than k facilities should be opened. Thus the algorithm proceeds by first clustering clients together. One of the main assumptions

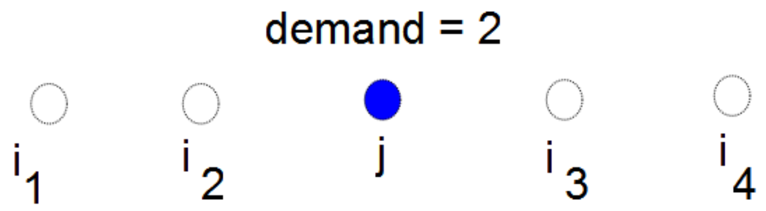

Figure 4.2: An example illustrating that co-located clients may have different assignments

that makes the algorithm work is that co-located clients are served by the same set of open facilities. This is certainly true in the classical k-median problem, because co-located clients have the same set of facilities closest to their location, and thus all demands located at the same place can easily be served by the same open facilities (there being no constraints on the amount of demand that can be served by an open facility). This, however, fails to hold in the robust version of the k-median problem. This is because in the robust version, it helps to distribute clients to numerous facilities as it keeps the maximum relocation cost incurred, in the event of closing of facilities, minimal. The following simple example illustrates this:

The location j has 2 demands, and there are 4 facilities, i_1, i_2, i_3, i_4, where the facilities and demands can be thought of as located on a number line, with i_1 at -2, i_2 at -1, j at origin, i_3 at 1 and finally i_4 at 2. Let us say $k = 4$ and $\beta = 1$, and all facility opening costs are 0. Then the optimal solution will open all the four facilities, and allot one demand to i_2 and one to 1_3, giving a connection cost of 2. In case i_2 gets closed, its demand is allocated to i_1 and when i_3 is closed, its demand is allocated to i_4. This gives a maximum relocation cost of 1, and thus cost of this solution is 3. If we assign both the demands to i_2, then the connection cost is 2 and the maximum relocation cost is also 2. This gives a solution of cost $4 > 3$. Hence it is not necessary that co-located clients are served by the same set of facilities.

Chapter 5

Stochastic Steiner Tree

Robustness can be introduced in networks in multiple ways. Some ideas of robustness aim at designing networks that are resilient to failures of components, while others incorporate other notions of uncertainties in the network. Stochastic network design models the uncertainties via a probability distribution over the possible scenarios, and this enables one to work with the expected cost of the solution with regard to this probability distribution. Among the more popular models is that of *two stage stochastic optimization with recourse* in which only distributional data is made available in the first stage, and the user is allowed to build a basic low cost network to avoid high costs in the second stage when the data is made known fully but the prices are inflated.

Many variants of this model deal with the nature of the scenario set and how the scenario set is made available to the user. The scenario set can be *finite*, in which case one can list the probability distribution (although this may take exponential space), or *infinite*, in which case it may be difficult to specify the probability distribution concisely. In a *black box model*, the user is allowed to make a polynomial number of queries to a *black box*, and each such query returns a scenario drawn according to the probability distribution. These samples are then used to frame an approximation to the actual probability distribution. Another variant specifies the probability with which each component occurs in the scenario that is presented to the user in the second stage and assumes a product combination; that is, the probability of a scenario occurring is just the product of the probabilities of occurrence of the components in the scenario and the probabilities of non-occurrence of the components not in the scenario (assuming independence). Other variants of this model deal with the inflation in costs that occurs in the second stage. One can assume a uniform inflation across all scenarios, or one can assume a different inflation factor for each scenario.

One can even generalize further and assume separate inflation factors for each component and each scenario.

In this chapter, we consider the stochastic version of the *Steiner tree problem (SST)*, which seeks to find a tree spanning a given subset of the vertex set. The vertices in this subset are called *terminals*. The cost of a solution is the sum of the costs of the edges in this network. In the two-stage stochastic Steiner tree problem, the set of terminals is not known beforehand. In the first stage, only a probability distribution over possible scenarios is known, where a scenario consists of a set of terminals that may come up in the second stage. In the second stage, the actual scenario, i.e. the set of terminals to be spanned by the network is revealed, but the edges are more expensive by an *inflation factor* in the second stage as compared to the first stage. Thus, we are allowed to build a partial network in the first stage at a lower cost, and in the second stage, when the set of terminals become known, we must build a *recourse* network to satisfy this set, but incurring a higher cost. A closely related problem is the stochastic version of the *Steiner forest problem (SSF)*, in which the scenarios are collection of source sink pairs.

Approximation algorithms for SST come in two flavors. In the *rooted* version of the problem, it is assumed that all the scenarios share a common vertex which is called as the *root*. In the *unrooted* version, such a common vertex is absent. This distinction is non-trivial, and normally algorithms for the unrooted versions are powerful enough to solve problems like Multicommodity Rent or Buy problem [23]. Gupta and Pál [23] gave the first constant factor approximation algorithm for SST . Fleischer, Könemann, Leonardi and Schäfer [18] gave an improved 6-approximation algorithm. All these algorithms are in the *black box model*, where polynomial number of queries are made to a black box oracle to estimate the probability distribution on the scenarios. For the rooted SST, Gupta, Pál, Ravi and Sinha [24] gave a 3.55-approximation algorithm for the black box model. They also give a 8-approximation algorithm for the SSF in the *independent decision model* (i.e. a pair (s_i, t_i) may require to be connected with a probability π_i independently of all other pairs). For the general black box oracle model, nothing was known until recently, when Gupta and Kumar [21] gave the first (and so far the only) constant approximation algorithm for the SSF problem. The techniques used in their work are built upon the basic ideas of work on SST by Gupta, Ravi and Sinha in [25], albeit the modifications required are very involved and difficult to comprehend.

The discussion in this chapter is mainly based on the work of Gupta, Ravi and Sinha in [25]. Our aim was to be able to come up a cleaner way of presenting Gupta and Kumar's algorithm for the SSF, based on this earlier work on SST. Though we were unable to come up with such a refinement, we were able to obtain a cleaner presentation of Gupta, Ravi and Sinha's algorithm on SST, and in the process improved its performance guarantee (though there are algorithms with better approximation factors based on cost shares). We assume a *finite scenario* model, i.e. the set of terminals in the second stage is one of a finite set of distinct and specified sets. We present a 20-approximation algorithm for the rooted SST (in finite scenario model) which is an improvement to the 40-approximation algorithm of Gupta, Ravi and Sinha in [25].

5.1 Problem Definition

Let $G = (V, E)$ be an undirected graph, with a cost function $c: E \rightarrow \mathbb{R}^+$. $c(e)$ denotes the cost of an edge e in the first stage. The *root vertex* is labeled as r. Let $c(H) = \sum_{e \in H} c(e)$ denote the cost of a subgraph H of G. Given a terminal set R with $r \in R$, let $span(R)$ denote the set of trees that are subgraphs of G that span R. We work with the finite scenario model here, i.e. there are m scenarios and the k^{th} scenario is specified by a set of terminals S_k, a probability of occurrence p_k, and an inflation factor σ_k, which indicates that if scenario k comes up in the second stage, then every edge is σ_k times more expensive as compared to its first-stage cost. We have, $\sum_k p_k = 1$, because exactly one scenario materializes in the second stage. The aim is to find a set of edges E_0 so as to minimize :

$$c(E_0) + \mathbf{E}[\sigma_R c(E_R)] \text{ such that } E_0 \cup E_R \in \text{span}(R)$$

which can be written as:

$$c(E_0) + \sum_{k=1}^{m} p_k \cdot \sigma_k \cdot c(E_k) \text{ such that } E_0 \cup E_k \in \text{span}(S_k)\ \forall k \tag{5.1}$$

where E_k is the set of recourse edges bought in the second stage to satisfy the scenario S_k. Note that once S_k and the set of edges bought in first stage E_0 are known, the set E_k can be computed easily by (approximately) solving a suitable Steiner tree instance.

As is standard and noted in [25], the following assumptions can be made without loosing the generality of the problem:

- The cost function c follows the *triangle inequality*.
- Each terminal occurs in at most one scenario S_k, i.e. the sets $\{S_k : k = 1, \ldots, m\}$ are mutually disjoint.

Because the cost function c follows the triangle inequality, it can very well be viewed as a distance function, i.e. as a metric. Thus, throughout this chapter the terms *distance* and *cost* are used interchangeably.

5.2 LP formulation

Gupta, Ravi and Sinha [25] extend the undirected cut formulation of the deterministic version of the Steiner tree problem to obtain the following integer program:

$$\begin{aligned}
\text{minimize:} \quad & \sum_e c(e) \cdot x_e^0 + \sum_{k=1}^{m} p_k \cdot \sigma_k \sum_e c(e) \cdot x_e^k \\
\text{subject to:} \quad & \forall S : r \notin S,\ S \cap S_k \neq \phi,\ \forall k, \quad \sum_{e \in \delta(S)} \left(x_e^0 + x_e^k\right) \geq 1, && (5.2) \\
& \forall k,\ e \in E, \quad x_e^k \geq 0 && (5.3) \\
& \forall k,\ e \in E, \quad x_e^k \in \mathbb{Z}. && (5.4)
\end{aligned}$$

Here, $x_e^0 = 1$ indicates that edge e is bought in the first stage. Also, $x_e^k = 1$ indicates that edge e is bought in the second stage when the scenario S_k arises in the second stage. A linear relaxation of this integer program would drop the constraint (5.4). Let us call the resulting linear program as SST-LP1. The set of edges bought in the first stage is the set $E_0 = \{e : x_e^0 = 1\}$. In general, as shown in [25], for an optimal solution to the above integer program, the set E_0 need not form a tree. However, Gupta et al in [25] also show that one can transform a solution (x^0, x^k) to the above integer program, into a solution $(\bar{x}^0, \bar{x}^k)$, such that the first-stage edges in this new solution form a tree containing the root vertex r, and $Cost(\bar{x}^0, \bar{x}^k) \leq 2 \cdot Cost(x^0, x^k)$. Thus, by loosing a factor of 2, one can obtain this nice structure to the edges bought in the first stage that they form a tree containing the root vertex r.

A consequence of the above observation is that, in the final solution, the path from a terminal t to the root r is composed of second-stage edges followed by first-stage edges. In other words, the path p_t connecting terminal t to root r in the final network is such that there is a vertex v_t, called as the *transition vertex*, on this path p_t such that all edges e on the path $p_t(t, v_t)$ are edges bought in the second stage (i.e. $x_e^k = 1$), and all edges e' on the path $p_t(v_t, r)$ are edges bought in the first stage (i.e., $x_{e'}^0 = 1$). This property is referred to as the *monotonicity property*, and helps us view the stochastic Steiner tree problem in a slightly different way.

Let t be a terminal in some scenario S_k (i.e. $t \in \cup_{k=1}^{m} S_k$), and let P_t be the set of all simple paths with t and r as end points. Let $p \in P_t$, and let v be a vertex on this path p. Let $f(p,v) = 1$ if the terminal t uses this path p to connect to root (in case its scenario gets chosen), and vertex v is the transition vertex, i.e. all edges on the path $p(t,v)$ are second-stage edges, and all edges on path $p(v,r)$ are first-stage edges (where $p(a,b)$ represents the portion of the path p with end points at a and b, where a and b are vertices on the path p). With this notation in place, the variables of the above integer program translate to following:

$$\forall e \in E,\ t \in S_k,\ \forall k, \quad x_e^0 \geq \sum_{p \in P_t,\ v \in p\colon\ e \in p(v,r)} f(p,v) \tag{5.5}$$

$$\forall e \in E,\ t \in S_k,\ \forall k, \quad x_e^k \geq \sum_{p \in P_t,\ v \in p\colon\ e \in p(t,v)} f(p,v) \tag{5.6}$$

The variable x_e^0 is the maximum flow sent on this edge by any scenario as a first-stage flow. Thus, the integer program can be written as:

$$\text{minimize:} \quad \sum_e c(e) \cdot x_e^0 + \sum_{k=1}^{m} p_k \cdot \sigma_k \sum_e c(e) \cdot x_e^k$$

$$\text{subject to:} \quad \forall S : r \notin S,\ S \cap S_k \neq \phi,\ \forall k, \quad \sum_{e \in \delta(S)} \left(x_e^0 + x_e^k\right) \geq 1, \tag{5.7}$$

$$\forall t \in S_k,\ \forall k, \quad \sum_{p \in P_t, v \in p} f(p,v) = 1 \tag{5.8}$$

$$\forall e \in E,\ t \in S_k,\ \forall k, \quad x_e^0 \geq \sum_{p \in P_t,\ v \in p\colon\ e \in p(v,r)} f(p,v) \tag{5.9}$$

$$\forall e \in E,\ t \in S_k,\ \forall k, \quad x_e^k \geq \sum_{p \in P_t,\ v \in p\colon\ e \in p(t,v)} f(p,v) \tag{5.10}$$

$$\forall k,\ t \in S_k,\ p \in P_t\ v \in p, \quad f(p,v) \in \{0,1\}. \tag{5.11}$$

The constraints (5.8) - (5.10) imply the constraint (5.7), but it is included in the integer program just for the sake of clarity. The linear programming relaxation of the above integer program modifies the last constraint to the following constraint:

$$\forall k,\ t \in S_k,\ p \in P_t,\ v \in p, \quad 0 \le f(p,v) \le 1$$

Let us call the resulting linear program as SST-LP2. As already discussed, cost of the optimal solution to SST-LP2 is at most twice the cost of the optimal solution to SST-LP1 as proved in Lemma 3.1 in [25].

5.3 Rounding Algorithm

The fractional solution to the above linear program can be thought of as flow from each terminal to the root. A terminal t sends fractional flows along various paths in P_t, and each such path p has a transition vertex $v \in p$. The values of x_e^0 and x_e^k are suitably obtained from these flow values. Once these flow values are known, the constraints (5.9) and (5.10) fully specify the solution to the linear program.

The rounding algorithm we present here closely follows the rounding algorithm of [25], but with a somewhat cleaner and simpler exposition, which also results in an inmproved approximation factor. The algorithm is divided in roughly three phases. In the first phase, each scenario builds a part of its second-stage solution, which is a Steiner forest. In the second phase, we build the first-stage tree T_0, and finally in the third phase, some scenarios extend their second-stage solution to connect the forest built in the first phase to the tree T_0. We now describe each phase in detail, and simultaneously prove that the cost of the network built in that phase can be paid for by the appropriate portions of the solution to the linear program SST-LP2.

5.3.1 Phase 1: Second-Stage Component Growth

We construct a second-stage solution for each scenario using the second-stage variables x_e^k for scenario k, and the flow values $f(p,v)$ for $p \in P_t$ and $v \in p$. For the sake of clarity, let us define some notation. For a terminal t and a set S containing t and not containing r, let $F_i(t,S)$ be the flow on paths in P_t which have their transition vertex *inside* the set S, and let $F_o(t,S)$ be the flow on paths in P_t which have their transition vertex *outside* the

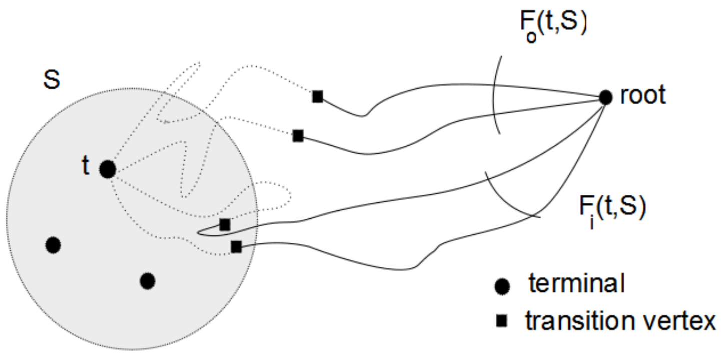

Figure 5.1: $F_i(t, S)$ and $F_o(t, S)$

set S, i.e.

$$F_i(t,S) := \sum_{p\in P_t,\ v\in p,\ v\in S} f(p,v)$$
$$F_o(t,S) := \sum_{p\in P_t,\ v\in p,\ v\notin S} f(p,v)$$

We need to define a notion of *active* and *inactive* sets for our primal dual algorithm. Here, *moat* refers to subset of vertices that can grow to include other vertices during the algorithm. Let $\alpha > 1$ be a fixed real number, whose value we will determine later.

Definition 5.3.1 (**Inactive, Active**)**.** Let set S be such that $S \cap S_k \neq \phi$ and $r \notin S$, for some scenario k. Then S is said to be:

- *Inactive*: $\exists t \in S \cap S_k$, such that $F_i(t, S) \geq \frac{1}{\alpha}$.
- *Active*: $\forall t \in S \cap S_k$, $F_i(t, S) \leq \frac{1}{\alpha}$, or equivalently, $F_o(t, S) \geq 1 - \frac{1}{\alpha}$.

□

The following lemma relates the flow values to the x-variables, and also highlights a *monotonicity* property which we will exploit in the primal dual algorithm. The results of the lemma are straightforward to derive using the definitions of the respective terms, and are stated here for the sake of completion and clarity to the reader, because they play a key role in the algorithm.

Lemma 5.3.2. *For a scenario k, let sets S and T be such that they separate some terminals in S_k from root r, i.e. $r \notin S$, $r \notin T$, $S \cap S_k \neq \phi$, $T \cap S_k \neq \phi$, and let $S \subseteq T$. Then the following results hold:*

1. $\sum_{e \in \delta(S)} x_e^k \geq F_o(t, S)$
2. $\sum_{e \in \delta(S)} x_e^0 \geq F_i(t, S)$
3. $F_i(t, S) \leq F_i(t, T)$
4. $F_o(t, S) \geq F_o(t, T)$
5. *If S is an inactive set, then T is also an inactive set.*
6. *If T is an active set, then S is also an active set.*

Given a set of valid moats $\mathcal{M}_k$, let $\mathcal{A}_k$ be the set of active moats, and $\mathcal{I}_k$ be the set of inactive moats in $\mathcal{M}_k$. Our aim is to grow some active moats in a primal-dual algorithm and to build some second-stage forests for each scenarios. We will then connect all these forests by a first-stage tree containing the root. The advantage of doing things this way is that the building of the first-stage trees continues from where the second-stage construction left off, making the algorithm more streamlined and, in essence, very similar to the primal dual algorithm for Steiner tree, where we grow moats from each terminal.

Following the primal dual treatment of the stochastic Steiner tree algorithm in [2], [25], which is mimicked by the Steiner forest algorithm in [19], we start with the collection of moats $\mathcal{M}_k$ to be the *minimal active sets*, i.e. the singleton sets comprised of the terminals in S_k. The dual variables z_S are initialized to 0 for all active sets S. Also, $\mathcal{A}_k$ and $\mathcal{I}_k$ be the corresponding active and inactive moats in $\mathcal{M}_k$. We also maintain a set of *special* moats $\mathcal{G}_k$, which are active sets but they stop growing because they have enough dual to connect to the network of scenario k. Every moat $M \in \mathcal{G}_k$ would have enough dual to connect to another moat M' which will be called its *parent*. $\mathcal{G}_k$ is initialized to ϕ. At any time, $\mathcal{A}_k$, $\mathcal{I}_k$ and $\mathcal{G}_k$ are mutually disjoint and it holds that $\mathcal{A}_k \cup \mathcal{I}_k \cup \mathcal{G}_k = \mathcal{M}_k$. As is standard in primal-dual algorithms, we introduce a notion of time τ, which is initialized to 0.

Growing of moats: Only active moats *grow*. We increase the dual variables for the active sets in $\mathcal{A}_k$, uniformly with time, at the same rate, till an edge becomes tight. An edge e is said to be tight if the total dual of all the active sets incident on this edge sums up to the cost of this edge, i.e. $\sum_{S:S \text{ is active}, \, e \in \delta(S)} z_S = c(e)$. The following possibilities arise:

- Edge e is incident to two active moats M, M'. Remove them from $\mathcal{M}_k$ and $\mathcal{A}_k$. Let $M'' = M \cup M'$. Insert M'' in $\mathcal{M}_k$ and also in the appropriate set $\mathcal{A}_k$ or $\mathcal{I}_k$ depending on whether M'' is active or inactive.
- Edge e is incident to an active moat M and a non-growing moat M', i.e. $M' \in \mathcal{I}_k \cup \mathcal{G}_k$. Remove M from $\mathcal{A}_k$ and insert it in $\mathcal{G}_k$. Mark M' as the parent of M.
- Edge e is incident to just one active moat M, and no other moats. Then M simply grows to include both the end points of the edge e. If M becomes inactive after this growth, it is removed from $\mathcal{A}_k$ and inserted in $\mathcal{I}_k$.

Termination time: For every non-growing moat $M \in \mathcal{I}_k \cup \mathcal{G}_k$, we also record the time τ_M which is the value of τ when the moat M stopped growing.

This moat growing process finishes when there are no more active moats, i.e. $\mathcal{A}_k = \phi$. The primal dual process of [2] and [19] grows moats precisely in the same way as we have described it here, and it simultaneously constructs a Steiner tree T_M connecting the terminals in $S_k \cap M$ for a moat M. The following result follows from their analysis:

Lemma 5.3.3. *For any non-growing moat $M \in \mathcal{I}_k \cup \mathcal{G}_k$, at any stage of the primal dual process, we have, $2 \cdot \tau_M + \sum_{e \in T_M} c(e) \leq 2 \cdot \sum_{S \subseteq M} z_S$.*

Also, following result is similar to the Lemma 3.4 in [25]. This result charges the cost of building these Steiner trees of individual moats to the second-stage variables x_e^k.

Lemma 5.3.4. *For any scenario S_k, we have, $\sum_{S \subseteq V} z_S \leq \left(\frac{\alpha}{\alpha-1}\right) \cdot \sum_e c(e) x_e^k$*

Proof. Consider the problem of covering all the active sets that are subsets of some non-growing moat $M \in \mathcal{I}_k \cup \mathcal{G}_k$, i.e. find a collection of edges of minimum cost, such that every active set is incident to at least one edge from the collection. This can be cast as a linear program. Let $\mathcal{A}$ be the collection of active sets which are subset of some non-growing moat M, i.e. $\mathcal{A} = \{S \subseteq M \mid S \text{ is an active set}, M \in \mathcal{I}_k \cup \mathcal{G}_k\}$. The primal program (*LP1*) is:

$$\begin{aligned} \text{minimize:} \quad & \sum_e c(e) \cdot y_e \\ \text{subject to:} \quad & \forall S \in \mathcal{A}, \quad \sum_{e \in \delta(S)} y_e \geq 1, \\ & \forall e, \quad y_e \geq 0. \end{aligned}$$

The corresponding dual program (*LP2*) is:

$$\begin{aligned} \text{maximize:} \quad & \sum_{S \in \mathcal{A}} \hat{z}_S \\ \text{subject to:} \quad & \forall e, \quad \sum_{S \in \mathcal{A}:\, e \in \delta(S)} \hat{z}_S \le c(e), \\ & \forall S \in \mathcal{A}, \quad \hat{z}_S \ge 0. \end{aligned}$$

By Lemma 5.3.2, and definition of *active* sets, for a set $S \in \mathcal{A}$ and a terminal $t \in S$, we have,

$$\sum_{e \in \delta(S)} x_e^k \;\ge\; F_o(t, S) \;\ge\; 1 - \frac{1}{\alpha} \quad \Rightarrow \quad \sum_{e \in \delta(S)} \left(\frac{\alpha}{\alpha - 1} \right) \cdot x_e^k \;\ge\; 1$$

And thus, $y_e = \left(\frac{\alpha}{\alpha-1}\right) \cdot x_e^k$ is a feasible solution to *LP1*. Also, the dual z_S grown by our primal dual process is a feasible solution to the dual program *LP2* because we never violate any edge constraints when we construct these z-variables. Thus $\forall S \in \mathcal{A}$, $\hat{z}_S = z_S$ is a feasible solution for the dual program. Thus, by weak duality, we have:

$$\sum_{S \subseteq V} z_S \;=\; \sum_{S \in \mathcal{A}} z_S \;\le\; \left(\frac{\alpha}{\alpha - 1} \right) \cdot \sum_e x_e^k$$

The first equality holds because only active sets get non-zero dual values in the primal dual process. □

We will also need following lemma in coming phases about the above moat growing process.

Lemma 5.3.5. *Let M be a moat in $\mathcal{A}_k$ at time τ. Then, for every vertex $v \in M$, there is a terminal $t \in M$, such that $d(v, t) \le \tau$. Thus, for a moat $M \in \mathcal{I}_k \cup \mathcal{G}_k$ and a vertex $v \in M$, there is a terminal $t \in M$, such that $d(v, t) \le \tau_M$.*

Proof. Let M be an active moat at time τ, and R_M be the collection of all the terminals in the moat M. The proof of this result follows from the fact that $M = \bigcup_{t \in R_M} B(t, \tau)$, where $B(t, \tau)$ is the collection of all the vertices that are a distance of at most τ from t. □

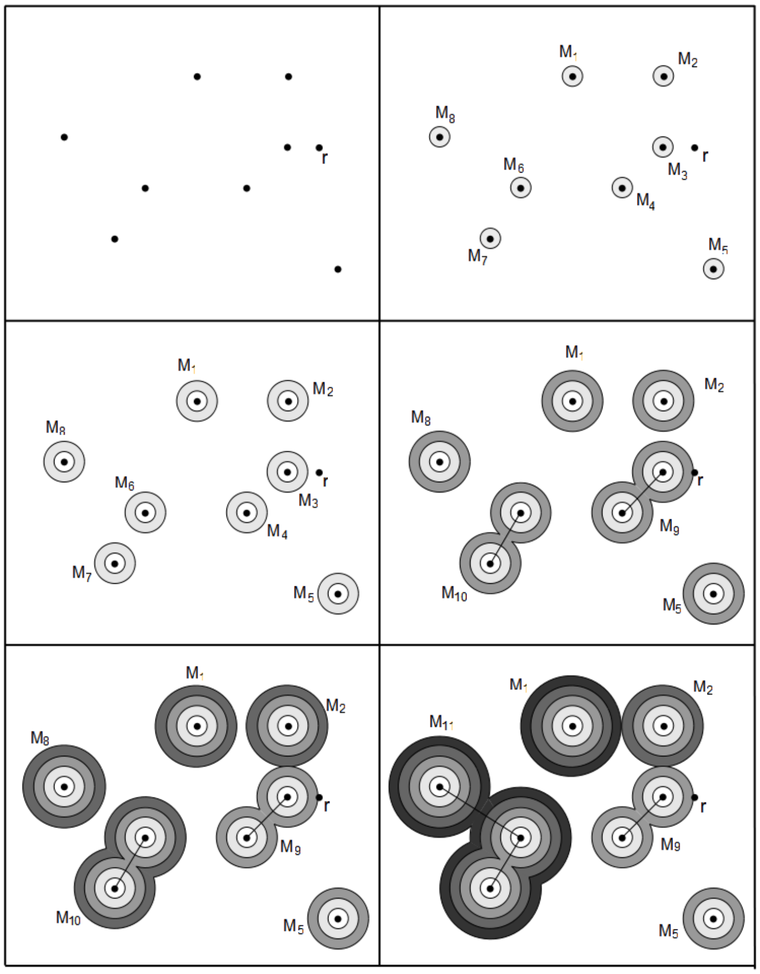

Figure 5.2: Growing moats for a scenario k, building a part of the second-stage network for this scenario. At the end of the primal dual process, $\mathcal{A}_k = \phi$, $\mathcal{I}_k = \{M_5, M_9, M_{11}\}$ and $\mathcal{G}_k = \{M_1, M_2\}$.

5.3.2 Phase 2: First-Stage Tree

We can now construct our first-stage tree. The idea is to continue from the inactive moats created by the second-stage component growth. Let $\mathcal{I} = \cup_{k=1}^{m} \mathcal{I}_k$, be the collection of all the inactive moats at the end of the initial primal dual algorithm on each scenario. The moats in a particular set $\mathcal{I}_k$ are disjoint and form a valid set of moats, but this is no longer true for $\mathcal{I}$, because moats from two different scenarios may overlap. Thus we need to choose a set of *representative moats* which are disjoint (and hence form a valid collection of moats), contract them and build a Steiner tree T on top of the contracted moats which will be the first-stage tree. Having obtained T, we will need to do two things:

- Expand each contracted moat in the *representative moat set*, and connect the second-stage Steiner tree of these moats to the tree T using first-stage edges. This gives us the first-stage tree T_0.
- *Recourse second-stage paths*: For all the moats not in the representative set, we will need to expand their second-stage solution by adding second-stage paths that connect their Steiner trees to the tree T_0.

To be able to bound the cost of each of these steps, the set of *representative* moats need to be chosen wisely. As in [25], we will choose this set so that the cost of the first task listed above can be bounded in terms of the cost of the tree T_0, and the cost of the second task above can be bounded in terms of the second-stage variables x_e^k. We will need notion of *diameter* of a moat and *distance* between moats:

Definition 5.3.6. Diameter $d(M)$ of a moat $M \in \mathcal{I}$ is defined as:

$$d(M) := 2 \cdot \tau_M + \sum_{e \in T_M} c(e)$$

□

Definition 5.3.7. Distance $d(M, M')$ between two moats $M, M' \in \mathcal{I}$ is defined to be the distance between closest vertices in M and M', i.e.

$$d(M, M') := \min\{d(u, v) : u \in M,\ v \in M'\}$$

□

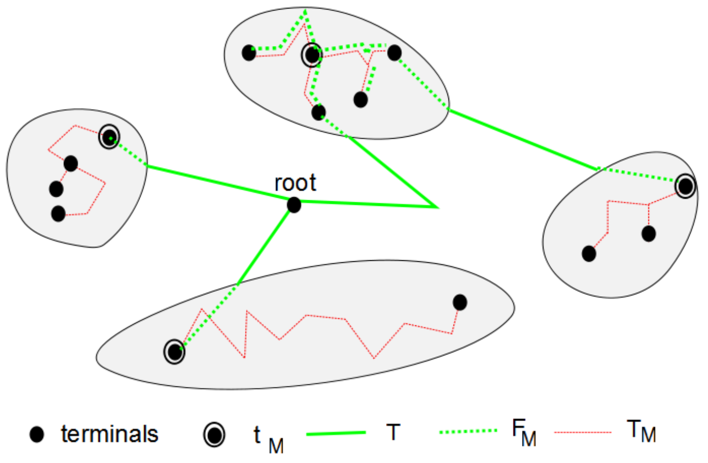

Figure 5.3: Illustrating the construction of the first-stage tree T_0

Note that, by definition, if M, M' are such that $M \cap M' \neq \phi$, then, $d(M, M') = 0$. We have enough ground work laid out to define the properties of the set of *representative* moats. Let $\beta \geq 1$ be a fixed real number whose value we will determine later. The following lemma is very similar to the Proposition 3.3 in [25].

Lemma 5.3.8. *There exists a set of moats $\mathcal{I}_R \subseteq \mathcal{I}$ (called the set of representative moats), computable in polynomial time such that:*

1. *If $M_1, M_2 \in \mathcal{I}_R$, then $d(M_1, M_2) \geq \frac{1}{\beta} \cdot (d(M_1) + d(M_2))$*
2. *For every moat $M \in \mathcal{I} \setminus \mathcal{I}_R$, we have a representative moat $rep(M) \in \mathcal{I}_R$, such that $d(M) \geq d(rep(M))$, and $d(M, rep(M)) \leq \frac{1}{\beta} \cdot (d(M) + d(rep(M)))$*

Proof. Proceed by examining all moats of $\mathcal{I}$ in increasing order of their diameters. If a moat M is being examined and is such that $d(M, M') \geq \frac{1}{\beta} \cdot (d(M) + d(M'))$ holds for all $M' \in \mathcal{I}_R$, then include M in $\mathcal{I}_R$. If not, then there exists a $M_0 \in \mathcal{I}_R$, such that $d(M_0, M) \leq \frac{1}{\beta} \cdot (d(M_0) + d(M))$. Define $rep(M) := M_0$ (i.e. M_0 is assigned as the representative moat of M in $\mathcal{I}_R$), and proceed to the next moat. □

For every moat $M \in \mathcal{I}$, we will label one terminal in M as t_M. This terminal t_M is the vertex at which the tree T_M of the moat will definitely meet the first-stage tree T_0, thus ensuring that all the terminals of a scenario k, present in the representative moats, are connected by the first and second-stage solutions together. The choice of t_M is descibed in step (3) below. Our aim is to build a low-cost Steiner tree over the collection of moats $\mathcal{I}_R$. The following steps lead to the construction of our first-stage tree T_0:

1. Contract each moat $M \in \mathcal{I}_R$ into a single vertex v_M.
2. Build a MST T on the set of terminals $\hat{R} = \{r\} \bigcup \{v_M : M \in \mathcal{I}_R\}$.
3. For every $M \in \mathcal{I}_R$, include a set of edges F_M that connects T_M to every T-edge incident on M, such that edges in F_M form a tree and contain a terminal from M. Label this terminal as t_M.
4. Our first-stage tree is $T_0 = T \cup \left(\bigcup_{M \in \mathcal{I}_R} F_M\right)$.

We need to describe how to choose the set of edges F_M in step (3) above. The choice of F_M depends on the degree of the vertex v_M corresponding to moat M in the tree T:

- *Case 1 (Degree of v_M in T is 1):* Let v be the vertex at which a T-edge is incident on moat M. Then t_M is the terminal in M closest to v, and F_M is the shortest path from v to t_M.
- *Case 2 (Degree of v_M in T is > 1):* Label any terminal in M as t_M. Connect each T-edge incident to M to the tree T_M using shortest paths. These shortest paths, together with the tree T_M, form the set F_M.

We will now bound the costs of T and F_M separately, giving us a bound on the cost of T_0.

Lemma 5.3.9. $c(T) \leq 2\alpha \sum_e x_e^0$

Proof. Let G' be the graph obtained by contracting the moats in $\mathcal{I}_R$, with $\hat{R}$ as the set of terminals (root and contracted moats). Let $\mathcal{S} = \{S : S \cap \hat{R} \neq \phi,\ r \notin S\}$. Consider the linear program for the Steiner tree problem:

$$\begin{aligned} \text{maximize:} \quad & \sum_{e \in E(G')} c(e) \cdot y_e \\ \text{subject to:} \quad & \forall S \in \mathcal{S}, \quad \sum_{e \in \delta(S)} y_e \geq 1, \\ & \forall e \in E(G'), \quad y_e \geq 0. \end{aligned}$$

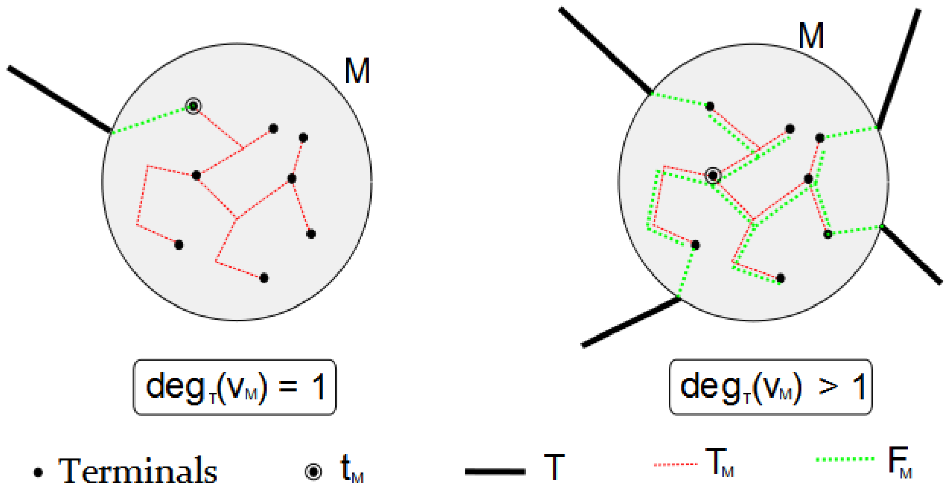

Figure 5.4: Illustrating choice of F_M and t_M for a moat M depending on the degree of the corresponding vertex v_M in the tree T

By Lemma 5.3.2, any set $S \in \mathcal{S}$ is an inactive set in the original graph, and by the same Lemma, $\exists t \in S$, such that

$$\sum_{e\in\delta(S)} x_e^0 \;\geq\; F_i(t,S) \;\geq\; \frac{1}{\alpha} \quad\Rightarrow\quad \sum_{e\in\delta(S)} \alpha\cdot x_e^0 \;\geq\; 1$$

And thus, $y_e = \alpha \cdot x_e^0$ is a feasible fractional solution to the above linear program.It is well known that the cost of an MST is at most twice the cost of the optimal fractional solution to the above LP. Thus, we have that,

$$c(T) \leq 2 \sum_{e\in E(G')} c(e)y_e \leq 2\alpha \sum_{e\in E(G')} c(e)x_e^0 \leq 2\alpha \sum_{e\in E(G)} c(e)x_e^0$$

□

Lemma 5.3.10. *Let $M \in \mathcal{I}$. If $u, v \in M$, then $d(u,v) \leq d(M)$*

Proof. Recall that $d(M) = 2\tau_M + c(T_M)$. Let t_1, t_2 be any two terminals in M. since all the terminals in M are connected by the Steiner tree T_M, thus, $d(t_1,t_2) \leq c(T_M)$. Also, from Lemma 5.3.5, for any vertex $v \in M$, there is a terminal $t_v \in M$, such that $d(v,t_v) \leq \tau_M$.

Thus, for every $u, v \in M$, there are some terminals $t_u, t_v \in M$, such that:

$$d(u,v) \;\leq\; d(u,t_u) + d(t_u,t_v) + d(t_v,v) \;\leq\; 2 \cdot \tau_M + c(T_M) = d(M)$$

□

We can bound the cost of the cost of edges in $\cup_{M \in \mathcal{I}_R} F_M$ in terms of the cost of the tree T.

Lemma 5.3.11. $c\left(\bigcup_{M \in \mathcal{I}_R} F_M\right) \;\leq\; \frac{\beta}{2} \cdot c(T)$

Proof. Let $M \in \mathcal{I}_R$. If $deg_T(v_M) = 1$, then, by the choice of edges in F_M and Lemma 5.3.5, we have that $c(F_M) \;\leq\; \tau_M \;\leq\; \frac{d(M)}{2}$. If $deg_T(v_M) > 1$, then $c(F_M) = c(T_M)+$ cost of shortest paths from the vertices where edges of T are incident at moat M to their nearest terminals. By Lemma 5.3.5, each of these shortest paths is at most τ_M in length. Thus,

$$c(F_M) \leq c(T_M) + deg_T(v_M) \cdot \tau_M \leq \frac{deg_T(v_M)}{2}\left(c(T_M) + 2 \cdot \tau_M\right) = deg_T(v_M) \cdot \frac{d(M)}{2}$$

Hence, we have,

$$c\left(\bigcup_{M \in \mathcal{I}_R} F_M\right) \;\leq\; \sum_{M \in \mathcal{I}_R} deg_T(v_M) \cdot \frac{d(M)}{2} \tag{5.12}$$

Let $e = (v_{M_1}, v_{M_2})$ be an edge in the tree T. We know that

$$c(e) \geq \frac{1}{\beta} \cdot (d(M_1) + d(M_2))$$

Summing over all edges, we get,

$$c(T) \geq \frac{1}{\beta} \cdot \sum_{M \in \mathcal{I}_R} deg_T(v_M) \cdot d(M) \tag{5.13}$$

From (5.12) and (5.13), we get,

$$c\left(\bigcup_{M \in \mathcal{I}_R} F_M\right) \;\leq\; \frac{\beta}{2} \cdot c(T)$$

□

Corollary 5.3.12. $c(T_0) \leq 2\alpha\left(1 + \frac{\beta}{2}\right) \cdot \sum_e c(e)x_e^0$

5.3.3 Phase 3: Extending Second-Stage Solution

For every moat $M \in \mathcal{I} \backslash \mathcal{I}_R$, let $v_M \in M$ and $v_{rep(M)} \in rep(M)$ be such that $d(v_M, v_{rep(M)}) = d(M, rep(M))$ (i.e. these are the vertices closest to each other in the two moats). Let t_M be the terminal in M closest to v_M. Let p_M be the following path:

$$p(M) \;=\; p(t_M, v_M) \;\cup\; p(v_M, v_{rep(M)}) \;\cup\; p(v_{rep(M)}, t_{rep(M)})$$

The second-stage network of moat M is $T_M \cup p_M$. We have the following bound on $c(p_M)$ by using the definition of $\mathcal{I}_R$ and Lemmas (5.3.5) and (5.3.10).

$$\begin{aligned} c(p_M) \;&\leq\; c(p(t, v_M)) + c(p(v_M, v_{rep(M)})) + c(p(v_{rep(M)}, t_{rep(M)})) \\ &\leq\; \tau_M + \frac{1}{\beta}(d(M) + d(rep(M))) + d(rep(M)) \\ &\leq\; \tau_M + \left(1 + \frac{2}{\beta}\right) d(M) \end{aligned}$$

For every scenario k, consider the moats in $\mathcal{G}_k$ in increasing order of their termination time τ_M. For a moat $M \in \mathcal{G}_k$, let $P(M) = \{M' \in \mathcal{G}_k \cup \mathcal{I}_k : \tau_{M'} \leq \tau_M\}$. Then, M was added to $\mathcal{G}_k$, because it had grown to *collide* with some set $M' \in P(M)$, i.e. an edge e incident on both M and M' had become tight. This set M' was labeled as the *parent* of M.

Lemma 5.3.13. *Let $M \in \mathcal{G}_k$, and $M' \in P(M)$ be such that it is labeled as the parent of M. Then, $\exists$ terminal $t \in M$ and $t' \in M'$, such that $d(t, t') \leq 2 \cdot \tau_M$.*

Proof. Let $e = (u, v)$, where $u \in M$, and $v \in M'$. Let τ_1 be the time when u gets included in M, and τ_2 be the time when v gets included in M'. Then, by Lemma 5.3.10, $\exists$ terminal $t \in M$, such that $d(t, u) \leq \tau_1$, and $\exists$ terminal $t' \in M'$, such that $d(t', v) \leq \tau_2$. Also, because edge e gets tight at time τ_M, thus, $(\tau_M - \tau_1) + (\tau_{M'} - \tau_2) = c(e)$. Thus,

$$d(t, t') \;\leq\; d(t, u) + c(e) + d(v, t') \;\leq\; \tau_1 + (\tau_M + \tau_{M'} - \tau_1 - \tau_2) + \tau_2 \;\leq\; 2 \cdot \tau_M$$

where the last inequality follows from the definition of $P(M)$. □

Thus, the cost of the second-stage network of a moat $M \in \mathcal{G}_k$ is at most $2 \cdot \tau_M + c(T_M) = d(M)$.

5.3.4 Putting Things Together

The cost of the second-stage network of a moat $M \in \mathcal{I} \setminus \mathcal{I}_R$ is:

$$\begin{aligned} c(T_M) + c(p_M) \;&\leq\; c(T_M) + \tau_M + \left(1 + \frac{2}{\beta}\right) d(M) \\ &\leq\; \left(2 + \frac{2}{\beta}\right) d(M) \end{aligned}$$

Cost of the second-stage network of a moat $M \in \mathcal{I}_R$ is:

$$c(T_M) \;\leq\; d(M) \;\leq\; \left(2 + \frac{2}{\beta}\right) d(M)$$

Cost of the second-stage network of a moat $M \in \mathcal{G}_k$ is:

$$2 \cdot \tau_M + c(T_M) \;=\; d(M) \;\leq\; \left(2 + \frac{2}{\beta}\right) d(M)$$

Also, from Lemmas (5.3.3) and (5.3.4) and the definition of $d(M)$, we have,

$$\sum_{M \in \mathcal{I}_k \cup \mathcal{G}_k} d(M) \;\leq\; 2\left(\frac{\alpha}{\alpha - 1}\right) \sum_{e} c(e) x_e^k$$

Thus, cost of the second-stage network for the scenario k is bounded by:

$$\begin{aligned} &\sum_{M \in \mathcal{I}_k \cup \mathcal{G}_k} 2\left(2 + \frac{2}{\beta}\right)\left(\frac{\alpha}{\alpha - 1}\right) \sum_{e \in M} c(e) x_e^k \\ &\leq\; 2\left(2 + \frac{2}{\beta}\right)\left(\frac{\alpha}{\alpha - 1}\right) \sum_{e \in E(G)} c(e) x_e^k \end{aligned}$$

Also, cost of the first-stage network, i.e. tree T_0 is bounded by Corollary 5.3.12 as follows:

$$c(T_0) \leq 2\alpha \left(1 + \frac{\beta}{2}\right) \cdot \sum_e c(e)x_e^0$$

Thus, expected cost of our solution is bounded by:

$$2\alpha \left(1 + \frac{\beta}{2}\right) \cdot \sum_e c(e)x_e^0 \; + \; 2\left(2 + \frac{2}{\beta}\right)\left(\frac{\alpha}{\alpha - 1}\right) \sum_{k=1}^{m} \sigma_k p_k \sum_{e \in E(G)} c(e)x_e^k$$

For $\alpha = 2.5$ and $\beta = 2$, the above expression reduces to:

$$10 \cdot \left(\sum_e c(e)x_e^0 + \sum_{k=1}^{m} \sigma_k p_k \sum_{e \in E(G)} c(e)x_e^k\right) \tag{5.14}$$

Thus, it is possible to round it to obtain an integral solution of cost at most 10 times the cost of the optimal fractional solution to SST-LP2. Consequently, the integral solution so obtained is within a factor of at most 20 of the optimal integral solution to SST-LP1 Thus, we have proved the following theorem:

Theorem 5.3.14. *Our algorithm is a* 20*-factor approximation algorithm for the Stochastic Steiner Tree Problem with finite number of scenarios.*

5.4 Stochastic Steiner Forest

As noted earlier, our motivation in studying this algorithm for the SST problem was to gain better insights into the (only) constant factor approximation algorithm for the SSF problem under the black box oracle model by Gupta and Kumar in [21]. In this section we discuss some problems that arise as a result of trying to extend the algorithm for the SST to an algorithm for SSF.

As noted by the authors in [21], a primal dual approach for the SST problem is very well suitable for the rooted version of the problem. The main idea of growing the moats first for the second stage solution, obtaining a collection of disjoint moats for each stage, and then choosing a collection of *good* (disjoint and far apart) moats from all the moats to proceed with another primal-dual algorithm to construct the first stage tree connecting the scenarios, works well for the rooted SST. This is beacuse, in rooted SST problem, the transition of a growing moat from the second stage to the first stage happens only once and is easy to work with. In the unrooted version of SST problem and even the SSF problem, one of the difficulties that arise is that such a transition is not a singular phenomenon. The algorithm for the rooted SST case rely heavily on the results in the Lemma 5.3.2, many of which fail to hold if the definitions are extended in an analogous way for the SSF problem.

In rooted SST, since all the terminals have to be eventually connected to the root, once the first stage solution (connecting core tree) is started to be build, the growing moats become inactive only when they collide either directly with the root or with another moat containing the root. However in the SSF problem, an active moat may cease to be active because a source sink pair is now satisfied, even though there may still be unsatisfied pairs in the growing moats but their contribution towards the boundary of the moat is not enough to pay for buying edges. Thus, the transition from SST to SSF is fairly involved, and requires new ideas.

References

[1] A. Aggarwal, L. Anand, M. Bansal, N. Garg, N. Gupta, S. Gupta, and S. Jain. A 3-approximation for facility location with uniform capacities. *Integer Programming and Combinatorial Optimization*, pages 149–162, 2010. 14

[2] Ajit Agrawal, Philip N. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized steiner problem on networks. *SIAM J. Comput.*, 24(3):440–456, 1995. 57, 58

[3] S. Ahmadian and C. Swamy. Improved approximation guarantees for lower-bounded facility location. *Arxiv preprint arXiv:1104.3128*, 2011. 14

[4] EML Beale. On minimizing a convex function subject to linear inequalities. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 173–184, 1955. 2

[5] A. Ben-Tal, T. Margalit, and A. Nemirovski. Robust modeling of multi-stage portfolio problems. *High performance optimization*, pages 303–328, 2000. 3

[6] A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs. *Operations Research Letters*, 25(1):1–14, 1999. 3

[7] A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 88(3):411–424, 2000. 3

[8] A. Ben-Tal and A. Nemirovski. On tractable approximations of uncertain linear matrix inequalities affected by interval uncertainty. *SIAM Journal on Optimization*, 12(3):811–833, 2002. 3

[9] D. Bertsimas, D.B. Brown, and C. Caramanis. Theory and applications of robust optimization. *Arxiv preprint arXiv:1010.5445*, 2010. 2, 3

[10] J. Byrka. An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 29–43, 2007. 9

[11] Jaroslaw Byrka, MohammadReza Ghodsi, and Aravind Srinivasan. Lp-rounding algorithms for facility-location problems. *CoRR*, abs/1007.3611, 2010. 23, 24, 25, 29, 42, 45

[12] M. Charikar, S. Guha, É. Tardos, and D.B. Shmoys. A constant-factor approximation algorithm for the k-median problem. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 1–10. ACM, 1999. 48

[13] S. Chechik, D. Peleg, and T. Schwentick. Robust fault tolerant uncapacitated facility location. In *27th International Symposium on Theoretical Aspects of Computer Science (STACS 2010)*, volume 5, pages 191–202. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010. 22, 23, 44

[14] F.A. Chudak and D.P. Williamson. Improved approximation algorithms for capacitated facility location problems. *Mathematical programming*, 102(2):207–222, 2005. 14

[15] G.B. Dantzig. Linear programming under uncertainty. *Stochastic Programming*, pages 1–11, 2011. 2

[16] L. El Ghaoui and H. Lebret. Robust solutions to least-squares problems with uncertain data. *SIAM Journal on Matrix Analysis and Applications*, 18:1035–1064, 1997. 3

[17] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998. 5

[18] L. Fleischer, J. K
"onemann, S. Leonardi, and G. Sch
"afer. Simple cost sharing schemes for multicommodity rent-or-buy and stochastic steiner tree. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 663–670. ACM, 2006. 51

[19] Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24(2):296–317, 1995. 57, 58

[20] S. Guha and S. Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of algorithms*, 31(1):228–248, 1999. 9

[21] A. Gupta and A. Kumar. A constant-factor approximation for stochastic steiner forest. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 659–668. ACM, 2009. 51, 68

[22] A. Gupta, V. Nagarajan, and R. Ravi. Thresholded covering algorithms for robust and max-min optimization. *Automata, Languages and Programming*, pages 262–274, 2010. 5

[23] A. Gupta and M. Pál. Stochastic steiner trees without a root. *Automata, Languages and Programming*, pages 1051–1063, 2005. 51

[24] A. Gupta, M. Pál, R. Ravi, and A. Sinha. Boosted sampling: approximation algorithms for stochastic optimization. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 417–426. ACM, 2004. 51

[25] Anupam Gupta, R. Ravi, and Amitabh Sinha. Lp rounding approximation algorithms for stochastic network design. *Math. Oper. Res.*, 32(2):345–364, 2007. 51, 52, 53, 55, 57, 58, 61, 62

[26] D.S. Hochbaum. Heuristics for the fixed cost median problem. *Mathematical programming*, 22(1):148–162, 1982. 8

[27] 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 (JACM)*, 48(2):274–296, 2001. 14

[28] J. Kamal and V. Vazirani. An approximation algorithm for the fault tolerant metric facility location problem. *Approximation Algorithms for Combinatorial Optimization*, pages 177–182, 2000. 22

[29] J.H. Lin and J.S. Vitter. Approximation algorithms for geometric median problems. *Information Processing Letters*, 44(5):245–249, 1992.

[30] J.H. Lin and J.S. Vitter. ε-approximations with minimum packing constraint violation (extended abstract). In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 771–782. ACM, 1992. 8, 10

[31] M. Mahdian, Y. Ye, and J. Zhang. A 2-approximation algorithm for the soft-capacitated facility location problem. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 149–162, 2003. 14

[32] S. Mudchanatongsuk, F. Ordónez, and J. Liu. Robust solutions for network design under transportation cost and demand uncertainty. *Journal of the Operational Research Society*, 59(5):652–662, 2008. 3

[33] M. Pál, T. Tardos, and T. Wexler. Facility location with nonuniform hard capacities. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 329–338. IEEE, 2001. 14

[34] M.R.K.C.G. Plaxton and R. Rajaraman. Analysis of a local search heuristic for facility location problems. *Algorithms*, pages 1–10, 1998. 14

[35] D.B. Shmoys and C. Swamy. Stochastic optimization is (almost) as easy as deterministic optimization. In *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*, pages 228–237. IEEE. 5

[36] D.B. Shmoys, É. Tardos, and K. Aardal. Approximation algorithms for facility location problems (extended abstract). In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 265–274. ACM, 1997. 8, 13, 15

[37] Z. Svitkina. Lower-bounded facility location. *ACM Transactions on Algorithms (TALG)*, 6(4):69, 2010. 14

[38] C. Swamy and D.B. Shmoys. Approximation algorithms for 2-stage stochastic optimization problems. *ACM SIGACT News*, 37(1):33–46, 2006. 1, 2

[39] J. Zhang, B. Chen, and Y. Ye. A multiexchange local search algorithm for the capacitated facility location problem. *Mathematics of Operations Research*, pages 389–403, 2005. 14