UNIVERSITY OF CALIFORNIA
RIVERSIDE

Approximation Algorithms for the Fault-Tolerant Facility Placement Problem

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Li Yan

June 2013

Dissertation Committee:

Professor Marek Chrobak, Chairperson
Professor Tao Jiang
Professor Stefano Lonardi
Professor Neal Young

The Dissertation of Li Yan is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

I would thank my advisor, Professor Marek Chrobak, for bringing me into the PhD program at the University of California, Riverside, and for his guidance and patience with my studies and research during the past five years. I am also grateful to the committee, Professor Tao Jiang, Professor Stefano Lonardi, and Professor Neal Young for helpful discussions and comments on my research and this dissertation.

The supportive environment of the Algorithms and Computational Biology Lab and the Computer Science Department has made my PhD study here a pleasant experience and I am grateful to Claire Yu-ting Huang, Wei Li, Jonathan Dautrich, Monik Khare, Arman Yosefia and other labmates in the algorithm lab for helpful discussions and stimulation of ideas.

My labmates Monik Khare and Jonathan Dautrich, despite their tight schedules, read a preliminary version of this dissertation with great patience and attention to detail. Both offered many helpful comments and suggested numerous corrections. Toby Gustafson from the Computer Science Department, and my fellow graduate students Shiwen Chen, Dave Gomboc and Rachid Ounit have read later revisions and they have helped improve the presentation significantly. Dorian Nogneng, a visiting student to our lab, read part of the dissertation and pointed out several flaws in the theorems and their proofs.

Kristin Noone in the Graduate Writing Center, and Stephanie Chang in the undergraduate program have proofread this dissertation and corrected a number of grammatical errors.

During this final stage of my PhD study, our family had a new member, little

Terry. Being with a baby as laid-back as Terry has kept me from getting overly stressed. I would also like to take this opportunity to show appreciation to my wife, Ying Liu, for handling most of the family responsibilities and bearing with me for staying up late working on the dissertation and the defense.

To my parents, who always have faith in my endeavors.

ABSTRACT OF THE DISSERTATION

Approximation Algorithms for the Fault-Tolerant Facility Placement Problem

by

Li Yan

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, June 2013
Professor Marek Chrobak, Chairperson

In this thesis, we have studied the Fault-Tolerant Facility Placement problem (FTFP). In the FTFP problem, we are given a set of sites at which we can open facilities, and a set of clients with demands. To satisfy demands, clients must be connected to open facilities. The goal is to satisfy all clients' demands while minimizing the combined cost of opening facilities and connecting clients to facilities. The problem is NP-hard, and hence we study approximation algorithms and their performance guarantees. Approximation algorithms are algorithms that run in polynomial time with provable performance bounds relative to optimal solutions.

We present two techniques with which we develop several LP-rounding algorithms with progressively improved approximation ratios. The best ratio we have is 1.575. We also study primal-dual approaches. In particular, we show that a natural greedy algorithm analyzed using the dual-fitting technique gives an approximation ratio of $O(\log n)$. On the negative side, under a natural assumption, we give an example showing that the dual-fitting analysis cannot give a ratio better than $O(\log n / \log\log n)$.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Related Problems

Facility location problems are a class of problems of both theoretical interest and practical significance. In general, facility location problems are about selecting a set of sites to open facilities and servicing clients by connecting them to facilities. A classical application is to set up warehouses to distribute commodities to retailers. On the one hand, having more warehouses helps reduce shipping cost, because every retailer can be close to some warehouse. On the other hand, setting up and maintaining a warehouse can be costly. In this scenario, there is a trade-off between having more warehouses and cheaper shipping and having fewer warehouses at the expense of shipping. Another application is to place content servers in a computer network. In the network, we have a number of client machines that need to access files from one of the servers. Having more servers allows faster access for all clients. However, keeping a large number of servers around requires substantial hardware and software investment, as well as committing operation engineers to keep the servers up and running. A more recent example concerns today's web giants like Google, Facebook and Amazon. These web-based companies have geographically distributed development offices. All these offices require access to large amounts of data from a handful of data centers, located in a few carefully chosen locations. On the one hand, building a data center is costly. The electricity bill alone would discourage the plan of having an excessive number of data centers. On the other hand, demands from all offices need to be addressed, or engineers would sit idling while waiting for data transfer to complete. We shall keep using the development office and data center example to illustrate several aspects of facility location problems.

First, setting up a data center incurs a cost and the cost varies with locations. The real estate rent in Kansas usually does not compare to even a fraction of that in San Francisco or New York. The electricity rate is also very different in mid-west areas from that in California. Other factors like potential natural disasters, such as tornadoes and earthquakes could also be factored into the cost estimate to build a data center in a certain location. In short, different locations have different costs to set up a data center.

Second, different offices may have different demands. A larger office may require access to several data centers, either for speedy data transfer, or for redundancy when one or more of the data centers appear to be offline.

Third, data centers may have capacities. It is thus reasonable to put a limit on the number of offices that a certain data center is able to serve. However, to keep the problem simple, we may or may not have this constraint in our problem definition.

Facility location problems have long been an active topic in both Operations Research and Computer Science. Early works on facility location problems can be found in [32, 38, 5]; see also the book by Mirchandani and Francis [41]. For a review of more recent work, see Shmoys' survey articles [44, 42], Vygen's lecture notes [47], and the book by Williamson and Shmoys [18]. Problems that have been considered include

- *the Uncapacitated Facility Location problem* [1], where each site can open at most one facility and each client needs to be connected to one facility,

- *the Fault-Tolerant Facility Location problem* [29, 22, 10], where each site can open at most one facility, and each client has a demand, which is the number of facilities to which it needs to connect,

- *the Capacitated Facility Location problem* [43, 37, 40, 51], where a site can open at most one facility and each client connects to one facility, but every facility has a capacity, which is the maximum number of clients it can accept,

- *the k-median problem* [12, 28, 4, 14], in which there is no cost to open a facility, each site can open at most one facility, and the total number of facilities opened cannot be more than k,

- *the Multi-level Facility Location problem* [2, 1, 50, 9, 31], in which we are given k disjoint sets of facilities and each client needs to be serviced by a sequence of k facilities with one from each set,

[1] This is a well studied problem and references are too numerous to be placed here, for a recent survey, see Vygen's lecture notes [47].

- *the Online Facility Location problem* [39, 20, 19], where demand points arrive one a time and must be assigned irrevocably upon arrival,

- *the Prize-Collecting Facility Location problem* [13], where a site can open at most one facility and each client can either connect to one facility, or stay unconnected, in which case a penalty is counted towards final cost.

In all the problems above, there is a cost to open a facility at a site, and there is a cost to connect a client to a facility. The goal is to minimize the total cost of opening facilities and connecting clients to facilities. In the Prize-Collecting Facility Location problem, we also pay a penalty for each client that is not connected.

The simplest variant, known as the Uncapacitated Facility Location problem (UFL), is also the one that has been studied most extensively.

Problem 1 (The UFL Problem) *In the* UFL *problem, we are given a set of sites* $\mathbb{F}$ *and a set of clients* $\mathbb{C}$. *Each site* i *can open one facility and each client* j *needs to connect to one facility. Facilities are uncapacitated, meaning that a facility can accept connections from any number of clients. We are also given the facility opening cost* f_i *for each site* i, *and the distance* d_{ij} *between a site* i *and a client* j. *The problem asks us to find a set of sites on which to open facilities and to connect clients to facilities, in such a way that the total cost is minimized. The total cost is the sum of the facility opening cost and the client connection cost.*

The UFL problem with general distances has an algorithm with an approximation ratio of $O(\log n)$, where n is the number of clients. This algorithm is credited to Hochbaum [25].

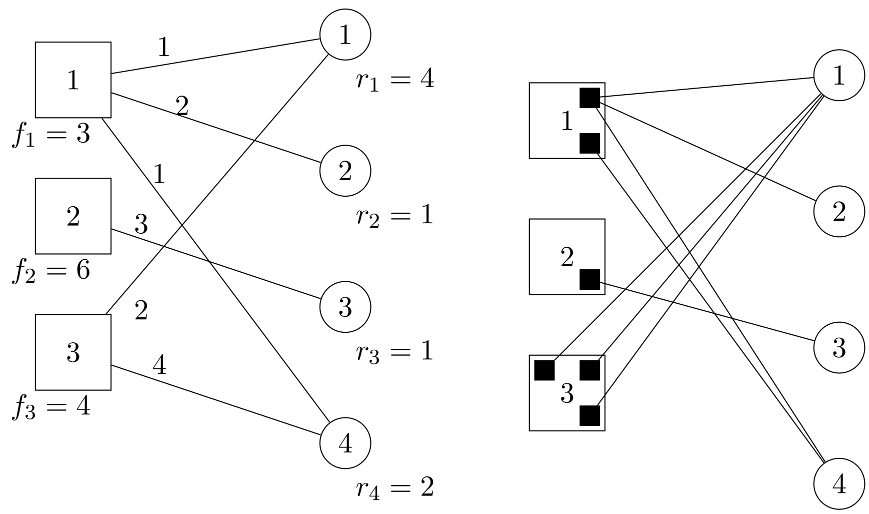
A matching lower bound of $\Omega(\log n)$ on the approximation ratio is immediate, as the UFL problem contains the well-known Set Cover problem as a special case [25]. When the distances form a metric, that is, they satisfy the triangle inequality, there are algorithms with constant approximation ratios. Shmoys, Tardos and Aardal [43] were the first ones to obtain an $O(1)$-approximation algorithm for the UFL problem. Currently, the best known approximation ratio is 1.488 by Li [33]. On the other hand, Guha and Khuller [21] showed a lower bound of 1.463 on the approximation ratio, under the assumption that $\mathsf{NP} \not\subseteq \mathsf{DTIME}(n^{O(\log\log n)})$. Sviridenko [47] weakened the underlying assumption to $\mathsf{P} \neq \mathsf{NP}$.

1.2 Fault-Tolerant Facility Placement

The problem studied in this thesis is called the Fault-Tolerant Facility Placement problem (FTFP), which generalizes the UFL problem. The differences between the UFL problem and the FTFP problem are: each client j now has a demand r_j that could be more than one, and at each site i we can open one or more facilities. More formally, we define the FTFP problem as:

Problem 2 (The FTFP Problem) *In the Fault-Tolerant Facility Placement problem, we are given a set of sites $\mathbb{F}$ and a set of clients $\mathbb{C}$. Each client $j \in \mathbb{C}$ has a demand r_j. Opening one facility at a site i incurs a cost of f_i. Making one connection from a client j to a facility at a site i incurs a cost of the distance d_{ij}. The problem asks for a solution in which every client j is connected to r_j different facilities and the total cost of opening facilities and connecting clients to facilities is minimized.*

One example is given in Figure 1.1, along with an integral feasible solution to that instance. The FTFP problem with general distances is well-understood, with a greedy algorithm achieving an approximation ratio of H_n where n is the number of clients and H_n is the n^{th} harmonic number (we show this result in Section 6.2), and a matching lower bound of H_n on the approximation ratio because FTFP contains the Set Cover problem as a special case. In this thesis we only consider the metric version of the FTFP problem, where the distances d_{ij} satisfy the triangle inequality; see Figure 1.2 for an illustration.



(a) The FTFP instance. (b) An integral feasible solution.

Figure 1.1: An instance of the FTFP problem with an integral feasible solution. The cost of the solution is $2f_1 + f_2 + 3f_3 + d_{11} + d_{12} + 2d_{14} + d_{23} + 3d_{31} = 38$.

The FTFP problem contains the UFL problem as a special case. In fact, by setting $r_j = 1$ for all clients j in FTFP, we get the UFL problem. Another problem that is closely related to the FTFP problem is the Fault-Tolerant Facility Location problem (FTFL) mentioned earlier. The difference between FTFP and FTFL is that, in FTFP we can open any number of facilities at a site, while in FTFL we can open at most one facility

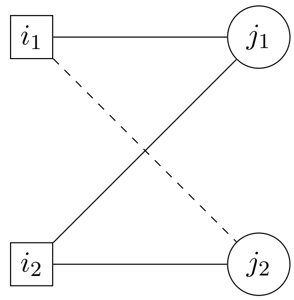


Figure 1.2: Illustration of the triangle inequality. For any two sites i_1 and i_2, and any two clients j_1 and j_2, we have $d_{i_1 j_2} \leq d_{i_1 j_1} + d_{i_2 j_1} + d_{i_2 j_2}$.

at a site.

1.3 Organization of Dissertation

The rest of this thesis is organized as follows: In Chapter 2 we present related work for UFL, FTFL and our problem FTFP; in Chapter 3 we give the LP for FTFP and describe structural properties of the optimal fractional solution that we use for our algorithms; in Chapter 4 we describe two main techniques, demand reduction and adaptive partitioning, that allow us to obtain a fractional solution with additional structural properties; in Chapter 5 we show how to round the obtained fractional solution to an integral solution with bounded cost; in Chapter 6 we present preliminary results of the primal-dual approach; in Chapter 7 we conclude this thesis with a discussion of open problems.

Chapter 2

Related Work

In this chapter we review the history of two problems closely related to our FTFP problem, the Uncapacitated Facility Location problem (UFL), and the Fault-Tolerant Facility Location problem (FTFL). We finish this chapter with an overview of known results and our work for the FTFP problem.

In all three problems, we are given a set of sites $\mathbb{F}$, a set of clients $\mathbb{C}$, the facility opening cost f_i for a site i, the distance d_{ij} between a site i and a client j, and the demand r_j for a client j. The differences between the problems are:

- UFL: all demands are 1; that is $r_j = 1$ for all clients j. Then we need not more than 1 facility at a site.
- FTFL: some demands may be more than 1, but each site can open at most 1 facility.
- FTFP: some demands may be more than 1, and each site can open any number of facilities.

For all three problems above, we assume the metric version. That is, the distances satisfy the triangle inequality: for any two sites i_1 and i_2 and two clients j_1 and j_2, we have

$$d_{i_1 j_2} \leq d_{i_1 j_1} + d_{i_2 j_1} + d_{i_2 j_2}.$$

In designing algorithms for all three problems, UFL, FTFL, and FTFP, we have two competing goals: on the one hand, we want to open as few facilities as possible so that our facility cost is small; on the other hand, we need as many facilities as possible so that every client can connect to nearby facilities. The main challenge, therefore, is to find a solution with a balanced facility cost and connection cost. We shall see how this balance is achieved in several known algorithms for UFL and FTFL. These two problems have been well studied in the past and a number of approximation algorithms for them are known. In particular, the LP-rounding algorithms for UFL inspired our approach for the FTFP problem. For this reason, we explain the LP-rounding algorithms for UFL in the coming section with full details, aiming at developing an intuition for our LP-rounding algorithms for the FTFP problem.

2.1 Related Work for UFL

The Uncapacitated Facility Location problem (UFL) is the simplest variant of the Facility Location problems, and has received the most attention. A surprising fact is that a wide range of different techniques for designing approximation algorithms have been found successful in achieving good ratios, as shown in Table 2.1. One alteration of terminology: to be consistent with the UFL literature, instead of saying "opening facilities at sites", we simply say “opening or closing facilities” without mentioning the sites, since in the UFL problem we have not more than one facility at each site.

author	technique	ratio	year
Shmoys, Tardos and Aardal	LP-rounding	3.16	1997 [43]
Chudak and Shmoys	LP-rounding	1.736	1998 [16]
Sviridenko	LP-rounding	1.582	2002 [45]
Jain and Vazirani	primal-dual	3	2001 [28]
Jain *et al.*	dual-fitting	1.61	2003 [26]
Arya *et al.*	local-search	3	2001 [3]
Byrka and Aardal	hybrid	1.50	2007 [6]
Li	hybrid	1.488 (best result)	2011 [33]
Guha and Khuller (*)	lower-bound	1.463	1998 [21]

Table 2.1: A history of approximation algorithms for the UFL problem and their ratios. The word *hybrid* refers to using two independent algorithms and returning the better solution of the two. The last row with an asterisk gives a lower bound on the approximation ratio.

2.1.1 Hardness Results

Before we delve into the approximation results for the UFL problem, we review the hardness results, which establish a limit on the best ratio with which we can approximate the UFL problem, under the well-respected assumption that $\mathsf{P} \neq \mathsf{NP}$. Since both the FTFL problem and the FTFP problem contain the UFL problem as a special case, all the hardness results mentioned here carry over to FTFL and FTFP as well. Regarding UFL, the following result is known:

Theorem 3 *[21] The metric UFL problem is* MaxSNP*-hard.*

The proof is by a reduction from the B-Vertex-Cover problem, which is a well-known MaxSNP-hard problem. The idea is to show that, for any constant ϵ, given a $(1+\epsilon)$-approximation algorithm for the UFL problem, we can construct a $(1+\epsilon')$-approximation algorithm for the B-Vertex-Cover problem with $\epsilon' = \epsilon(1+B)$. Given Theorem 3, we know that there exists some constant c such that no approximation algorithm with ratio less than c is possible unless $\mathsf{P} = \mathsf{NP}$. The best known such constant c for the UFL problem is $c = 1.463$, according to Guha and Khuller [21].

2.1.2 Linear Program Formulation

The analysis of approximation algorithms for NP-hard problems requires an estimate on the optimal solution's cost. For NP-hard problems, the task to compute the optimal solution's cost itself is also NP-hard. One alternative is to formulate an integer program for the problem and relax the integrality constraints to obtain a linear program (LP). Solving the LP gives an optimal fractional solution. The value of the fractional solution is then used to estimate the optimal integral solution's cost. In Appendix A.1 we give a brief introduction to Integer Programming, Linear Programming and their application for the UFL problem.

For the UFL problem, the LP formulated by Balinski [5] is now standard. We start with an integer program, in which we use a variable $y_i \in \{0,1\}$ to indicate whether a facility $i \in \mathbb{F}$ is open or not, and a variable $x_{ij} \in \{0,1\}$ to indicate whether a client j is connected to a facility i. Relaxing the integrality constraints, we obtain the following LP (2.1) for the UFL problem. Observe that we do not need an explicit constraint of $x_{ij} \leq 1$ or $y_i \leq 1$, as any optimal solution to the LP (2.1) must satisfy these two constraints automatically. The LP is:

$$\boxed{\begin{array}{lll} \text{minimize} & \sum_{i \in \mathbb{F}} f_i y_i + \sum_{i \in \mathbb{F}, j \in \mathbb{C}} d_{ij} x_{ij} & \\ \text{subject to} & y_i - x_{ij} \geq 0 & \forall i \in \mathbb{F}, j \in \mathbb{C} \\ & \sum_{i \in \mathbb{F}} x_{ij} \geq 1 & \forall j \in \mathbb{C} \\ & x_{ij} \geq 0, y_i \geq 0 & \forall i \in \mathbb{F}, j \in \mathbb{C} \end{array}} \tag{2.1}$$

The dual program is:

$$\boxed{\begin{array}{lll} \text{maximize} & \sum_{j \in \mathbb{C}} \alpha_j & \\ \text{subject to} & \sum_{j \in \mathbb{C}} \beta_{ij} \leq f_i & \forall i \in \mathbb{F} \\ & \alpha_j - \beta_{ij} \leq d_{ij} & \forall i \in \mathbb{F}, j \in \mathbb{C} \\ & \alpha_j \geq 0, \beta_{ij} \geq 0 & \forall i \in \mathbb{F}, j \in \mathbb{C} \end{array}} \tag{2.2}$$

Two general schemes using LP to design approximation algorithms are, the LP-rounding scheme and the primal-dual scheme. LP-rounding algorithms start with calling an LP-solver to obtain an optimal fractional solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$, and then round the fractional solution in such a way that feasibility is preserved, while the solution's cost does not increase by much. However, primal-dual algorithms do not require solving the LP, and the use of LP in the algorithms and their analysis is implicit. Those algorithms work by constructing an integral feasible primal solution and a feasible (fractional) dual solution simultaneously. Moreover, the cost of these two solutions are related. Assuming that the primal program is a minimization program, the Weak Duality Theorem, Theorem 26 in the appendix, tells us that we can use the cost of this dual solution as a lower bound on the optimal value for the primal program. We are then able to derive an approximation ratio by comparing the cost of the primal solution and the cost of the dual solution computed by the algorithm.

2.1.3 Approximation Algorithms

In this subsection, we give an overview of known approximation algorithms for the UFL problem, including LP-rounding algorithms, primal-dual algorithms, and local-search algorithms. All of them, except local-search algorithms, make use of the LPs (2.1) and (3.2) just described.

LP-rounding Algorithms

The first $O(1)$-approximation algorithm was obtained by Shmoys, Tardos and Aardal [43], using LP-rounding. The Shmoys *et al.*'s algorithm has also established a general framework that underpins all subsequent LP-rounding algorithms. In this framework, the clients are partitioned into clusters and each cluster has a representative client. The rounding algorithm guarantees that each representative has a nearby facility to connect to, and the rest of the clients then connect to those facilities via their representatives. This clustering structure is used in all known LP-rounding algorithms for the UFL problem.

The Shmoys *et al.*'s algorithm achieved a ratio of 3.16. Since their algorithm has made several greedy choices, it has left quite some room for improvements. Chudak and Shmoys [15] were the first ones to use the idea of randomized rounding for UFL. Roughly speaking, in their algorithm, each facility i is opened with probability y_i^*, which is given by the optimal fractional solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ to the LP (2.1). The expected connection cost is estimated using a provably worse random process, in which each facility is opened independently. The expected connection cost of this alternative process is easier to analyze. Chudak and Shmoys obtained a ratio of $1 + 2/e = 1.736$. Sviridenko [45] further improved the ratio to 1.582, by using a scaled version of the optimal fractional solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$, and by a judicious choice of some distribution of the scaling parameter. The rounding process is called *pipage rounding*, a deterministic rounding process that takes advantage of the concave property of some cost function. The analysis is highly technical.

Primal-dual Algorithms

Primal-dual algorithms do not require solving the LP and thus have a lower time complexity. For UFL, two primal-dual algorithms have achieved impressive approximation ratios: the first algorithm, given by Jain and Vazirani [28], achieved a ratio of 3; the second algorithm by Jain, Markakis, Mahdian, Saberi and Vazirani [26] achieved a ratio of 1.61.

For the Jain and Vazirani's algorithm with a ratio of 3, the approximation ratio is obtained via a relaxed version of the complementary slackness conditions. These conditions provide a bound on the cost of the primal solution in terms of the cost of the dual solution. For a maximization primal program, the cost of any feasible dual solution is a lower bound on the cost of the optimal primal solution. More on the complementary slackness conditions and their use for UFL can be found in Appendix A.1.2.

A slightly different primal-dual based algorithm was proposed by Jain, Markakis, Mahdian, Saberi and Vazirani [26]. They analyzed a greedy algorithm that repeatedly picks the most cost-effective star until all clients are connected. A star consists of one facility and a subset of clients. The cost of a star is the sum of the facility opening cost and connection cost of the clients to that facility. The cost-effectiveness of a star is the cost of the star divided by the number of clients in that star.

The algorithm iterates until all clients are connected. In each iteration, the algorithm picks the best star, connects all member clients to the facility, and sets the facility cost to zero. The clients that were in the star are removed from future considerations, but the facility could be reused for future stars.

Jain *et al.* analyzed the greedy algorithm and its variant using the dual-fitting technique. They first showed that the greedy algorithm can be interpreted as a process of growing a dual solution and updating a primal solution. Moreover, the cost of the primal solution is equal to the cost of the dual solution. It might appear that we have solved the UFL problem optimally, although we know that this cannot be the case, as the UFL problem is NP-hard. The catch is that the dual solution computed by the greedy algorithm is not feasible. The next step is to find a common factor γ, such that the dual solution after shrunk by γ (divided by γ), becomes feasible. That factor γ is then the desired approximation ratio. They showed that their two algorithms have approximation ratios of 1.861 [1] and 1.61 respectively.

Other Algorithms

A still different approach is local-search, in which we start with some feasible integral solution, make local moves to improve that solution, and stop when a local optimum

[1] An improved analysis by Mahdian showed the actual ratio is 1.81.

is achieved. The set of allowed local moves needs to be chosen carefully: admitting more powerful moves allows the local optimum to be closer to the global optimum, while restricting to a few simple moves results in faster algorithms and easier analysis. Korupolu, Plaxton and Rajaraman [30] were the first ones to study the local-search approaches for the UFL problem. They analyzed a heuristic and showed that the heuristic achieved a ratio of 5. Arya *et al.* [3] showed that their local-search algorithm achieved a ratio of 3. These algorithms do not make use of the LP and have been of special interest for the k-median problem and the Capacitated Facility location problem. For the k-median problem, the $(3 + \epsilon)$-approximation algorithm by Arya *et al.* [3] remains the algorithm with the best approximation ratio. For the Capacitated Facility Location problem, the natural LP has a large integrality gap and the only known $O(1)$-approximation algorithm by Pal *et al.* [40] uses local-search.

Best Result. To date, the best-known approximation algorithms for UFL are due to Byrka [6] with a ratio of 1.5, and a follow-up work by Li [33] with a ratio of 1.488. Both algorithms use a combination of two algorithms: one is an LP-rounding algorithm and the other is the 1.61 greedy algorithm by Jain, Mahdian and Saberi [27]. To explain the hybrid approach requires the notion of *bifactor analysis*; for this reason, we postpone the discussion of Byrka's and Li's work for now and introduce the bifactor analysis first.

2.1.4 Bifactor Analysis

Given that the cost of a UFL solution consists of two parts, the facility cost and the connection cost, a notion of bifactor approximation is appropriate. This notion was introduced by Jain *et al.* in [26]. An algorithm with a facility cost of F_{ALG} (sum of f_i for all open facilities i) and a connection cost of C_{ALG} (sum of d_{ij} for connected pairs of (i,j)), is said to be a (γ_f, γ_c)- approximation if, for every feasible solution SOL with a facility cost of F_{SOL} and a connection cost of C_{SOL}, we have

$$F_{\text{ALG}} + C_{\text{ALG}} \leq \gamma_f F_{\text{SOL}} + \gamma_c C_{\text{SOL}}.$$

In particular, the above holds if we substitute in an optimal fractional solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ for SOL. The solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ has facility cost $F^* = \sum_{i \in \mathbb{F}} f_i y_i^*$ and connection cost $C^* = \sum_{j \in \mathbb{C}} d_{ij} x_{ij}^*$. Therefore, for a (γ_f, γ_c)-approximation algorithm ALG, we have

$$F_{\text{ALG}} + C_{\text{ALG}} \leq \gamma_f F^* + \gamma_c C^*.$$

The notion of bifactor approximation is helpful when an algorithm has imbalanced factors of γ_f and γ_c. It is easy to see that such an algorithm has an approximation ratio of $\max\{\gamma_f, \gamma_c\}$. However, more can be said, as there are techniques like cost scaling and greedy augmentation to balance these two factors, thus achieving a better overall approximation ratio. The techniques of cost scaling and greedy augmentation, and their use for balancing the two factors were introduced by Guha, Khuller and Charikar [21, 11]. For example, the primal-dual algorithm by Jain and Vazirani [29] is a (1,3)-approximation algorithm; using cost scaling and greedy augmentation, it is possible to show that this algorithm can achieve a ratio of 1.85 [11].

Finally we return to the hybrid algorithms by Byrka and Aardal [7], and Li [33]. Byrka and Aardal gave an LP-rounding algorithm with bifactor $(1.68, 1.37)$, and showed that this algorithm, when combined with the $(1.11, 1.78)$-approximation algorithm by Jain *et al.* [26], gave a ratio of 1.50. Li showed that by choosing a nontrivial distribution of the scaling factor of Byrka's algorithm, the analysis can be refined to show an overall ratio of 1.488. The 1.488 ratio is currently the best known approximation result.

2.1.5 LP-rounding Algorithms

We now present a more detailed description of the LP-rounding algorithms, as our LP-rounding algorithms for FTFP are built on the LP-rounding algorithms for UFL.

The Motivation of Rounding

Every LP-rounding algorithm for UFL starts with solving the LP (2.1) to obtain an optimal fraction solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$. Then we need to round the fractional solution to an integral solution $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$.

An integral solution with a small cost would have each client connected to a nearby facility and few facilities open. Consider a client j. To get a handle on the connection cost, we would like the client j to connect to some neighboring facility $i \in N(j)$, where the neighborhood of j is defined as $N(j) \stackrel{\text{def}}{=} \{i \in \mathbb{F} : x^*_{ij} > 0\}$. For the sake of the connection cost, it is desirable that every client has a neighboring facility open, as those are facilities not too far away. However, this, in general, is not possible, or we would have to open too many facilities, and thus incur a high facility cost. An alternative is to select a subset of clients, denoted by $C' \subseteq \mathbb{C}$ and only require clients in C' have a neighboring facility open. Clients outside C' are then connected to a facility via some client in C'. The connection cost for clients in $\mathbb{C} \setminus C'$ are bounded using the triangle inequality. For this strategy to work, every client j outside C' needs to be able to find some client j' in C' such that both $d_{jj'} \stackrel{\text{def}}{=} \min_{i \in \mathbb{F}} d_{ij} + d_{ij'}$ and $d_{\phi(j')j'}$ are small. Here $\phi(j')$ is the facility that j' connects to.

The Clustering Structure

The clustering structure produced by the Shmoys, Tardos and Aardal's algorithm is depicted in Figure 2.1. These authors were inspired by the filtering and rounding work by Lin and Vitter [35]. The structure has three interesting properties:

- First, each cluster of clients has a representative [2].
- Second, the neighborhoods of the representatives are disjoint.
- Third, each client shares a neighbor with its representative.

A Simple 4-approximation

To see how this clustering structure helps rounding, we use a simple 4-approximation algorithm by Chudak [16] as an example. In this algorithm, clusters are formed by repeatedly picking a non-clustered client with the minimum α_j^* value as a new representative, where $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$ is the optimal dual solution; clients sharing a neighboring facility with the new representative then become members of that cluster. Once all clients have been clustered, the algorithm opens the cheapest facility in each representative's neighborhood. All clients in the same cluster connect to the only facility open in their representative's neighborhood.

[2] In the literature, the representative is called *center*.

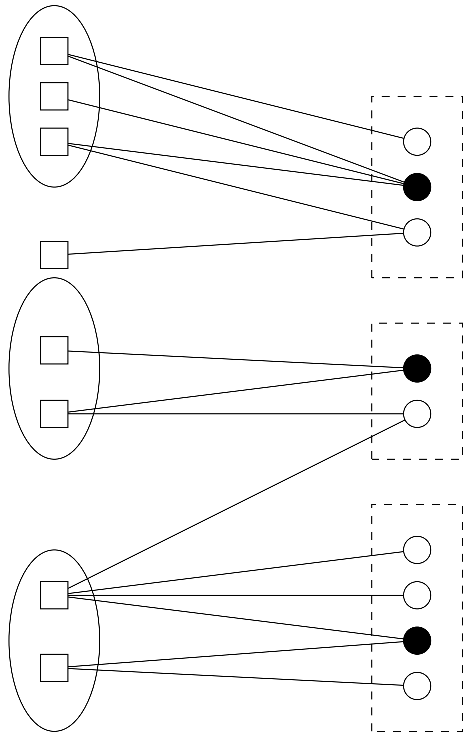


Figure 2.1: An illustration of the clustering structure of LP-rounding algorithms for UFL. Rectangles are facilities and circles are clients. Dashed boxes indicate clusters. The solid circle in each box denotes the representative of that cluster. An edge is drawn from a client to a neighboring facility. An ellipse indicates the neighborhood of a representative.

The facility cost of this solution is not more than $F^* = \sum_{i \in \mathbb{F}} f_i y_i^*$, because every facility that is opened can have its cost bounded by the average facility cost of the neighborhood. The connection cost of a representative j' is not more than $\alpha_{j'}^*$, because the complementary slackness conditions imply that $d_{ij'} \leq \alpha_{j'}^*$ for every facility i in $N(j')$. The connection cost for a non-representative client j can be bounded by the triangle inequality, that is $d_{i'j} \leq d_{i'j'} + d_{ij'} + d_{ij}$, where j' is the representative of j, and i' is the facility opened in j''s neighborhood, and i is a common neighbor of both j and j'. Using the complementary slackness conditions, the distance of $d_{i'j}$ is not more than $\alpha_{j'}^* + \alpha_{j'}^* + \alpha_j^*$, which is not more than $3\alpha_j^*$ because the representative j' has the minimum $\alpha_{j'}^*$ value among all clients in its cluster. Summarizing, the solution has a facility cost at most F^* and a connection cost not more than $3\sum_{j\in\mathbb{C}} \alpha_j^* = 3\,\mathrm{LP}^*$, where $\mathrm{LP}^* = F^* + C^*$. Since $\mathrm{LP}^* \leq \mathrm{OPT}$, where OPT is the optimal integral solution's cost, the algorithm is a 4-approximation.

2.2 Related Work for FTFL

The Fault-Tolerant Facility Location problem (FTFL) was introduced by Jain and Vazirani [29]. The LP for FTFL is

$$\boxed{\begin{array}{lll} \text{minimize} & \displaystyle\sum_{i\in\mathbb{F}} f_i y_i + \sum_{i\in\mathbb{F}, j\in\mathbb{C}} d_{ij} x_{ij} & \\ \text{subject to} & y_i - x_{ij} \geq 0 & \forall i \in \mathbb{F}, j \in \mathbb{C} \\ & \displaystyle\sum_{i\in\mathbb{F}} x_{ij} \geq r_j & \forall j \in \mathbb{C} \\ & y_i \leq 1 & \forall i \in \mathbb{F} \\ & x_{ij} \geq 0, y_i \geq 0 & \forall i \in \mathbb{F}, j \in \mathbb{C} \end{array}} \tag{2.3}$$

The constraint $\sum_{i\in\mathbb{F}} x_{ij} \geq r_j$ is present because, unlike in UFL, a client j now needs to be connected to r_j different facilities. The constraint $y_i \leq 1$ is necessary because in the FTFL problem a site can open at most 1 facility.

Jain and Vazirani adapted their primal-dual algorithm for UFL to FTFL and obtained a ratio of $3 \ln \max_j r_j$. The first $O(1)$-approximation algorithm was given by Guha, Meyerson and Munagala [22], using an LP-rounding algorithm similar to the Shmoys, Tardos and Aardal's [43] algorithm for the UFL problem. Swamy and Shmoys [46] gave an improved algorithm with a ratio of 2.076. Their algorithm uses the pipage rounding technique. The current best known approximation ratio is 1.7245, according to Byrka, Srinivasan and Swamy [10]. Their algorithm uses dependent rounding and a laminar clustering structure.

We note that all of the known $O(1)$-approximation algorithms for FTFL are LP-rounding algorithms and they require the fractional optimal solution of the LP as the first step. This step can be computationally expensive. Given the success of primal-dual based approaches for UFL, it is natural to ask whether such algorithms could be adapted to FTFL with good ratios. To the best of the author's knowledge, there has been no success in obtaining a primal-dual algorithm for FTFL with a sub-logarithmic ratio. This is in stark contrast to the fact that two different primal-dual algorithms [29, 26] have achieved constant ratios for UFL.

2.3 Related Work for FTFP

Our problem, the Fault-Tolerant Facility Placement problem (FTFP), was introduced by Xu and Shen [48] [3]. The study of FTFP was partly motivated to obtain a better understanding of the implication of the fault-tolerant requirement on facility location problems. Xu and Shen's results [48], and a follow-up work by Liao and Shen [34], seem to be valid only for a special case of FTFP.

Our work on the FTFP problem begins with a 4-approximation LP-rounding algorithm [49]. In this thesis we present significantly improved results for FTFP. Using LP-rounding algorithms, we achieved a ratio of 1.575, which matches the best known LP-based ratio for UFL [8]. The LP-rounding results are explained in Chapter 5. For primal-dual algorithms, we present our preliminary results in Chapter 6, in which we give an explanation of the difficulties in obtaining an $O(1)$-approximation using primal-dual based approaches.

[3] In their paper they call the problem the Fault-Tolerant Facility Allocation problem, or FTFA.

Chapter 3

Linear Program

In this chapter we give the linear program (LP) for the FTFP problem. The structure of the linear program implies a simple reduction for a special case of FTFP. We describe this reduction in Section 3.3. For the optimal fractional solution used for our approximation algorithms, we assume a structural property called *completeness*. The exact definition and a procedure to obtain such a solution is given in Section 3.4. The discussion in this chapter prepares the reader for Chapter 4, where we introduce our main techniques: *demand reduction* and *adaptive partitioning*.

3.1 Notation and Definition

For the reader's convenience, we repeat the problem definition and the notation before introducing the LP. In the Fault-Tolerant Facility Placement problem (FTFP), we denote the set of sites as $\mathbb{F}$, and the set of clients as $\mathbb{C}$. Each client $j \in \mathbb{C}$ has a demand r_j, meaning that j needs to be connected to r_j different facilities. The distance between a site i and a client j is denoted as d_{ij}. Opening one facility at a site i incurs a cost of f_i. Making one connection from a client j to a facility at a site i incurs a cost of d_{ij}. Thus, an FTFP instance is fully specified by $(\mathbb{F}, \mathbb{C}, \{r_j\}, \{f_i\}, \{d_{ij}\})$. We consider the metric version, in which the distances satisfy the triangle inequality: for any two sites i_1, i_2 and any two clients j_1, j_2, we have

$$d_{i_1 j_2} \leq d_{i_1 j_1} + d_{i_2 j_1} + d_{i_2 j_2}.$$

A solution to the FTFP problem is a vector $(\boldsymbol{x}, \boldsymbol{y})$, where $x_{ij} \in \{0, 1, 2, \ldots\}$ denotes the number of connections between a site i and a client j, and $y_i \in \{0, 1, 2 \ldots\}$ denotes the number of facilities opened at site i. The solution is *feasible* if $y_i \geq x_{ij}$ for every $i \in \mathbb{F}, j \in \mathbb{C}$ and $\sum_{i \in \mathbb{F}} x_{ij} = r_j$ for all clients $j \in \mathbb{C}$. The total cost of the solution is $\sum_{i \in \mathbb{F}} f_i y_i + \sum_{i \in \mathbb{F}, j \in \mathbb{C}} d_{ij} x_{ij}$. We call the first term $\sum_{i \in \mathbb{F}} f_i y_i$ the *facility cost* of the solution, and the second term $\sum_{i \in \mathbb{F}, j \in \mathbb{C}} d_{ij} x_{ij}$ the *connection cost* of the solution. Our goal is to find a feasible solution with the minimum total cost.

3.2 Linear Program for FTFP

The FTFP problem has a natural Integer Programming (IP) formulation. Let y_i represent the number of facilities opened at a site i, and let x_{ij} represent the number of connections from a client j to facilities at a site i. If we relax the integrality constraints, we obtain the following LP:

$$\boxed{\begin{array}{lll} \text{minimize} & cost(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i \in \mathbb{F}} f_i y_i + \sum_{i \in \mathbb{F}, j \in \mathbb{C}} d_{ij} x_{ij} & \\ \text{subject to} & y_i - x_{ij} \geq 0 & \forall i \in \mathbb{F}, j \in \mathbb{C} \\ & \sum_{i \in \mathbb{F}} x_{ij} \geq r_j & \forall j \in \mathbb{C} \\ & x_{ij} \geq 0, y_i \geq 0 & \forall i \in \mathbb{F}, j \in \mathbb{C} \end{array}} \tag{3.1}$$

The dual program is:

$$
\boxed{\begin{array}{lll}
\text{maximize} & \sum_{j\in\mathbb{C}} r_j\alpha_j & \\
\text{subject to} & \sum_{j\in\mathbb{C}}\beta_{ij} \le f_i & \forall i\in\mathbb{F} \\
& \alpha_j - \beta_{ij} \le d_{ij} & \forall i\in\mathbb{F}, j\in\mathbb{C} \\
& \alpha_j \ge 0, \beta_{ij} \ge 0 & \forall i\in\mathbb{F}, j\in\mathbb{C}
\end{array}} \tag{3.2}
$$

In each of our algorithms, we will fix some optimal solutions of the LPs (3.1) and (3.2) that we will denote by $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ and $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$, respectively.

With $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ fixed, we can define the optimal facility cost as $F^* = \sum_{i\in\mathbb{F}} f_i y_i^*$ and the optimal connection cost as $C^* = \sum_{i\in\mathbb{F}, j\in\mathbb{C}} d_{ij}x_{ij}^*$. Then $\text{LP}^* = cost(\boldsymbol{x}^*, \boldsymbol{y}^*) = F^* + C^*$ is the joint optimal value of (3.1) and (3.2). We can also associate with each client j its fractional connection cost $C_j^* = \sum_{i\in\mathbb{F}} d_{ij}x_{ij}^*$. Clearly, $C^* = \sum_{j\in\mathbb{C}} C_j^*$. Throughout this thesis, we will use notation OPT for the optimal integral solution of (3.1). OPT is the value we wish to approximate, but, since $\text{OPT} \ge \text{LP}^*$, we can instead use LP^* to estimate the approximation ratio of our algorithms.

3.3 Special Case with Uniform Demands

If we compare the LP (3.1) and its dual (3.2) for FTFP to the LP (2.1) and its dual (2.2) for UFL, these two LPs are very similar. In fact, the dual constraints of the two problems are identical. This makes one wonder whether there is a simple reduction from FTFP to UFL, so that we can take advantage of almost all known approximation algorithms for UFL and use them to solve FTFP. Unfortunately, we are not aware of such a reduction for FTFP with general demands. For the special case when all demands are equal, we observe that any fractional solution $(\boldsymbol{x}, \boldsymbol{y})$ to the LP (3.1), when scaled down by

$R = r_j$ (since all r_j's are equal), is a feasible solution to an UFL instance with the same set of sites, the same set of clients, and the same facility opening costs and the same distances. If we solve that UFL instance with an approximation algorithm, and duplicate the solution R times, we obtain an integral solution to the FTFP instance. It is not hard to see that the approximation ratio is preserved. This shows a simple reduction from FTFP to UFL for the uniform demands case.

3.4 Completeness and Facility Splitting

In this section, we describe a structural property. In our LP-rounding algorithms for FTFP, we use a fixed fractional optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$, and we assume that solution possess this structural property. This property is called *completeness*. Moreover, we give a procedure to obtain such a complete solution.

Define $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ to be *complete* if $x^*_{ij} > 0$ implies that $x^*_{ij} = y^*_i$ for all i, j. In other words, each connection either uses a site fully or not at all. As shown by Chudak and Shmoys [15], we can modify the given instance by adding at most $|\mathbb{C}|$ sites to obtain an equivalent instance that has a complete optimal solution, where "equivalent" means that the values of F^*, C^* and LP^*, as well as OPT, are not affected. Roughly, the argument is this: we notice that, without loss of generality, for each client k there exists at most one site i such that $0 < x^*_{ik} < y^*_i$. We can then perform the following *facility splitting* operation on site i: introduce a new site i', let $y^*_{i'} = y^*_i - x^*_{ik}$, redefine y^*_i to be x^*_{ik}, and then for each client j redistribute x^*_{ij} so that i retains as much connection value as possible and i' receives the rest. Specifically, we set

$$
\begin{aligned}
&y^*_{i'} \leftarrow y^*_i - x^*_{ik},\ y^*_i \leftarrow x^*_{ik}, \quad \text{and} \\
&x^*_{i'j} \leftarrow \max(x^*_{ij} - x^*_{ik}, 0),\ x^*_{ij} \leftarrow \min(x^*_{ij}, x^*_{ik}) \quad \text{for all } j \neq k.
\end{aligned}
$$

This operation eliminates the partial connection between k and i and does not create any new partial connections. Each client can split at most one site and hence we shall have at most $|\mathbb{C}|$ more sites.

Given the above discussion, without loss of generality, we can assume that the optimal fractional solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is complete. In fact, this assumption will greatly simplify some of the arguments in the paper. Additionally, we will frequently use the facility splitting operation described above in our algorithms to obtain fractional solutions with desirable properties.

Chapter 4

Techniques

In this chapter we introduce two techniques: *demand reduction* and *adaptive partitioning*. These techniques together produce a structured fractional solution to the LP (3.1). This structured solution possesses a number of nice properties, of which we then take advantage in our LP-rounding algorithms (Chapter 5) to obtain integral solutions with good approximation ratios. A diagram of our general approach employing the two techniques to obtain an approximation algorithm is depicted in Figure 4.1. In this process, we start with an optimal fraction solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ and split into two parts; we then solve the fractional part with an integral solution; finally, we combine this solution and the integral part obtained earlier to get a solution to the original instance.

Our first technique, demand reduction, allows us to confine our attention to a restricted version of the FTFP problem in which all demands r_j's are not too large. This restriction then sets the stage for the application of our second technique, adaptive partitioning, with which we obtain an FTFP instance with facilities created at sites (not opened

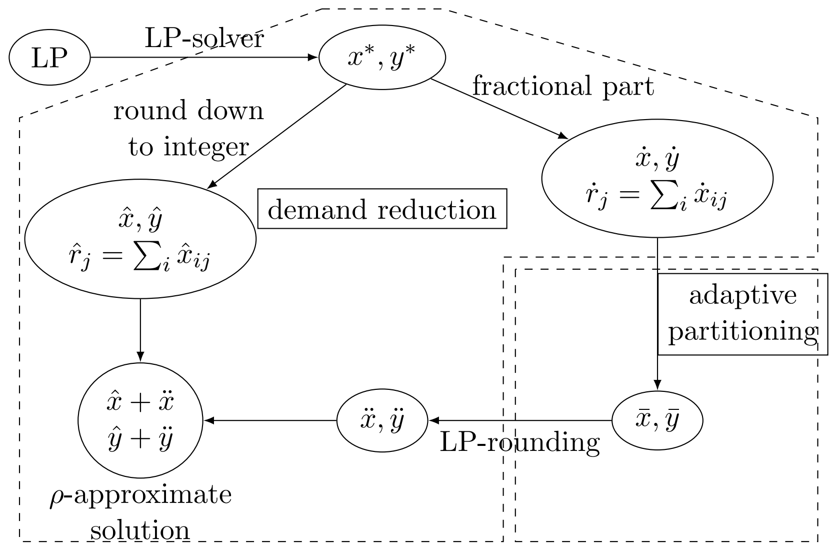


Figure 4.1: A diagram of the general approach using demand reduction and adaptive partitioning to design LP-rounding algorithms for the FTFP problem.

yet) and unit demand points derived from clients. Then, our job is to decide which facilities to open and how to connect unit demands to open facilities. We would like to point out that we still need to observe the fault-tolerant requirement, that is unit demands originated from the same client must connect to different facilities. We shall see that our adaptive partitioning step deals with the fault-tolerant requirement smoothly.

4.1 Demand Reduction

This section presents the demand reduction technique that reduces the FTFP problem for arbitrary demands to a special case where demands are bounded by $|\mathbb{F}|$, the number of sites. (The formal statement is a little more technical – see Theorem 5.) The demand reduction step prepares us for our second technique, adaptive partitioning (Section 4.4), as well as our LP-rounding algorithms (Chapter 5), since those steps process individual demands of each client one by one, and thus they critically rely on the demands being bounded polynomially in terms of $|\mathbb{F}|$ and $|\mathbb{C}|$ to keep the overall running time polynomial.

The reduction is based on an optimal fractional solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ to the LP (3.1). From the optimality of this solution, we can also assume that $\sum_{i\in\mathbb{F}} x^*_{ij} = r_j$ for all $j \in \mathbb{C}$. As explained in Chapter 3, we can assume that $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is complete, that is $x^*_{ij} > 0$ implies $x^*_{ij} = y^*_i$ for all i, j. We split this solution into two parts, namely $(\boldsymbol{x}^*, \boldsymbol{y}^*) = (\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) + (\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}})$, where

$$\begin{aligned} &\hat{y}_i \leftarrow \lfloor y^*_i \rfloor, \quad \hat{x}_{ij} \leftarrow \lfloor x^*_{ij} \rfloor \quad \text{and} \\ &\dot{y}_i \leftarrow y^*_i - \lfloor y^*_i \rfloor, \quad \dot{x}_{ij} \leftarrow x^*_{ij} - \lfloor x^*_{ij} \rfloor \end{aligned}$$

for all i, j. Now we construct two FTFP instances $\hat{\mathcal{I}}$ and $\dot{\mathcal{I}}$ with the same parameters as the original instance, except that the demand of each client j is $\hat{r}_j = \sum_{i\in\mathbb{F}} \hat{x}_{ij}$ in instance $\hat{\mathcal{I}}$ and $\dot{r}_j = \sum_{i\in\mathbb{F}} \dot{x}_{ij} = r_j - \hat{r}_j$ in instance $\dot{\mathcal{I}}$. It is obvious that if we have integral solutions to both $\hat{\mathcal{I}}$ and $\dot{\mathcal{I}}$ then, when added together, they form an integral solution to the original instance. Moreover, we have the following lemma.

Lemma 4 (i) $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ *is a feasible integral solution to instance* $\hat{\mathcal{I}}$.

(ii) $(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}})$ *is a feasible fractional solution to instance* $\dot{\mathcal{I}}$.

(iii) $\dot{r}_j \le |\mathbb{F}|$ *for every client* j.

Proof. (i) For feasibility, we need to verify that the constraints of the LP (3.1) are satisfied. Directly from the definition, we have $\hat{r}_j = \sum_{i\in\mathbb{F}} \hat{x}_{ij}$. For any i and j, by the feasibility of $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ we have $\hat{x}_{ij} = \lfloor x^*_{ij} \rfloor \le \lfloor y^*_i \rfloor = \hat{y}_i$.

(ii) From the definition, we have $\dot{r}_j = \sum_{i\in\mathbb{F}} \dot{x}_{ij}$. It remains to show that $\dot{y}_i \ge \dot{x}_{ij}$

for all i, j. If $x^*_{ij} = 0$, then $\dot{x}_{ij} = 0$ and we are done. Otherwise, by completeness, we have $x^*_{ij} = y^*_i$. Then $\dot{y}_i = y^*_i - \lfloor y^*_i \rfloor = x^*_{ij} - \lfloor x^*_{ij} \rfloor = \dot{x}_{ij}$.

(iii) From the definition of $\dot{x}_{ij}$ we have $\dot{x}_{ij} < 1$. Then the bound follows from the definition of $\dot{r}_j$. ■

Notice that our construction relies on the completeness assumption; in fact, it is easy to give an example where $(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}})$ would not be feasible if we used a non-complete optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$. Note also that the solutions $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ and $(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}})$ are in fact optimal for their corresponding instances, for if a better solution to $\hat{\mathcal{I}}$ or $\dot{\mathcal{I}}$ existed, it could give us a solution to $\mathcal{I}$ with a smaller objective value. We would also like to comment that completeness, although simplifying our argument here and afterwards, is not essential. For our demand reduction, we can deal with non-complete fractional solutions by taking $\hat{y}_i = (y^*_i - 1)_+$ [1] and $\hat{x}_{ij} = \min\{\lfloor x^*_{ij} \rfloor, \hat{y}_i\}$, and $\dot{y}_i = y^*_i - \hat{y}_i$, $\dot{x}_{ij} = x^*_{ij} - \hat{x}_{ij}$. For this set of fractional values, item (i) and (ii) of Lemma 4 remain valid, and (iii) now reads: $\dot{r}_j < 2|\mathbb{F}|$ for every client j.

Theorem 5 *Suppose that there is a polynomial-time algorithm $\mathcal{A}$ that, for any instance of* FTFP *with maximum demand bounded by $|\mathbb{F}|$, computes an integral solution that approximates the fractional optimum of this instance within factor $\rho \geq 1$. Then there is a ρ-approximation algorithm $\mathcal{A}'$ for* FTFP.

Proof. Given an FTFP instance with arbitrary demands, Algorithm $\mathcal{A}'$ works as follows: it solves the LP (3.1) to obtain a fractional optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$, which we may assume to be complete, then it constructs instances $\hat{\mathcal{I}}$ and $\dot{\mathcal{I}}$ described above, applies algorithm $\mathcal{A}$ to $\dot{\mathcal{I}}$, and finally combines (by adding the values) the integral solution $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ of $\hat{\mathcal{I}}$ and the

[1] The notation $(\cdot)_+$ means taking maximum of the term or 0.

integral solution of $\dot{\mathcal{I}}$ produced by $\mathcal{A}$. This clearly produces a feasible integral solution for the original instance $\mathcal{I}$. The solution produced by $\mathcal{A}$ has a cost at most $\rho \cdot \mathit{cost}(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}})$, because $(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}})$ is feasible for $\dot{\mathcal{I}}$. Thus the cost of $\mathcal{A}'$ is at most

$$\mathit{cost}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) + \rho \cdot \mathit{cost}(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}}) \leq \rho(\mathit{cost}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) + \mathit{cost}(\dot{\boldsymbol{x}}, \dot{\boldsymbol{y}})) = \rho \cdot \mathrm{LP}^* \leq \rho \cdot \mathrm{OPT},$$

where the first inequality follows from $\rho \geq 1$. This completes the proof. ■

The demand reduction step has two nice consequences which we describe in the next two sections. In Section 4.2 we give a reduction from FTFP to FTFL. In Section 4.3, we give a precise statement that confirms an intuitively appealing argument: when all demands r_j are large, the fractional optimal solution is very close to an integral solution. In other words, we can round the fractional solution to an integral solution with almost the same cost.

4.2 Reduction from FTFP to FTFL

Given the demand reduction technique, we may assume that we are working with a restricted version of FTFP, in which every demand r_j is not more than the number of sites $|\mathbb{F}|$. In this case, we can reduce FTFP to FTFL. The restriction on the demands r_j is essential to keep the size of the FTFL instance polynomially bounded by $|\mathbb{F}| \cdot |\mathbb{C}|$, where $|\mathbb{F}|$ is the number of sites and $|\mathbb{C}|$ is the number of clients.

For the reduction, we simply create $|\mathbb{F}|$ facilities (not opened yet) at each site. Thus, we have an FTFL instance where every client j has a demand r_j, and a set of facilities that could be opened. We are then able to use any known LP-rounding algorithm for FTFL to solve this FTFL instance. The solution to this FTFL instance can be converted trivially to a solution to that FTFP instance. Moreover, the approximation ratio for FTFL is preserved for FTFP. Given that FTFL has an LP-rounding 1.7245-approximation algorithm by Byrka, Srinivasan and Swamy [10], it is immediate that FTFP has an approximation algorithm with the same ratio. However, as we will show in Chapter 5, FTFP can be approximated with a better ratio of 1.575. Thus, from the standpoint of approximation, FTFP is more amenable than FTFL.

4.3 Asymptotic Approximation Ratio for Large Demands

When all demands are large, we expect that the fractional optimal solution to LP (3.1) is very close to an integral solution. Thus, it is reasonable to expect that, in this case, the optimal fractional solution can be rounded to an integral solution with almost the same cost. We turn this intuition into a concrete statement, which is Theorem 6.

Theorem 6 *Given an* FTFP *instance* $(\mathbb{F}, \mathbb{C}, \{r_j\}, \{f_i\}, \{d_{ij}\})$*, let* $m = |\mathbb{F}|$ *be the number of sites, and let* $Q = \min_{j \in \mathbb{C}} r_j$ *be the minimum demand; then there is an approximation algorithm with a ratio of* $(1 + O(m/Q))$*. In particular, when* Q *is large compared to* m*, this ratio approaches* 1*.*

Proof. First, we solve the LP (3.1) to obtain an optimal fractional solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$. Then, we apply demand reduction to obtain two instances: $\hat{\mathcal{I}}$ and $\dot{\mathcal{I}}$. For the $\hat{\mathcal{I}}$ instance, we already have an optimal integral solution, namely $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$. We now deal with the $\dot{\mathcal{I}}$ instance.

From Lemma 4 we know that for every client j, the value of $\dot{r}_j$ is no more than $m = |\mathbb{F}|$. Thus, we can solve the $\dot{\mathcal{I}}$ instance by creating m copies of UFL instances with the same parameters $(\mathbb{F}, \mathbb{C}, \{f_i\}, \{d_{ij}\})$; see Figure 4.2 for an illustration.

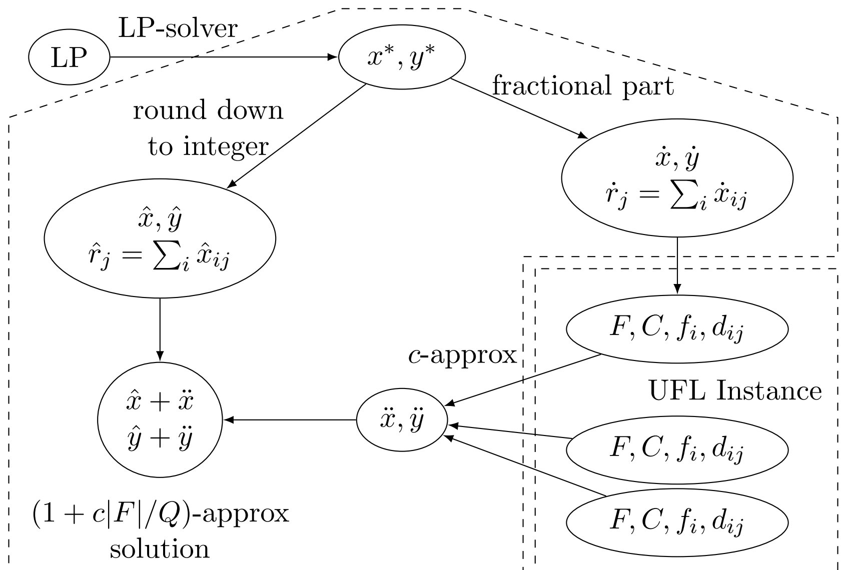


Figure 4.2: Illustration of the approach of reducing FTFP to UFL for the special FTFP instance with all demands r_j being large.

Clearly, by combining the integral solutions to the m copies of UFL instances, we obtain an integral solution to the $\dot{\mathcal{I}}$ instance. We might have to remove some redundant facilities and connections, but this only reduces the total cost. Using a c-approximation algorithm for UFL [2], we obtain a solution to the $\dot{\mathcal{I}}$ instance with a cost not more than $cm \cdot \mathrm{LP}^*_{\mathrm{UFL}}$, where $\mathrm{LP}^*_{\mathrm{UFL}}$ is the cost of an optimal fractional solution to the UFL instance. Moreover, it is easy to see that $(\boldsymbol{x}^*/Q, \boldsymbol{y}^*/Q)$ is a feasible fractional solution to the UFL

[2] For the claim about the cost of the integral solution to $\dot{\mathcal{I}}$ being no more than $cm \cdot \mathrm{LP}^*_{\mathrm{UFL}}$ to hold, we actually need more than the UFL algorithm being c-approximation. What we need is that the integral solution to the UFL instance should have a cost no more than c times the cost of an optimal fractional solution. However, all LP-rounding algorithms for UFL satisfy this requirement.

instance, because we have $r_j \geq Q$ for every client j. It follows that

$$\mathrm{LP}^*_{\mathrm{UFL}} \leq \mathrm{LP}^*/Q,$$

where LP^* is the cost of an optimal fractional solution to the original FTFP instance $\mathcal{I}$. Therefore, we have an integral solution to the instance $\dot{\mathcal{I}}$ with a cost at most $(cm/Q)\mathrm{LP}^*$. Let S_1 be this integral solution, and let $S_2 = (\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ be the solution to the instance $\hat{\mathcal{I}}$. By combining S_1 and S_2, we obtain a feasible integral solution to the instance $\mathcal{I}$. The total cost of this combined solution is no more than

$$\mathit{cost}(S_1) + \mathit{cost}(S_2) \leq (cm/Q)\mathrm{LP}^* + \mathrm{OPT} \leq (1 + cm/Q)\mathrm{OPT} = 1 + O(m/Q)\mathrm{OPT}.$$

As usual, OPT denotes the cost of an optimal integral solution to the original FTFP instance $\mathcal{I}$. We have $\mathrm{LP}^* \leq \mathrm{OPT}$ because the cost of an optimal fractional solution is a lower bound on the cost of any feasible integral solution. ■

4.4 Adaptive Partitioning

In this section we develop our second technique, which we call *adaptive partitioning*. Given an FTFP instance and an optimal fractional solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ to the LP (3.1), we split each client j into r_j individual *unit demand points* (or just *demands*), and we split each site i into not more than $|\mathbb{F}| + 2R|\mathbb{C}|^2$ *facility points* (or *facilities*), where $R = \max_{j\in\mathbb{C}} r_j$. We denote the demand set by $\overline{\mathbb{C}}$ and the facility set by $\overline{\mathbb{F}}$, respectively. We will also partition $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ into a fractional solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ for the split instance. We will typically use symbols ν and μ to index demands and facilities respectively, that is $\bar{\boldsymbol{x}} = (\bar{x}_{\mu\nu})$ and $\bar{\boldsymbol{y}} = (\bar{y}_\mu)$. An illustration is given in Figure 4.3.

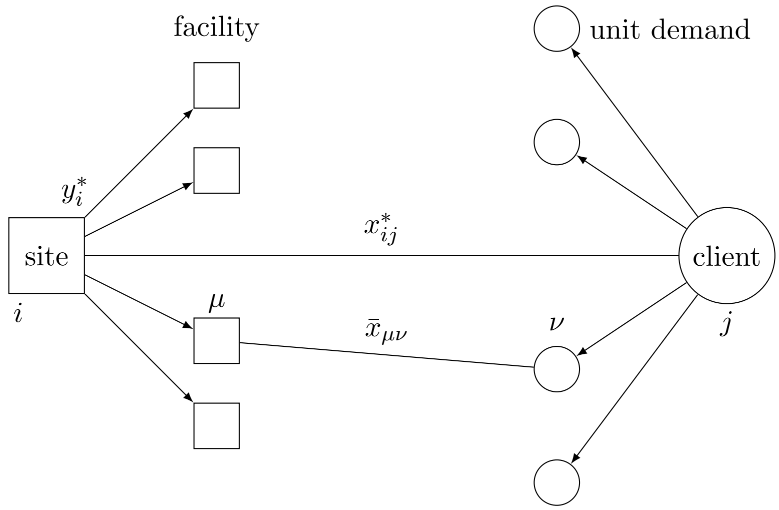


Figure 4.3: A diagram of adaptive partitioning. A site i is split into a set of facilities μ, and a client j is split into a set of unit demands ν. The adaptive partitioning process also defines fractional values for each facility μ and for each pair of μ and ν.

As before, the *neighborhood of a demand* ν is $\overline{N}(\nu) = \{\mu \in \overline{\mathbb{F}} : \bar{x}_{\mu\nu} > 0\}$, see Figure 4.4.

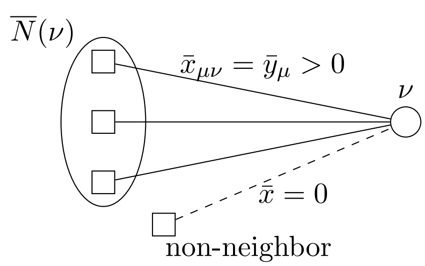


Figure 4.4: Illustration of the neighborhood $\overline{N}(\nu)$ of a demand ν.

We will use notation $\nu \in j$ to mean that ν is a demand of client j; similarly, $\mu \in i$ means that facility μ is on site i. Different demands of the same client (that is, $\nu, \nu' \in j$) are called *siblings*. Further, we use the convention that $f_\mu = f_i$ for $\mu \in i$, $\alpha_\nu^* = \alpha_j^*$ for $\nu \in j$ and $d_{\mu\nu} = d_{\mu j} = d_{ij}$ for $\mu \in i$ and $\nu \in j$. We define $C_\nu^{\text{avg}} = \sum_{\mu \in \overline{N}(\nu)} d_{\mu\nu}\bar{x}_{\mu\nu} = \sum_{\mu \in \overline{\mathbb{F}}} d_{\mu\nu}\bar{x}_{\mu\nu}$. One can think of C_ν^{avg} as the average connection cost of demand ν, if we chose a connection to facility μ with probability $\bar{x}_{\mu\nu}$. In our partitioned fractional solution we guarantee for every ν that $\sum_{\mu\in\overline{\mathbb{F}}}\bar{x}_{\mu\nu}=1$.

Some demands in $\overline{\mathbb{C}}$ will be designated as *primary demands* and the set of primary demands will be denoted by P. By definition we have $P\subseteq\overline{\mathbb{C}}$. In addition, we will use the overlap structure between demand neighborhoods to define a mapping that assigns each demand $\nu\in\overline{\mathbb{C}}$ to some primary demand $\kappa\in P$. As shown in the rounding algorithms in later sections, for each primary demand we guarantee exactly one open facility in its neighborhood, while for a non-primary demand, there is constant probability that none of its neighbors open. In this case we estimate its connection cost by the distance to the facility opened in its assigned primary demand's neighborhood. For this reason the connection cost of a primary demand must be "small" compared to the non-primary demands assigned to it. We also need sibling demands assigned to different primary demands to satisfy the fault-tolerance requirement. Specifically, this partitioning will be constructed to satisfy a number of properties that are detailed below.

(PS) *Partitioned solution.* Vector $(\bar{\boldsymbol{x}},\bar{\boldsymbol{y}})$ is a partition of $(\boldsymbol{x}^*,\boldsymbol{y}^*)$, with unit-value demands, that is,

1. $\sum_{\mu\in\overline{\mathbb{F}}}\bar{x}_{\mu\nu}=1$ for each demand $\nu\in\overline{\mathbb{C}}$.
2. $\sum_{\mu\in i,\nu\in j}\bar{x}_{\mu\nu}=x^*_{ij}$ for each site $i\in\mathbb{F}$ and client $j\in\mathbb{C}$.
3. $\sum_{\mu\in i}\bar{y}_{\mu}=y^*_i$ for each site $i\in\mathbb{F}$.

(CO) *Completeness.* Solution $(\bar{\boldsymbol{x}},\bar{\boldsymbol{y}})$ is complete, that is $\bar{x}_{\mu\nu}\neq 0$ implies $\bar{x}_{\mu\nu}=\bar{y}_{\mu}$, for all $\mu\in\overline{\mathbb{F}},\nu\in\overline{\mathbb{C}}$.

(PD) *Primary demands.* Primary demands satisfy the following conditions:

1. For any two different primary demands $\kappa, \kappa' \in P$ we have $\overline{N}(\kappa) \cap \overline{N}(\kappa') = \emptyset$.

2. For each site $i \in \mathbb{F}$, $\sum_{\mu \in i} \sum_{\kappa \in P} \bar{x}_{\mu\kappa} \leq y_i^*$.

3. Each demand $\nu \in \overline{\mathbb{C}}$ is assigned to one primary demand $\kappa \in P$ such that

   (a) $\overline{N}(\nu) \cap \overline{N}(\kappa) \neq \emptyset$, and

   (b) $C_\nu^{\text{avg}} + \alpha_\nu^* \geq C_\kappa^{\text{avg}} + \alpha_\kappa^*$.

(SI) *Siblings.* For any pair ν, ν' of different siblings we have

1. $\overline{N}(\nu) \cap \overline{N}(\nu') = \emptyset$.

2. If ν is assigned to a primary demand κ then $\overline{N}(\nu') \cap \overline{N}(\kappa) = \emptyset$. In particular, by Property (PD.3(a)), this implies that different sibling demands are assigned to different primary demands.

As we shall demonstrate in Chapter 5, these properties allow us to extend known UFL rounding algorithms to obtain an integral solution to our FTFP problem with a matching approximation ratio. Our partitioning is "adaptive" in the sense that it is constructed one demand at a time, and the connection values for the demands of a client depend on the choice of earlier demands, of this or other clients, and their connection values. We would like to point out that the adaptive partitioning process for the 1.575-approximation algorithm (Section 5.3) is more subtle than that for the 3-approximation (Section 5.1) and the 1.736-approximation algorithms (Section 5.2), due to the introduction of close and far neighborhood.

Implementation of Adaptive Partitioning. We now describe an algorithm for partitioning the instance and the fractional solution so that the properties (PS), (CO), (PD), and (SI) are satisfied. Recall that $\overline{\mathbb{F}}$ and $\overline{\mathbb{C}}$, respectively, denote the sets of facilities and demands that will be created in this stage, and $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ is the partitioned solution to be computed.

The adaptive partitioning algorithm consists of two phases: Phase 1 is called the partitioning phase and Phase 2 is called the augmenting phase. Phase 1 is done in iterations, where in each iteration we find the "best" client j and create a new demand ν out of it. This demand either becomes a primary demand itself, or it is assigned to some existing primary demand. We call a client j *exhausted* when all its r_j demands have been created and assigned to some primary demands. Phase 1 completes when all clients are exhausted. In Phase 2 we ensure that every demand has a total connection values $\bar{x}_{\mu\nu}$ equal to 1, that is condition (PS.1).

For each site i we will initially create one "big" facility μ with initial value $\bar{y}_\mu = y_i^*$. While we partition the instance, creating new demands and connections, this facility may end up being split into more facilities to preserve completeness of the fractional solution. Also, we will gradually decrease the fractional connection vector for each client j, to account for the demands already created for j and their connection values. These decreased connection values will be stored in an auxiliary vector $\widetilde{\boldsymbol{x}}$. The intuition is that $\widetilde{\boldsymbol{x}}$ represents the part of $\boldsymbol{x}^*$ that still has not been allocated to existing demands and future demands can use $\widetilde{\boldsymbol{x}}$ for their connections. For technical reasons, $\widetilde{\boldsymbol{x}}$ will be indexed by facilities (rather than sites) and clients, that is $\widetilde{\boldsymbol{x}} = (\widetilde{x}_{\mu j})$. At the beginning, we set $\widetilde{x}_{\mu j} \leftarrow x_{ij}^*$ for each $j \in \mathbb{C}$, where $\mu \in i$ is the single facility created initially at site i. At each step, whenever we create a new demand ν for a client j, we will define its values $\bar{x}_{\mu\nu}$ and appropriately reduce the values $\widetilde{x}_{\mu j}$, for all facilities μ. We will deal with two types of neighborhoods, with respect to $\widetilde{\boldsymbol{x}}$ and $\bar{\boldsymbol{x}}$, that is $\widetilde{N}(j) = \{\mu \in \overline{\mathbb{F}} : \widetilde{x}_{\mu j} > 0\}$ for $j \in \mathbb{C}$ and $\overline{N}(\nu) = \{\mu \in \overline{\mathbb{F}} : \bar{x}_{\mu\nu} > 0\}$ for $\nu \in \overline{\mathbb{C}}$. During this process we preserve the completeness (CO) of the fractional solutions $\widetilde{\boldsymbol{x}}$ and $\bar{\boldsymbol{x}}$. More precisely, the following properties will hold for every facility μ after every iteration,

(c1) For each demand ν either $\bar{x}_{\mu\nu} = 0$ or $\bar{x}_{\mu\nu} = \bar{y}_\mu$. This is the same condition as condition (CO), yet we repeat it here as (c1) needs to hold after every iteration, while condition (CO) only applies to the final partitioned fractional solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$.

(c2) For each client j, either $\widetilde{x}_{\mu j} = 0$ or $\widetilde{x}_{\mu j} = \bar{y}_\mu$.

A full description of the algorithm is given in Pseudocode 1. Initially, the set U of non-exhausted clients contains all clients, the set $\overline{\mathbb{C}}$ of demands is empty, the set $\overline{\mathbb{F}}$ of facilities consists of one facility μ on each site i with $\bar{y}_\mu = y_i^*$, and the set P of primary demands is empty (Lines 1–4). In one iteration of the while loop (Lines 5–8), for each client j we compute a quantity called $\mathrm{tcc}(j)$ (tentative connection cost), that represents the average distance from j to the set $\widetilde{N}_1(j)$ of the nearest facilities μ whose total connection value to j (the sum of $\widetilde{x}_{\mu j}$'s) equals 1. This set is computed by Procedure NearestUnitChunk() (see Pseudocode 2, Lines 1–9), which adds facilities to $\widetilde{N}_1(j)$ in order of non-decreasing distance, until the total connection value is exactly 1. (The procedure actually uses the $\bar{y}_\mu$ values, which are equal to the connection values, by the completeness condition (c2).) This may require splitting the last added facility and adjusting the connection values so that conditions (c1) and (c2) are preserved.

Pseudocode 1 Algorithm: Adaptive Partitioning

Input: $\mathbb{F}$, $\mathbb{C}$, $(\boldsymbol{x}^*, \boldsymbol{y}^*)$
Output: $\overline{\mathbb{F}}$, $\overline{\mathbb{C}}$, $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ ▷ Unspecified $\bar{x}_{\mu\nu}$'s and $\widetilde{x}_{\mu j}$'s are assumed to be 0
1: $\widetilde{\boldsymbol{r}} \leftarrow \boldsymbol{r}, U \leftarrow \mathbb{C}, \overline{\mathbb{F}} \leftarrow \emptyset, \overline{\mathbb{C}} \leftarrow \emptyset, P \leftarrow \emptyset$ ▷ Phase 1
2: **for** each site $i \in \mathbb{F}$ **do**
3:     create a facility μ at i and add μ to $\overline{\mathbb{F}}$
4:     $\bar{y}_\mu \leftarrow y^*_i$ and $\widetilde{x}_{\mu j} \leftarrow x^*_{ij}$ for each $j \in \mathbb{C}$
5: **while** $U \neq \emptyset$ **do**
6:     **for** each $j \in U$ **do**
7:         $\widetilde{N}_1(j) \leftarrow \textsc{NearestUnitChunk}(j, \overline{\mathbb{F}}, \widetilde{\boldsymbol{x}}, \bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ ▷ see Pseudocode 2
8:         $\mathrm{tcc}(j) \leftarrow \sum_{\mu \in \widetilde{N}_1(j)} d_{\mu j} \cdot \widetilde{x}_{\mu j}$
9:     $p \leftarrow \arg\min_{j \in U}\{\mathrm{tcc}(j) + \alpha^*_j\}$
10:     create a new demand ν for client p
11:     **if** $\widetilde{N}_1(p) \cap \overline{N}(\kappa) \neq \emptyset$ for some primary demand $\kappa \in P$ **then**
12:         assign ν to κ
13:         $\bar{x}_{\mu\nu} \leftarrow \widetilde{x}_{\mu p}$ and $\widetilde{x}_{\mu p} \leftarrow 0$ for each $\mu \in \widetilde{N}(p) \cap \overline{N}(\kappa)$
14:     **else**
15:         make ν primary, $P \leftarrow P \cup \{\nu\}$, assign ν to itself
16:         set $\bar{x}_{\mu\nu} \leftarrow \widetilde{x}_{\mu p}$ and $\widetilde{x}_{\mu p} \leftarrow 0$ for each $\mu \in \widetilde{N}_1(p)$
17:     $\overline{\mathbb{C}} \leftarrow \overline{\mathbb{C}} \cup \{\nu\}, \widetilde{r}_p \leftarrow \widetilde{r}_p - 1$
18:     **if** $\widetilde{r}_p = 0$ **then** $U \leftarrow U \setminus \{p\}$
19: **for** each client $j \in \mathbb{C}$ **do** ▷ Phase 2
20:     **for** each demand $\nu \in j$ **do** ▷ each client j has r_j demands
21:         **if** $\sum_{\mu \in \overline{N}(\nu)} \bar{x}_{\mu\nu} < 1$ **then** $\textsc{AugmentToUnit}(\nu, j, \overline{\mathbb{F}}, \widetilde{\boldsymbol{x}}, \bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ ▷ see Pseudocode 2

The next step is to pick a client p with minimum $\mathrm{tcc}(p) + \alpha^*_p$ and create a demand ν for p (Lines 9–10). If $\widetilde{N}_1(p)$ overlaps the neighborhood of some existing primary demand κ (if there are multiple such κ's, pick any of them), we assign ν to κ, and ν acquires all the connection values $\widetilde{x}_{\mu p}$ between client p and facility μ in $\widetilde{N}(p) \cap \overline{N}(\kappa)$ (Lines 11–13). Note that although we check for overlap with $\widetilde{N}_1(p)$, we then move all facilities in the intersection with $\widetilde{N}(p)$, a bigger set, into $\overline{N}(\nu)$. An example is given in Figure 4.5.

Pseudocode 2 Helper functions used in Pseudocode 1

1: **function** NearestUnitChunk$(j, \overline{\mathbb{F}}, \widetilde{\boldsymbol{x}}, \bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ ▷ upon return, $\sum_{\mu \in \widetilde{N}_1(j)} \widetilde{x}_{\mu j} = 1$
2: Let $\widetilde{N}(j) = \{\mu_1, ..., \mu_q\}$ where $d_{\mu_1 j} \leq d_{\mu_2 j} \leq \ldots \leq d_{\mu_q j}$
3: Let l be such that $\sum_{k=1}^{l} \bar{y}_{\mu_k} \geq 1$ and $\sum_{k=1}^{l-1} \bar{y}_{\mu_k} < 1$
4: Create a new facility σ at the same site as μ_l and add it to $\overline{\mathbb{F}}$ ▷ split μ_l
5: Set $\bar{y}_\sigma \leftarrow \sum_{k=1}^{l} \bar{y}_{\mu_k} - 1$ and $\bar{y}_{\mu_l} \leftarrow \bar{y}_{\mu_l} - \bar{y}_\sigma$
6: For each $\nu \in \overline{\mathbb{C}}$ with $\bar{x}_{\mu_l \nu} > 0$ set $\bar{x}_{\mu_l \nu} \leftarrow \bar{y}_{\mu_l}$ and $\bar{x}_{\sigma \nu} \leftarrow \bar{y}_\sigma$
7: For each $j' \in \mathbb{C}$ with $\widetilde{x}_{\mu_l j'} > 0$ (including j) set $\widetilde{x}_{\mu_l j'} \leftarrow \bar{y}_{\mu_l}$ and $\widetilde{x}_{\sigma j'} \leftarrow \bar{y}_\sigma$
8: (All other new connection values are set to 0)
9: **return** $\widetilde{N}_1(j) = \{\mu_1, \ldots, \mu_{l-1}, \mu_l\}$
10: **function** AugmentToUnit$(\nu, j, \overline{\mathbb{F}}, \widetilde{\boldsymbol{x}}, \bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ ▷ ν is a demand of client j
11: **while** $\sum_{\mu \in \overline{\mathbb{F}}} \bar{x}_{\mu\nu} < 1$ **do** ▷ upon return, $\sum_{\mu \in \overline{N}(\nu)} \bar{x}_{\mu\nu} = 1$
12: Let η be any facility such that $\widetilde{x}_{\eta j} > 0$
13: **if** $1 - \sum_{\mu \in \overline{\mathbb{F}}} \bar{x}_{\mu\nu} \geq \widetilde{x}_{\eta j}$ **then**
14: $\bar{x}_{\eta\nu} \leftarrow \widetilde{x}_{\eta j}, \widetilde{x}_{\eta j} \leftarrow 0$
15: **else**
16: Create a new facility σ at the same site as η and add it to $\overline{\mathbb{F}}$ ▷ split η
17: Let $\bar{y}_\sigma \leftarrow 1 - \sum_{\mu \in \overline{\mathbb{F}}} \bar{x}_{\mu\nu}, \bar{y}_\eta \leftarrow \bar{y}_\eta - \bar{y}_\sigma$
18: Set $\bar{x}_{\sigma\nu} \leftarrow \bar{y}_\sigma,\ \bar{x}_{\eta\nu} \leftarrow 0,\ \widetilde{x}_{\eta j} \leftarrow \bar{y}_\eta,\ \widetilde{x}_{\sigma j} \leftarrow 0$
19: For each $\nu' \neq \nu$ with $\bar{x}_{\eta\nu'} > 0$, set $\bar{x}_{\eta\nu'} \leftarrow \bar{y}_\eta,\ \bar{x}_{\sigma\nu'} \leftarrow \bar{y}_\sigma$
20: For each $j' \neq j$ with $\widetilde{x}_{\eta j'} > 0$, set $\widetilde{x}_{\eta j'} \leftarrow \bar{y}_\eta, \widetilde{x}_{\sigma j'} \leftarrow \bar{y}_\sigma$
21: (All other new connection values are set to 0)

The other case is when $\widetilde{N}_1(p)$ is disjoint from the neighborhoods of all existing primary demands. Then, in Lines 15–16, ν becomes itself a primary demand and we assign ν to itself. It also inherits the connection values to all facilities $\mu \in \widetilde{N}_1(p)$ from p (recall that $\widetilde{x}_{\mu p} = \bar{y}_\mu$), with all other $\bar{x}_{\mu\nu}$ values set to 0. An example for this case is given in Figure 4.6.

At this point all primary demands satisfy Property (PS.1), but this may not be true for non-primary demands. For those demands we still may need to adjust the $\bar{x}_{\mu\nu}$ values so that the total connection value for ν, that is $\mathrm{conn}(\nu) \stackrel{\text{def}}{=} \sum_{\mu \in \overline{\mathbb{F}}} \bar{x}_{\mu\nu}$, is equal 1. This is accomplished by Procedure AugmentToUnit() (definition in Pseudocode 2, Lines 10–21) that allocates to $\nu \in j$ some of the remaining connection values $\widetilde{x}_{\mu j}$ of client j (Lines 19–21).

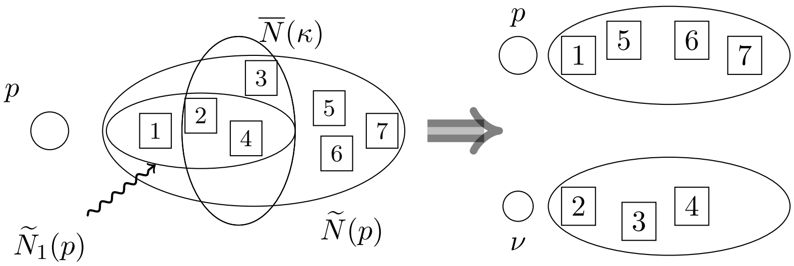


Figure 4.5: An example of the first case in Phase 1 of adaptive partitioning. Here p is the client that creates a demand ν in this iteration, and $\widetilde{N}_1(p)$ overlaps with $\overline{N}(\kappa)$ for some primary demand κ. Facilities 2 and 4 are in the intersection of $\widetilde{N}_1(p)$ and $\overline{N}(\kappa)$ for the overlap check. We move these two facilities, and additionally facility 3, into $\overline{N}(\nu)$, because facility 3 is also in the intersection of $\widetilde{N}(p)$ and $\overline{N}(\kappa)$.

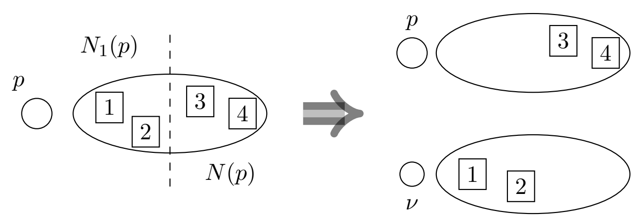


Figure 4.6: An example of the second case in Phase 1 of adaptive partitioning. Here p is the client that creates a demand ν and ν becomes a primary demand with a neighborhood of $\widetilde{N}_1(p)$, which consists of facilities 1 and 2 in this example. The neighborhood $\widetilde{N}(p)$ keeps the remaining facilities; that is facilities 3 and 4 in this example.

AugmentToUnit() will repeatedly pick any facility η with $\widetilde{x}_{\eta j} > 0$. If $\widetilde{x}_{\eta j} \leq 1 - \mathrm{conn}(\nu)$, then the connection value $\widetilde{x}_{\eta j}$ is reassigned to ν. Otherwise, $\widetilde{x}_{\eta j} > 1 - \mathrm{conn}(\nu)$, in which case we split η so that connecting ν to one of the created copies of η will make $\mathrm{conn}(\nu)$ equal 1, and we'll be done. An example in Figure 4.7 illustrates this process.

Notice that we start with $|\mathbb{F}|$ facilities and in each iteration of the while loop in Line 5 (Pseudocode 1) each client causes at most one split. We have a total of not more than $R|\mathbb{C}|$ iterations as in each iteration we create one demand. (Recall that $R = \max_j r_j$.) In Phase 2 we do an augment step for each demand ν and this creates not more than $R|\mathbb{C}|$ new facilities. So the total number of facilities we created will be at most $|\mathbb{F}| + R|\mathbb{C}|^2 + R|\mathbb{C}| \leq$

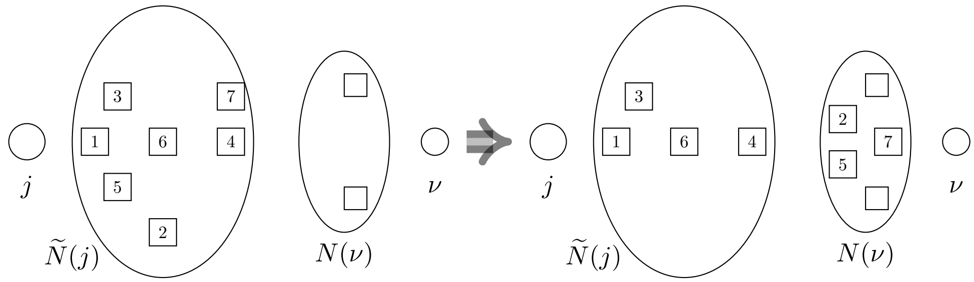


Figure 4.7: An example of one step in Phase 2 of adaptive partitioning. In this example, we move facilities $2, 5$ and 7 from $\widetilde{N}(j)$ to $\overline{N}(\nu)$, to make $\overline{N}(\nu)$ have a total fractional value of 1.

$|\mathbb{F}| + 2R|\mathbb{C}|^2$, which is polynomial in $|\mathbb{F}| + |\mathbb{C}|$ due to our earlier bound on R.

Example. We now illustrate our partitioning algorithm with an example, where the FTFP instance has four sites and four clients. The demands are $r_1 = 1$ and $r_2 = r_3 = r_4 = 2$. The facility costs are $f_i = 1$ for all i. The distances are defined as follows: $d_{ii} = 3$ for $i = 1, 2, 3, 4$ and $d_{ij} = 1$ for all $i \neq j$. Solving the LP(3.1), we obtain the fractional solution given in Table 4.1a.

x^*_{ij}	1	2	3	4	y^*_i
1	0	$\frac{4}{3}$	$\frac{4}{3}$	$\frac{4}{3}$	$\frac{4}{3}$
2	$\frac{1}{3}$	0	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
3	$\frac{1}{3}$	$\frac{1}{3}$	0	$\frac{1}{3}$	$\frac{1}{3}$
4	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	0	$\frac{1}{3}$

(a)

$\bar{x}_{\mu\nu}$	$1'$	$2'$	$2''$	$3'$	$3''$	$4'$	$4''$	$\bar{y}_\mu$
$\dot{1}$	0	1	0	1	0	1	0	1
$\ddot{1}$	0	0	$\frac{1}{3}$	0	$\frac{1}{3}$	0	$\frac{1}{3}$	$\frac{1}{3}$
$\dot{2}$	$\frac{1}{3}$	0	0	0	$\frac{1}{3}$	0	$\frac{1}{3}$	$\frac{1}{3}$
$\dot{3}$	$\frac{1}{3}$	0	$\frac{1}{3}$	0	0	0	$\frac{1}{3}$	$\frac{1}{3}$
$\dot{4}$	$\frac{1}{3}$	0	$\frac{1}{3}$	0	$\frac{1}{3}$	0	0	$\frac{1}{3}$

(b)

Table 4.1: An example of an execution of the partitioning algorithm. (a) An optimal fractional solution x^*, y^*. (b) The partitioned solution. j' and j'' denote the first and second demand of a client j, and i and $\ddot{i}$ denote the first and second facility at site i.

It is easily seen that the fractional solution in Table 4.1a is optimal and complete ($x^*_{ij} > 0$ implies $x^*_{ij} = y^*_i$). The dual optimal solution has all $\alpha^*_j = 4/3$ for $j = 1, 2, 3, 4$.

Now we perform Phase 1, the adaptive partitioning, following the description in Pseudocode 1. To streamline the presentation, we assume that all ties are broken in favor of lower-numbered clients, demands or facilities. First we create one facility at each of the four sites, denoted as $\dot{1}$, $\dot{2}$, $\dot{3}$ and $\dot{4}$ (Line 2–4, Pseudocode 1). We then execute the "while" loop in Line 5 Pseudocode 1. This loop will have seven iterations. Consider the first iteration. In Line 7–8 we compute $\mathrm{tcc}(j)$ for each client $j = 1, 2, 3, 4$ in U. When computing $\widetilde{N}_1(2)$, facility $\dot{1}$ will get split into $\dot{1}$ and $\ddot{1}$ with $\bar{y}_{\dot{1}} = 1$ and $\bar{y}_{\ddot{1}} = 1/3$. (This will happen in Line 4–7 of Pseudocode 2.) Then, in Line 9 we will pick client $p = 1$ and create a demand denoted as $1'$ (see Table 4.1b). Since there are no primary demands yet, we make $1'$ a primary demand with $\overline{N}(1') = \widetilde{N}_1(1) = \{\dot{2}, \dot{3}, \dot{4}\}$. Notice that client 1 is exhausted after this iteration and U becomes $\{2, 3, 4\}$.

In the second iteration we compute $\mathrm{tcc}(j)$ for $j = 2, 3, 4$ and pick client $p = 2$, from which we create a new demand $2'$. We have $\widetilde{N}_1(2) = \{\dot{1}\}$, which is disjoint from $\overline{N}(1')$. So we create a demand $2'$ and make it primary, and set $\overline{N}(2') = \{\dot{1}\}$. In the third iteration we compute $\mathrm{tcc}(j)$ for $j = 2, 3, 4$ and again we pick client $p = 2$. Since $\widetilde{N}_1(2) = \{\ddot{1}, \dot{3}, \dot{4}\}$ overlaps with $\overline{N}(1')$, we create a demand $2''$ and assign it to $1'$. We also set $\overline{N}(2'') = \overline{N}(1') \cap \widetilde{N}(2) = \{\dot{3}, \dot{4}\}$. After this iteration client 2 is exhausted and we have $U = \{3, 4\}$.

In the fourth iteration we compute $\mathrm{tcc}(j)$ for client $j = 3, 4$. We pick $p = 3$ and create demand $3'$. Since $\widetilde{N}_1(3) = \{\dot{1}\}$ overlaps $\overline{N}(2')$, we assign $3'$ to $2'$ and set $\overline{N}(3') = \{\dot{1}\}$.

In the fifth iteration we compute $\text{tcc}(j)$ for client $j = 3, 4$ and pick $p = 3$ again. At this time $\widetilde{N}_1(3) = \{\ddot{1}, \dot{2}, \dot{4}\}$, which overlaps with $\overline{N}(1')$. So we create a demand $3''$ and assign it to $1'$, as well as set $\overline{N}(3'') = \{\dot{2}, \dot{4}\}$.

In the last two iterations we will pick client $p = 4$ twice and create demands $4'$ and $4''$. For $4'$ we have $\widetilde{N}_1(4) = \{\dot{1}\}$ so we assign $4'$ to $2'$ and set $\overline{N}(4') = \{\dot{1}\}$. For $4''$ we have $\widetilde{N}_1(4) = \{\ddot{1}, \dot{2}, \dot{3}\}$ and we assign it to $1'$, as well as set $\overline{N}(4'') = \{\dot{2}, \dot{3}\}$.

Now that all clients are exhausted we perform Phase 2, the augmenting phase, to construct a fractional solution in which all demands have total connection value equal to 1. We iterate through each of the seven demands created, that is $1', 2', 2'', 3', 3'', 4', 4''$. $1'$ and $2'$ already have neighborhoods with total connection value of 1, so nothing will change in the first two iterations. $2''$ has $\dot{3}, \dot{4}$ in its neighborhood, with total connection value of $2/3$, and $\widetilde{N}(2) = \{\ddot{1}\}$ at this time, so we add $\ddot{1}$ into $\overline{N}(2'')$ to make $\overline{N}(2'') = \{\ddot{1}, \dot{3}, \dot{4}\}$ and now $2''$ has total connection value of 1. Similarly, $3''$ and $4''$ each get $\ddot{1}$ added to their neighborhood and end up with total connection value of 1. The other two demands, namely $3'$ and $4'$, each have $\dot{1}$ in its neighborhood so each of them has already its total connection value equal 1. This completes Phase 2.

The final partitioned fractional solution is given in Table 4.1b. We have created a total of five facilities $\dot{1}, \ddot{1}, \dot{2}, \dot{3}, \dot{4}$, and seven demands, $1', 2', 2'', 3', 3'', 4', 4''$. It can be verified that all the stated properties are satisfied.

Correctness. We now show that all the required properties (PS), (CO), (PD) and (SI) are satisfied by the above construction.

Properties (PS) and (CO) follow directly from the algorithm. (CO) is implied by the completeness condition (c1) that the algorithm maintains after each iteration. Condition (PS.1) is a result of calling Procedure AugmentToUnit() in Line 21. To see that (PS.2) holds, note that at each step the algorithm maintains the invariant that, for every $i \in \mathbb{F}$ and $j \in \mathbb{C}$, we have $\sum_{\mu \in i} \sum_{\nu \in j} \bar{x}_{\mu\nu} + \sum_{\mu \in i} \widetilde{x}_{\mu j} = x^*_{ij}$. In the end, we will create r_j demands for each client j, with each demand $\nu \in j$ satisfying (PS.1), and thus $\sum_{\nu \in j} \sum_{\mu \in \overline{\mathbb{F}}} \bar{x}_{\mu\nu} = r_j$. This implies that $\widetilde{x}_{\mu j} = 0$ for every facility $\mu \in \overline{\mathbb{F}}$, and (PS.2) follows. (PS.3) holds because every time we split a facility μ into μ' and μ'', the sum of $\bar{y}_{\mu'}$ and $\bar{y}_{\mu''}$ is equal to the old value of $\bar{y}_{\mu}$.

Now we deal with properties in group (PD). First, (PD.1) follows directly from the algorithm, Pseudocode 1 (Lines 14–16), since every primary demand has its neighborhood fixed when created, and that neighborhood is disjoint from those of the existing primary demands.

Property (PD.2) follows from (PD.1), (CO) and (PS.3). In more detail, it can be justified as follows. By (PD.1), for each $\mu \in i$ there is at most one $\kappa \in P$ with $\bar{x}_{\mu\kappa} > 0$ and we have $\bar{x}_{\mu\kappa} = \bar{y}_{\mu}$ due do (CO). Let $K \subseteq i$ be the set of those μ's for which such $\kappa \in P$ exists, and denote this κ by κ_μ. Then, using conditions (CO) and (PS.3), we have $\sum_{\mu \in i} \sum_{\kappa \in P} \bar{x}_{\mu\kappa} = \sum_{\mu \in K} \bar{x}_{\mu\kappa_\mu} = \sum_{\mu \in K} \bar{y}_{\mu} \leq \sum_{\mu \in i} \bar{y}_{\mu} = y^*_i$.

Property (PD.3(a)) follows from the way the algorithm assigns primary demands. When demand ν of client p is assigned to a primary demand κ in Lines 11–13 of Pseudocode 1, we move all facilities in $\widetilde{N}(p) \cap \overline{N}(\kappa)$ (the intersection is nonempty) into $\overline{N}(\nu)$, and we never remove a facility from $\overline{N}(\nu)$. We postpone the proof for (PD.3(b)) to Lemma 9.

Finally we argue that the properties in group (SI) hold. (SI.1) is easy, since for any client j, each facility μ is added to the neighborhood of at most one demand $\nu \in j$, by setting $\bar{x}_{\mu\nu}$ to $\bar{y}_{\mu}$, while other siblings ν' of ν have $\bar{x}_{\mu\nu'} = 0$. Note that right after a demand $\nu \in p$ is created, its neighborhood is disjoint from the neighborhood of p, that is $\overline{N}(\nu) \cap \widetilde{N}(p) = \emptyset$, by Lines 11–13 of the algorithm. Thus all demands of p created later will have neighborhoods disjoint from the set $\overline{N}(\nu)$ before the augmenting phase 2. Furthermore, Procedure AugmentToUnit() preserves this property, because when it adds a facility to $\overline{N}(\nu)$ then it removes it from $\widetilde{N}(p)$, and in case of splitting, one resulting facility is added to $\overline{N}(\nu)$ and the other to $\widetilde{N}(p)$. Property (SI.2) is shown below in Lemma 7.

It remains to show Properties (PD.3(b)) and (SI.2). We show them in the lemmas below, thus completing the description of our adaptive partitioning process.

Lemma 7 *Property (SI.2) holds after the Adaptive Partitioning stage.*

Proof. Let $\nu_1, \ldots, \nu_{r_j}$ be the demands of a client $j \in \mathbb{C}$, listed in the order of creation, and, for each $q = 1, 2, \ldots, r_j$, denote by κ_q the primary demand that ν_q is assigned to. After the completion of Phase 1 of Pseudocode 1 (Lines 5–18), we have $\overline{N}(\nu_s) \subseteq \overline{N}(\kappa_s)$ for $s = 1, \ldots, r_j$. Since any two primary demands have disjoint neighborhoods, we have $\overline{N}(\nu_s) \cap \overline{N}(\kappa_q) = \emptyset$ for any $s \neq q$, that is Property (SI.2) holds right after Phase 1.

After Phase 1 all neighborhoods $\overline{N}(\kappa_s), s = 1, \ldots, r_j$ have already been fixed and they do not change in Phase 2. None of the facilities in $\widetilde{N}(j)$ appear in any of $\overline{N}(\kappa_s)$ for $s = 1, \ldots, r_j$, by the way we allocate facilities in Lines 13 and 16. Therefore during the augmentation process in Phase 2, when we add facilities from $\widetilde{N}(j)$ to $\overline{N}(\nu)$, for some $\nu \in j$ (Line 19–21 of Pseudocode 1), all the required disjointedness conditions will be preserved.

■

We need one more lemma before proving our last property (PD.3(b)). For a client j and a demand ν, we use notation $\mathrm{tcc}^{\nu}(j)$ for the value of $\mathrm{tcc}(j)$ at the time when ν was created. (It is not necessary that $\nu \in j$ but we assume that j is not exhausted at that time.)

Lemma 8 *Let η and ν be two demands, with η created no later than ν, and let $j \in \mathbb{C}$ be a client that is not exhausted when ν is created. Then we have*

(a) $\mathrm{tcc}^{\eta}(j) \le \mathrm{tcc}^{\nu}(j)$*, and*

(b) if $\nu \in j$ then $\mathrm{tcc}^{\eta}(j) \le C_{\nu}^{\mathrm{avg}}$.

Proof. We focus first on the time when demand η is about to be created, right after the call to NearestUnitChunk() in Pseudocode 1, Line 7. Let $\widetilde{N}(j) = \{\mu_1, ..., \mu_q\}$ with all facilities μ_s ordered according to non-decreasing distance from j. Consider the following linear program,

$$\boxed{\begin{array}{rl} \text{minimize} & \displaystyle\sum_s d_{\mu_s j} z_s \\ \text{subject to} & \displaystyle\sum_s z_s \ge 1 \\ & 0 \le z_s \le \widetilde{x}_{\mu_s j} \quad \text{for all } s \end{array}}$$

This is a fractional minimum knapsack covering problem (with knapsack size equal 1) and its optimal fractional solution is the greedy solution, whose value is exactly $\mathrm{tcc}^{\eta}(j)$.

On the other hand, we claim that $\mathrm{tcc}^{\nu}(j)$ can be thought of as the value of some feasible solution to this linear program, and that the same is true for C_{ν}^{avg} if $\nu \in j$. Indeed, each of these quantities involves some later values $\widetilde{x}_{\mu j}$, where μ could be one of the facilities μ_s or a new facility obtained from splitting. For each s, however, the sum of all values $\widetilde{x}_{\mu j}$, over the facilities μ that were split from μ_s, cannot exceed the value $\widetilde{x}_{\mu_s j}$ at the time when

η was created, because splitting facilities preserves this sum and creating new demands for j can only decrease it. Therefore both quantities $\mathrm{tcc}^{\nu}(j)$ and C^{avg}_{ν} (for $\nu \in j$) correspond to some choice of the z_s variables (adding up to 1), and the lemma follows. ■

Lemma 9 *Property (PD.3(b)) holds after the Adaptive Partitioning stage.*

Proof. Suppose that demand $\nu \in j$ is assigned to some primary demand $\kappa \in p$. Then

$$C^{\mathrm{avg}}_{\kappa} + \alpha^*_{\kappa} \;=\; \mathrm{tcc}^{\kappa}(p) + \alpha^*_p \;\le\; \mathrm{tcc}^{\kappa}(j) + \alpha^*_j \;\le\; C^{\mathrm{avg}}_{\nu} + \alpha^*_{\nu}.$$

We now justify this derivation. By definition we have $\alpha^*_{\kappa} = \alpha^*_p$. Further, by the algorithm, if κ is a primary demand of client p, then $C^{\mathrm{avg}}_{\kappa}$ is equal to $\mathrm{tcc}(p)$ computed when κ is created, which is exactly $\mathrm{tcc}^{\kappa}(p)$. Thus the first equation is true. The first inequality follows from the choice of p in Line 9 in Pseudocode 1. The last inequality holds because $\alpha^*_j = \alpha^*_{\nu}$ (due to $\nu \in j$), and because $\mathrm{tcc}^{\kappa}(j) \le C^{\mathrm{avg}}_{\nu}$, which follows from Lemma 8. ■

We have thus proved that all properties (PS), (CO), (PD) and (SI) hold for our partitioned fractional solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$. In the following sections we show how to use these properties to round the fractional solution to an approximate integral solution. For the 3-approximation algorithm (Section 5.1) and the 1.736-approximation algorithm (Section 5.2), the first phase of the algorithm is exactly the same partitioning process as described above. However, the 1.575-approximation algorithm (Section 5.3) demands a more sophisticated partitioning process as the interplay between close and far neighborhood of sibling demands result in more delicate properties that our partitioned fractional solution must satisfy.

Chapter 5

LP-rounding Algorithms

In Section 4.4 of Chapter 4, we have seen that the adaptive partitioning technique produces a fractional solution for individual facilities and unit demand points with a number of structural properties. In this chapter we show how those properties help in designing LP-rounding algorithms with good approximation ratios. We start with a simple algorithm with ratio 3 that illustrates the main steps of a rounding algorithm and the use of the structural properties to derive an approximation ratio. A more refined rounding algorithm with ratio 1.736 is presented next, using the same partitioned fractional solution as a starting point. Our best approximation algorithm with ratio 1.575 is presented last. This algorithm needs a fractional solution with more sophisticated structure, compared to the one used by the first two algorithms.

5.1 Algorithm EGUP with Ratio 3

The algorithm we describe in this section achieves ratio 3. Although this is still quite far from our best ratio 1.575 that we derive later, we include this algorithm in the paper to illustrate, in a relatively simple setting, how the properties of our partitioned fractional solution are used in rounding it to an integral solution with cost not too far away from an optimal solution. The rounding approach we use here is an extension of the corresponding method for UFL described in [23].

Algorithm EGUP**.** At a high level, we would open exactly one facility for each primary demand κ, and each non-primary demand is connected to the facility opened for the primary demand it was assigned to.

More precisely, we apply a rounding process, guided by the fractional values $(\bar{y}_\mu)$ and $(\bar{x}_{\mu\nu})$, that produces an integral solution. This integral solution is obtained by choosing a subset of facilities in $\overline{\mathbb{F}}$ to open, and for each demand in $\overline{\mathbb{C}}$, specifying an open facility that this demand will be connected to. For each primary demand $\kappa \in P$, we want to open one facility $\phi(\kappa) \in \overline{N}(\kappa)$. To this end, we use randomization: for each $\mu \in \overline{N}(\kappa)$, we choose $\phi(\kappa) = \mu$ with probability $\bar{x}_{\mu\kappa}$, ensuring that exactly one $\mu \in \overline{N}(\kappa)$ is chosen. Note that $\sum_{\mu \in \overline{N}(\kappa)} \bar{x}_{\mu\kappa} = 1$, so this distribution is well-defined. We open this facility $\phi(\kappa)$ and connect to $\phi(\kappa)$ all demands that are assigned to κ.

In our description above, the algorithm is presented as a randomized algorithm. It can be de-randomized using the method of conditional expectations, which is commonly used in approximation algorithms for facility location problems and standard enough that presenting it here would be redundant. Readers less familiar with this field are recommended to consult [15], where the method of conditional expectations is applied in a context very similar to ours.

Analysis. We now bound the expected facility cost and connection cost by establishing the two lemmas below.

Lemma 10 *The expectation of facility cost F_{EGUP} of our solution is at most F^*.*

Proof. By Property (PD.1), the neighborhoods of primary demands are disjoint. Also, for any primary demand $\kappa \in P$, the probability that a facility $\mu \in \overline{N}(\kappa)$ is chosen as the open facility $\phi(\kappa)$ is $\bar{x}_{\mu\kappa}$. Hence the expected total facility cost is

$$\begin{aligned}
\mathbb{E}[F_{\mathrm{EGUP}}] &= \textstyle\sum_{\kappa \in P} \sum_{\mu \in \overline{N}(\kappa)} f_\mu \bar{x}_{\mu\kappa} \\
&= \textstyle\sum_{\kappa \in P} \sum_{\mu \in \overline{\mathbb{F}}} f_\mu \bar{x}_{\mu\kappa} \\
&= \textstyle\sum_{i \in \mathbb{F}} f_i \sum_{\mu \in i} \sum_{\kappa \in P} \bar{x}_{\mu\kappa} \\
&\leq \textstyle\sum_{i \in \mathbb{F}} f_i y_i^* = F^*,
\end{aligned}$$

where the inequality follows from Property (PD.2). ■

Lemma 11 *The expectation of connection cost C_{EGUP} of our solution is at most $C^* + 2 \cdot \mathrm{LP}^*$.*

Proof. For a primary demand κ, its expected connection cost is C^{avg}_κ because we choose facility μ with probability $\bar{x}_{\mu\kappa}$.

Consider a non-primary demand ν assigned to a primary demand $\kappa \in P$. Let μ be any facility in $\overline{N}(\nu) \cap \overline{N}(\kappa)$. Since μ is in both $\overline{N}(\nu)$ and $\overline{N}(\kappa)$, we have $d_{\mu\nu} \leq \alpha^*_\nu$ and $d_{\mu\kappa} \leq \alpha^*_\kappa$ (This follows from the complementary slackness conditions since $\alpha^*_\nu = \beta^*_{\mu\nu} + d_{\mu\nu}$

for each $\mu \in \overline{N}(\nu)$.). Thus, applying the triangle inequality, for any fixed choice of facility $\phi(\kappa)$ we have

$$d_{\phi(\kappa)\nu} \le d_{\phi(\kappa)\kappa} + d_{\mu\kappa} + d_{\mu\nu} \le d_{\phi(\kappa)\kappa} + \alpha^*_\kappa + \alpha^*_\nu.$$

Therefore the expected distance from ν to its facility $\phi(\kappa)$ is

$$\begin{aligned}\mathbb{E}[d_{\phi(\kappa)\nu}] &\le C^{\text{avg}}_\kappa + \alpha^*_\kappa + \alpha^*_\nu \\ &\le C^{\text{avg}}_\nu + \alpha^*_\nu + \alpha^*_\nu = C^{\text{avg}}_\nu + 2\alpha^*_\nu,\end{aligned}$$

where the second inequality follows from Property (PD.3(b)). From the definition of C^{avg}_ν and Property (PS.2), for any $j \in \mathbb{C}$ we have

$$\begin{aligned}\sum_{\nu\in j} C^{\text{avg}}_\nu &= \sum_{\nu\in j}\sum_{\mu\in\overline{\mathbb{F}}} d_{\mu\nu}\bar{x}_{\mu\nu} \\ &= \sum_{i\in\mathbb{F}} d_{ij} \sum_{\nu\in j}\sum_{\mu\in i} \bar{x}_{\mu\nu} \\ &= \sum_{i\in\mathbb{F}} d_{ij}x^*_{ij} = C^*_j.\end{aligned}$$

Thus, summing over all demands, the expected total connection cost is

$$\begin{aligned}\mathbb{E}[C_{\text{EGUP}}] &\le \textstyle\sum_{j\in\mathbb{C}}\sum_{\nu\in j}(C^{\text{avg}}_\nu + 2\alpha^*_\nu) \\ &= \textstyle\sum_{j\in\mathbb{C}}(C^*_j + 2r_j\alpha^*_j) = C^* + 2\cdot \text{LP}^*,\end{aligned}$$

completing the proof of the lemma. ■

Theorem 12 *Algorithm* EGUP *is a* 3*-approximation algorithm.*

Proof. By Property (SI.2), different demands from the same client are assigned to different primary demands, and by (PD.1) each primary demand opens a different facility. This ensures that our solution is feasible, namely each client j is connected to r_j different facilities (some possibly located on the same site). As for the total cost, Lemma 10 and Lemma 11 imply that the total cost is at most $F^* + C^* + 2\cdot \text{LP}^* = 3\cdot \text{LP}^* \le 3\cdot \text{OPT}$. ■

5.2 Algorithm ECHS with Ratio 1.736

In this section we improve the approximation ratio to $1 + 2/e \approx 1.736$. The improvement comes from a slightly modified rounding process and refined analysis. Note that the facility opening cost of Algorithm EGUP does not exceed that of the fractional optimum solution, while the connection cost could be far from the optimum, since we connect a non-primary demand to a facility in the neighborhood of its assigned primary demand and then estimate the distance using the triangle inequality. The basic idea to improve the estimate of the connection cost, following the approach of Chudak and Shmoys [15], is to connect each non-primary demand to its nearest neighbor when one is available, and to only use the facility opened by its assigned primary demand when none of its neighbors is open.

Algorithm ECHS**.** As before, the algorithm starts by solving the linear program and applying the adaptive partitioning algorithm described in Section 4.4 to obtain a partitioned solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$. Then we apply the rounding process to compute an integral solution (see Pseudocode 3).

We start, as before, by opening exactly one facility $\phi(\kappa)$ in the neighborhood of each primary demand κ (Line 2). For any non-primary demand ν assigned to κ, we refer to $\phi(\kappa)$ as the *target* facility of ν. In Algorithm EGUP, ν was connected to $\phi(\kappa)$, but in Algorithm ECHS we may be able to find an open facility in ν's neighborhood and connect ν to this facility. Specifically, the two changes in the algorithm are as follows:

(1) Each facility μ that is not in the neighborhood of any primary demand is opened,

independently, with probability $\bar{y}_\mu$ (Lines 4–5). Notice that if $\bar{y}_\mu > 0$ then, due to completeness of the partitioned fractional solution, we have $\bar{y}_\mu = \bar{x}_{\mu\nu}$ for some demand ν. This implies that $\bar{y}_\mu \leq 1$, because $\bar{x}_{\mu\nu} \leq 1$, by (PS.1).

(2) When connecting demands to facilities, a primary demand κ is connected to the only facility $\phi(\kappa)$ opened in its neighborhood, as before (Line 3). For a non-primary demand ν, if its neighborhood $\overline{N}(\nu)$ has an open facility, we connect ν to the closest open facility in $\overline{N}(\nu)$ (Line 8). Otherwise, we connect ν to its target facility (Line 10).

Pseudocode 3 Algorithm ECHS: Constructing Integral Solution

1: **for** each $\kappa \in P$ **do**
2:     choose one $\phi(\kappa) \in \overline{N}(\kappa)$, with each $\mu \in \overline{N}(\kappa)$ chosen as $\phi(\kappa)$ with probability $\bar{y}_\mu$
3:     open $\phi(\kappa)$ and connect κ to $\phi(\kappa)$
4: **for** each $\mu \in \overline{\mathbb{F}} - \bigcup_{\kappa \in P} \overline{N}(\kappa)$ **do**
5:     open μ with probability $\bar{y}_\mu$ (independently)
6: **for** each non-primary demand $\nu \in \overline{\mathbb{C}}$ **do**
7:     **if** any facility in $\overline{N}(\nu)$ is open **then**
8:         connect ν to the nearest open facility in $\overline{N}(\nu)$
9:     **else**
10:         connect ν to $\phi(\kappa)$ where κ is ν's assigned primary demand

Analysis. We shall first argue that the integral solution thus constructed is feasible, and then we bound the total cost of the solution. Regarding feasibility, the only constraint that is not explicitly enforced by the algorithm is the fault-tolerance requirement; namely that each client j is connected to r_j different facilities. Let ν and ν' be two different sibling demands of client j and let their assigned primary demands be κ and κ' respectively. Due to (SI.2) we know $\kappa \neq \kappa'$. From (SI.1) we have $\overline{N}(\nu) \cap \overline{N}(\nu') = \emptyset$. From (SI.2), we have $\overline{N}(\nu) \cap \overline{N}(\kappa') = \emptyset$ and $\overline{N}(\nu') \cap \overline{N}(\kappa) = \emptyset$. From (PD.1) we have $\overline{N}(\kappa) \cap \overline{N}(\kappa') = \emptyset$. It follows that $(\overline{N}(\nu) \cup \overline{N}(\kappa)) \cap (\overline{N}(\nu') \cup \overline{N}(\kappa')) = \emptyset$. Since the algorithm connects ν to some facility in $\overline{N}(\nu) \cup \overline{N}(\kappa)$ and ν' to some facility in $\overline{N}(\nu') \cup \overline{N}(\kappa')$, ν and ν' will be connected to different facilities.

We now show that the expected cost of the computed solution is bounded by $(1 + 2/e) \cdot \mathrm{LP}^*$. By (PD.1), every facility may appear in at most one primary demand's neighborhood, and the facilities open in Line 4–5 of Pseudocode 3 do not appear in any primary demand's neighborhood. Therefore, by linearity of expectation, the expected facility cost of Algorithm ECHS is

$$\mathbb{E}[F_{\mathrm{ECHS}}] = \sum_{\mu \in \overline{\mathbb{F}}} f_\mu \bar{y}_\mu = \sum_{i \in \mathbb{F}} f_i \sum_{\mu \in i} \bar{y}_\mu = \sum_{i \in \mathbb{F}} f_i y_i^* = F^*,$$

where the third equality follows from (PS.3).

To bound the connection cost, we adapt an argument of Chudak and Shmoys [15]. Consider a demand ν and denote by C_ν the random variable representing the connection cost for ν. Our goal now is to estimate $\mathbb{E}[C_\nu]$, the expected value of C_ν. Demand ν can either get connected directly to some facility in $\overline{N}(\nu)$ or indirectly to its target facility $\phi(\kappa) \in \overline{N}(\kappa)$, where κ is the primary demand to which ν is assigned. We will analyze these two cases separately.

In our analysis, in this section and the next one, we will use notation

$$D(A, \sigma) = \sum_{\mu \in A} d_{\mu\sigma} \bar{y}_\mu / \sum_{\mu \in A} \bar{y}_\mu$$

for the average distance between a demand σ and a set A of facilities. Note that, in particular, we have $C_\nu^{\mathrm{avg}} = D(\overline{N}(\nu), \nu)$.

We first estimate the expected cost $d_{\phi(\kappa)\nu}$ of the indirect connection. Let Λ^ν denote the event that some facility in $\overline{N}(\nu)$ is opened. Then

$$\mathbb{E}[C_\nu \mid \neg\Lambda^\nu] = \mathbb{E}[d_{\phi(\kappa)\nu} \mid \neg\Lambda^\nu] = D(\overline{N}(\kappa) \setminus \overline{N}(\nu), \nu). \tag{5.1}$$

Note that $\neg\Lambda^\nu$ implies that $\overline{N}(\kappa) \setminus \overline{N}(\nu) \neq \emptyset$, since $\overline{N}(\kappa)$ contains exactly one open facility, namely $\phi(\kappa)$.

Lemma 13 *Let ν be a demand assigned to a primary demand κ, and assume that $\overline{N}(\kappa) \setminus \overline{N}(\nu) \neq \emptyset$. Then*

$$\mathbb{E}[C_\nu \mid \neg\Lambda^\nu] \leq C_\nu^{\text{avg}} + 2\alpha_\nu^*.$$

Proof. By (5.1), we need to show that $D(\overline{N}(\kappa) \setminus \overline{N}(\nu), \nu) \leq C_\nu^{\text{avg}} + 2\alpha_\nu^*$. There are two cases to consider.

Case 1: There exists some $\mu' \in \overline{N}(\kappa) \cap \overline{N}(\nu)$ such that $d_{\mu'\kappa} \leq C_\kappa^{\text{avg}}$. In this case, for every $\mu \in \overline{N}(\kappa) \setminus \overline{N}(\nu)$, we have

$$d_{\mu\nu} \leq d_{\mu\kappa} + d_{\mu'\kappa} + d_{\mu'\nu} \leq \alpha_\kappa^* + C_\kappa^{\text{avg}} + \alpha_\nu^* \leq C_\nu^{\text{avg}} + 2\alpha_\nu^*,$$

using the triangle inequality, complementary slackness, and (PD.3(b)). By summing over all $\mu \in \overline{N}(\kappa) \setminus \overline{N}(\nu)$, it follows that $D(\overline{N}(\kappa) \setminus \overline{N}(\nu), \nu) \leq C_\nu^{\text{avg}} + 2\alpha_\nu^*$.

Case 2: Every $\mu' \in \overline{N}(\kappa) \cap \overline{N}(\nu)$ has $d_{\mu'\kappa} > C_\kappa^{\text{avg}}$. Since $C_\kappa^{\text{avg}} = D(\overline{N}(\kappa), \kappa)$, this implies that $D(\overline{N}(\kappa) \setminus \overline{N}(\nu), \kappa) \leq C_\kappa^{\text{avg}}$. Therefore, choosing an arbitrary $\mu' \in \overline{N}(\kappa) \cap \overline{N}(\nu)$, we obtain

$$D(\overline{N}(\kappa) \setminus \overline{N}(\nu), \nu) \leq D(\overline{N}(\kappa) \setminus \overline{N}(\nu), \kappa) + d_{\mu'\kappa} + d_{\mu'\nu} \leq C_\kappa^{\text{avg}} + \alpha_\kappa^* + \alpha_\nu^* \leq C_\nu^{\text{avg}} + 2\alpha_\nu^*,$$

where we again use the triangle inequality, complementary slackness, and (PD.3(b)).

Since the lemma holds in both cases, the proof is now complete. ■

We now continue our estimation of the connection cost. The next step of our analysis is to show that

$$\mathbb{E}[C_\nu] \leq C_\nu^{\text{avg}} + \frac{2}{e}\alpha_\nu^*. \tag{5.2}$$

The argument is divided into three cases. The first, easy case is when ν is a primary demand κ. According to the algorithm (see Pseudocode 3, Line 2), we have $C_\kappa = d_{\mu\kappa}$ with probability $\bar{y}_\mu$, for $\mu \in \overline{N}(\kappa)$. Therefore $\mathbb{E}[C_\kappa] = C_\kappa^{\text{avg}}$, so (5.2) holds.

Next, we consider a non-primary demand ν. Let κ be the primary demand that ν is assigned to. We first deal with the sub-case when $\overline{N}(\kappa) \setminus \overline{N}(\nu) = \emptyset$, which is the same as $\overline{N}(\kappa) \subseteq \overline{N}(\nu)$. Property (CO) implies that $\bar{x}_{\mu\nu} = \bar{y}_\mu = \bar{x}_{\mu\kappa}$ for every $\mu \in \overline{N}(\kappa)$, so we have $\sum_{\mu\in\overline{N}(\kappa)} \bar{x}_{\mu\nu} = \sum_{\mu\in\overline{N}(\kappa)} \bar{x}_{\mu\kappa} = 1$, due to (PS.1). On the other hand, we have $\sum_{\mu\in\overline{N}(\nu)} \bar{x}_{\mu\nu} = 1$, and $\bar{x}_{\mu\nu} > 0$ for all $\mu \in \overline{N}(\nu)$. Therefore $\overline{N}(\kappa) = \overline{N}(\nu)$ and C_ν has exactly the same distribution as C_κ. So this case reduces to the first case, namely we have $\mathbb{E}[C_\nu] = C_\nu^{\text{avg}}$, and (5.2) holds.

The last, and only non-trivial case is when $\overline{N}(\kappa) \setminus \overline{N}(\nu) \neq \emptyset$. We handle this case in the following lemma.

Lemma 14 *Assume that $\overline{N}(\kappa) \setminus \overline{N}(\nu) \neq \emptyset$. Then the expected connection cost of ν, conditioned on the event that at least one of its neighbor opens, satisfies*

$$\mathbb{E}[C_\nu \mid \Lambda^\nu] \leq C_\nu^{\text{avg}}.$$

Proof. The proof is similar to an analogous result in [15, 7]. For the sake of completeness we sketch here a simplified argument, adapted to our terminology and notation. The idea is to consider a different random process that is easier to analyze and whose expected connection cost is not better than that in the algorithm.

We partition $\overline{N}(\nu)$ into groups $G_1, ..., G_k$, where two different facilities μ and μ' are put in the same G_s, where $s \in \{1, \ldots, k\}$, if they both belong to the same set $\overline{N}(\kappa)$ for some primary demand κ. If some μ is not a neighbor of any primary demand, then it constitutes a singleton group. For each s, let $\bar{d}_s = D(G_s, \nu)$ be the average distance from ν to G_s. Assume that $G_1, ..., G_k$ are ordered by non-decreasing average distance to ν, that is $\bar{d}_1 \leq \bar{d}_2 \leq ... \leq \bar{d}_k$. For each group G_s, we select it, independently, with probability $g_s = \sum_{\mu \in G_s} \bar{y}_\mu$. For each selected group G_s, we open exactly one facility in G_s, where each $\mu \in G_s$ is opened with probability $\bar{y}_\mu / \sum_{\eta \in G_s} \bar{y}_\eta$.

So far, this process is the same as that in the algorithm (if restricted to $\overline{N}(\nu)$). However, we connect ν in a slightly different way, by choosing the smallest s for which G_s was selected and connecting ν to the open facility in G_s. This can only increase our expected connection cost, assuming that at least one facility in $\overline{N}(\nu)$ opens, so

$$
\begin{aligned}
\mathbb{E}[C_\nu \mid \Lambda^\nu] &\leq \frac{1}{\mathbb{P}[\Lambda^\nu]} \left(\bar{d}_1 g_1 + \bar{d}_2 g_2 (1-g_1) + \ldots + \bar{d}_k g_k (1-g_1)(1-g_2)\ldots(1-g_k) \right) \\
&\leq \frac{1}{\mathbb{P}[\Lambda^\nu]} \cdot \sum_{s=1}^{k} \bar{d}_s g_s \cdot \left(\sum_{t=1}^{k} g_t \prod_{z=1}^{t-1} (1-g_z) \right) && (5.3) \\
&= \sum_{s=1}^{k} \bar{d}_s g_s && (5.4) \\
&= C_\nu^{\text{avg}}. && (5.5)
\end{aligned}
$$

The proof for inequality (5.3) is given in A.2 (note that $\sum_{s=1}^{k} g_s = 1$), equality (5.4) follows from $\mathbb{P}[\Lambda^\nu] = 1 - \prod_{t=1}^{k}(1-g_t) = \sum_{t=1}^{k} g_t \prod_{z=1}^{t-1}(1-g_z)$, and (5.5) follows from the definition of the distances $\bar{d}_s$, probabilities g_s, and simple algebra. ■

Next, we show an estimate on the probability that none of ν's neighbors is opened by the algorithm.

Lemma 15 *The probability that none of* ν*'s neighbors is opened satisfies* $\mathbb{P}[\neg\Lambda^{\nu}] \leq 1/e$.

Proof. We use the same partition of $\overline{N}(\nu)$ into groups $G_1, ..., G_k$ as in the proof of Lemma 14. Denoting by g_s the probability that a group G_s is selected (and thus that it has an open facility), we have

$$\mathbb{P}[\neg\Lambda^{\nu}] = \prod_{s=1}^{k}(1 - g_s) \leq e^{-\sum_{s=1}^{k} g_s} = e^{-\sum_{\mu\in\overline{N}(\nu)} \bar{y}_{\mu}} = \frac{1}{e}.$$

In this derivation, we first use that $1 - x \leq e^{-x}$ holds for all x, the second equality follows from $\sum_{s=1}^{k} g_s = \sum_{\mu\in\overline{N}(\nu)} \bar{y}_{\mu}$ and the last equality follows from $\sum_{\mu\in\overline{N}(\nu)} \bar{y}_{\mu} = 1$. ■

We are now ready to estimate the unconditional expected connection cost of ν (in the case when $\overline{N}(\kappa) \setminus \overline{N}(\nu) \neq \emptyset$) as follows,

$$\begin{aligned}
\mathbb{E}[C_{\nu}] &= \mathbb{E}[C_{\nu} \mid \Lambda^{\nu}] \cdot \mathbb{P}[\Lambda^{\nu}] + \mathbb{E}[C_{\nu} \mid \neg\Lambda^{\nu}] \cdot \mathbb{P}[\neg\Lambda^{\nu}] \\
&\leq C_{\nu}^{\text{avg}} \cdot \mathbb{P}[\Lambda^{\nu}] + (C_{\nu}^{\text{avg}} + 2\alpha_{\nu}^{*}) \cdot \mathbb{P}[\neg\Lambda^{\nu}] \qquad (5.6) \\
&= C_{\nu}^{\text{avg}} + 2\alpha_{\nu}^{*} \cdot \mathbb{P}[\neg\Lambda^{\nu}] \\
&\leq C_{\nu}^{\text{avg}} + \frac{2}{e} \cdot \alpha_{\nu}^{*}. \qquad (5.7)
\end{aligned}$$

In the above derivation, inequality (5.6) follows from Lemmas 13 and 14, and inequality (5.7) follows from Lemma 15.

We have thus shown that the bound (5.2) holds in all three cases. Summing over all demands ν of a client j, we can now bound the expected connection cost of client j:

$$\mathbb{E}[C_j] = \textstyle\sum_{\nu\in j} \mathbb{E}[C_{\nu}] \leq \sum_{\nu\in j}(C_{\nu}^{\text{avg}} + \frac{2}{e} \cdot \alpha_{\nu}^{*}) = C_j^{*} + \frac{2}{e} \cdot r_j \alpha_j^{*}.$$

Finally, summing over all clients j, we obtain our bound on the expected connection cost,

$$\mathbb{E}[C_{\mathrm{ECHS}}] \leq C^* + \frac{2}{e} \cdot \mathrm{LP}^*.$$

Therefore we have established that our algorithm constructs a feasible integral solution with an overall expected cost

$$\mathbb{E}[F_{\mathrm{ECHS}} + C_{\mathrm{ECHS}}] \leq F^* + C^* + \frac{2}{e} \cdot \mathrm{LP}^* = (1 + 2/e) \cdot \mathrm{LP}^* \leq (1 + 2/e) \cdot \mathrm{OPT}.$$

Summarizing, we obtain the main result of this section.

Theorem 16 *Algorithm* ECHS *is a* $(1 + 2/e)$*-approximation algorithm for* FTFP*.*

5.3 Algorithm EBGS with Ratio 1.575

In this section we give our main result, a 1.575-approximation algorithm for FTFP, where 1.575 is the value of $\min_{\gamma \geq 1} \max\{\gamma, 1 + 2/e^{\gamma}, \frac{1/e + 1/e^{\gamma}}{1 - 1/\gamma}\}$, rounded to three decimal digits. This matches the ratio of the best known LP-rounding algorithm for UFL by Byrka *et al.* [8].

Recall that in Section 5.2 we showed how to compute an integral solution with facility cost bounded by F^* and connection cost bounded by $C^* + 2/e \cdot \mathrm{LP}^*$. Thus, while our facility cost does not exceed the optimal fractional facility cost, our connection cost is significantly larger than the connection cost in the optimal fractional solution. A natural idea is to balance these two ratios by reducing the connection cost at the expense of the facility cost. One way to do this would be to increase the probability of opening facilities, from $\bar{y}_\mu$ (used in Algorithm ECHS) to, say, $\gamma \bar{y}_\mu$, for some $\gamma > 1$. This increases the expected facility cost by a factor of γ but, as it turns out, it also reduces the probability that an indirect connection occurs for a non-primary demand to $1/e^{\gamma}$ (from the previous value $1/e$ in ECHS). As a consequence, for each primary demand κ, the new algorithm will select a facility to open from the nearest facilities μ in $\overline{N}(\kappa)$ such that the connection values $\bar{x}_{\mu\nu}$ sum up to $1/\gamma$, instead of 1 as in Algorithm ECHS. It is easily seen that this will improve the estimate on connection cost for primary demands. These two changes, along with a more refined analysis, are the essence of the approach in [8], expressed in our terminology.

Our approach can be thought of as a combination of the above ideas with the techniques of demand reduction and adaptive partitioning that we introduced earlier. However, our adaptive partitioning technique needs to be carefully modified, because now we will be using a more intricate neighborhood structure, with the neighborhood of each demand divided into two disjoint parts, and with restrictions on how parts from different demands can overlap.

We begin by describing properties that our partitioned fractional solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ needs to satisfy. Assume that γ is some constant such that $1 < \gamma < 2$. As mentioned earlier, the neighborhood $\overline{N}(\nu)$ of each demand ν will be divided into two disjoint parts. The first part, called the *close neighborhood* and denoted $\overline{N}_{\mathrm{cls}}(\nu)$, contains the facilities in $\overline{N}(\nu)$ nearest to ν with the total connection value equal $1/\gamma$, that is $\sum_{\mu \in \overline{N}_{\mathrm{cls}}(\nu)} \bar{x}_{\mu\nu} = 1/\gamma$. The second part, called the *far neighborhood* and denoted $\overline{N}_{\mathrm{far}}(\nu)$, contains the remaining facilities in $\overline{N}(\nu)$ (so $\sum_{\mu \in \overline{N}_{\mathrm{far}}(\nu)} \bar{x}_{\mu\nu} = 1 - 1/\gamma$). We restate these definitions formally below in Property (NB). Recall that for any set A of facilities and a demand ν, by $D(A, \nu)$ we denote the average distance between ν and the facilities in A, that is $D(A, \nu) = \sum_{\mu \in A} d_{\mu\nu} \bar{y}_{\mu} / \sum_{\mu \in A} \bar{y}_{\mu}$.

We will use notations $C^{\text{avg}}_{\text{cls}}(\nu) = D(\overline{N}_{\text{cls}}(\nu), \nu)$ and $C^{\text{avg}}_{\text{far}}(\nu) = D(\overline{N}_{\text{far}}(\nu), \nu)$ for the average distances from ν to its close and far neighborhoods, respectively. By the definition of these sets and the completeness property (CO), these distances can be expressed as

$$C^{\text{avg}}_{\text{cls}}(\nu) = \gamma \sum_{\mu \in \overline{N}_{\text{cls}}(\nu)} d_{\mu\nu}\bar{x}_{\mu\nu} \quad \text{and} \quad C^{\text{avg}}_{\text{far}}(\nu) = \frac{\gamma}{\gamma - 1} \sum_{\mu \in \overline{N}_{\text{far}}(\nu)} d_{\mu\nu}\bar{x}_{\mu\nu}.$$

We will also use notation $C^{\max}_{\text{cls}}(\nu) = \max_{\mu \in \overline{N}_{\text{cls}}(\nu)} d_{\mu\nu}$ for the maximum distance from ν to its close neighborhood. The average distance from a demand ν to its overall neighborhood $\overline{N}(\nu)$ is denoted as $C^{\text{avg}}(\nu) = D(\overline{N}(\nu), \nu) = \sum_{\mu \in \overline{N}(\nu)} d_{\mu\nu}\bar{x}_{\mu\nu}$. It is easy to see that

$$C^{\text{avg}}(\nu) = \frac{1}{\gamma} C^{\text{avg}}_{\text{cls}}(\nu) + \frac{\gamma - 1}{\gamma} C^{\text{avg}}_{\text{far}}(\nu). \tag{5.8}$$

Our partitioned solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ must satisfy the same partitioning and completeness properties as before, namely properties (PS) and (CO) in Section 4.4. In addition, it must satisfy a new neighborhood property (NB) and modified properties (PD') and (SI'), listed below.

(NB) *Neighborhoods.* For each demand $\nu \in \overline{\mathbb{C}}$, its neighborhood is divided into *close* and *far* neighborhood, that is $\overline{N}(\nu) = \overline{N}_{\text{cls}}(\nu) \cup \overline{N}_{\text{far}}(\nu)$, where

- $\overline{N}_{\text{cls}}(\nu) \cap \overline{N}_{\text{far}}(\nu) = \emptyset$,
- $\sum_{\mu \in \overline{N}_{\text{cls}}(\nu)} \bar{x}_{\mu\nu} = 1/\gamma$, and
- if $\mu \in \overline{N}_{\text{cls}}(\nu)$ and $\mu' \in \overline{N}_{\text{far}}(\nu)$ then $d_{\mu\nu} \le d_{\mu'\nu}$.

Note that the first two conditions, together with (PS.1), imply that $\sum_{\mu \in \overline{N}_{\text{far}}(\nu)} \bar{x}_{\mu\nu} = 1 - 1/\gamma$. When defining $\overline{N}_{\text{cls}}(\nu)$, in case of ties, which can occur when some facilities in $\overline{N}(\nu)$ are at the same distance from ν, we use a tie-breaking rule that is explained in the proof of Lemma 17 (the only place where the rule is needed).

(PD') *Primary demands.* Primary demands satisfy the following conditions:

1. For any two different primary demands $\kappa, \kappa' \in P$ we have $\overline{N}_{\mathrm{cls}}(\kappa) \cap \overline{N}_{\mathrm{cls}}(\kappa') = \emptyset$.
2. For each site $i \in \mathbb{F}$, $\sum_{\kappa \in P} \sum_{\mu \in i \cap \overline{N}_{\mathrm{cls}}(\kappa)} \bar{x}_{\mu\kappa} \leq y_i^*$. In the summation, as before, we overload notation i to stand for the set of facilities created on site i.
3. Each demand $\nu \in \overline{\mathbb{C}}$ is assigned to one primary demand $\kappa \in P$ such that
   (a) $\overline{N}_{\mathrm{cls}}(\nu) \cap \overline{N}_{\mathrm{cls}}(\kappa) \neq \emptyset$, and
   (b) $C^{\mathrm{avg}}_{\mathrm{cls}}(\nu) + C^{\mathrm{max}}_{\mathrm{cls}}(\nu) \geq C^{\mathrm{avg}}_{\mathrm{cls}}(\kappa) + C^{\mathrm{max}}_{\mathrm{cls}}(\kappa)$.

(SI') *Siblings.* For any pair $\nu, \nu' \in \overline{\mathbb{C}}$ of different siblings we have

1. $\overline{N}(\nu) \cap \overline{N}(\nu') = \emptyset$.
2. If ν is assigned to a primary demand κ then $\overline{N}(\nu') \cap \overline{N}_{\mathrm{cls}}(\kappa) = \emptyset$. In particular, by Property (PD'.3(a)), this implies that different sibling demands are assigned to different primary demands, since $\overline{N}_{\mathrm{cls}}(\nu')$ is a subset of $\overline{N}(\nu')$.

Modified adaptive partitioning. To obtain a fractional solution with the above properties, we employ a modified adaptive partitioning algorithm. As in Section 4.4, we have two phases. In Phase 1 we split clients into demands and create facilities on sites, while in Phase 2 we augment each demand's connection values $\bar{x}_{\mu\nu}$ so that the total connection value of each demand ν is 1. As the partitioning algorithm proceeds, for any demand ν, $\overline{N}(\nu)$ denotes the set of facilities with $\bar{x}_{\mu\nu} > 0$; hence the notation $\overline{N}(\nu)$ actually represents a dynamic set which gets fixed once the partitioning algorithm concludes both Phase 2. On the other hand, $\overline{N}_{\mathrm{cls}}(\nu)$ and $\overline{N}_{\mathrm{far}}(\nu)$ refer to the close and far neighborhoods at the time when $\overline{N}(\nu)$ is fixed.

Similar to the algorithm in Section 4.4, Phase 1 runs in iterations. Fix some iteration and consider any client j. As before, $\widetilde{N}(j)$ is the neighborhood of j with respect to the yet unpartitioned solution, namely the set of facilities μ such that $\widetilde{x}_{\mu j} > 0$. Order the facilities in this set as $\widetilde{N}(j) = \{\mu_1, ..., \mu_q\}$ with non-decreasing distance from j, that is $d_{\mu_1 j} \leq d_{\mu_2 j} \leq \ldots \leq d_{\mu_q j}$. Without loss of generality, there is an index l for which $\sum_{s=1}^{l} \widetilde{x}_{\mu_s j} = 1/\gamma$, since we can always split one facility to achieve this. Then we define $\widetilde{N}_{\mathrm{cls}}(j) = \{\mu_1, ..., \mu_l\}$. (Unlike close neighborhoods of demands, $\widetilde{N}_{\mathrm{cls}}(j)$ can vary over time.) We also use notation

$$\mathrm{tcc}_{\mathrm{cls}}(j) = D(\widetilde{N}_{\mathrm{cls}}(j), j) = \gamma \sum_{\mu \in \widetilde{N}_{\mathrm{cls}}(j)} d_{\mu j} \widetilde{x}_{\mu j} \quad \text{and} \quad \mathrm{dmax}_{\mathrm{cls}}(j) = \max_{\mu \in \widetilde{N}_{\mathrm{cls}}(j)} d_{\mu j}.$$

When the iteration starts, we first find a not-yet-exhausted client p that minimizes the value of $\mathrm{tcc}_{\mathrm{cls}}(p) + \mathrm{dmax}_{\mathrm{cls}}(p)$ and create a new demand ν for p. Now we have two cases:

Case 1: $\widetilde{N}_{\mathrm{cls}}(p) \cap \overline{N}(\kappa) \neq \emptyset$ for some existing primary demand $\kappa \in P$. In this case we assign ν to κ. As before, if there are multiple such κ, we pick any of them. We also fix $\bar{x}_{\mu\nu} \leftarrow \widetilde{x}_{\mu p}$ and $\widetilde{x}_{\mu p} \leftarrow 0$ for each $\mu \in \widetilde{N}(p) \cap \overline{N}(\kappa)$. Note that although we check for overlap between $\widetilde{N}_{\mathrm{cls}}(p)$ and $\overline{N}(\kappa)$, the facilities we actually move into $\overline{N}(\nu)$ include all facilities in the intersection of $\widetilde{N}(p)$, a bigger set, with $\overline{N}(\kappa)$.

At this time, the total connection value between ν and $\mu \in \overline{N}(\nu)$ is at most $1/\gamma$, since $\sum_{\mu \in \overline{N}(\kappa)} \bar{y}_{\mu} = 1/\gamma$ (this follows from the definition of neighborhoods for new primary demands in Case 2 below) and we have $\overline{N}(\nu) \subseteq \overline{N}(\kappa)$ at this point. Later in Phase 2 we will add additional facilities from $\widetilde{N}(p)$ to $\overline{N}(\nu)$ to make ν's total connection value equal to 1.

Case 2: $\widetilde{N}_{\mathrm{cls}}(p) \cap \overline{N}(\kappa) = \emptyset$ for all existing primary demands $\kappa \in P$. In this case we make ν a primary demand (that is, add it to P) and assign it to itself. We then move the facilities from $\widetilde{N}_{\mathrm{cls}}(p)$ to $\overline{N}(\nu)$, that is for $\mu \in \widetilde{N}_{\mathrm{cls}}(p)$ we set $\bar{x}_{\mu\nu} \leftarrow \widetilde{x}_{\mu p}$ and $\widetilde{x}_{\mu p} \leftarrow 0$.

It is easy to see that the total connection value of ν to $\overline{N}(\nu)$ is now exactly $1/\gamma$, that is $\sum_{\mu \in \overline{N}(\nu)} \bar{y}_\mu = 1/\gamma$. Moreover, facilities remaining in $\widetilde{N}(p)$ are all farther away from ν than those in $\overline{N}(\nu)$. As we add only facilities from $\widetilde{N}(p)$ to $\overline{N}(\nu)$ in Phase 2, the final $\overline{N}_{\mathrm{cls}}(\nu)$ contains the same set of facilities as the current set $\overline{N}(\nu)$. (More precisely, $\overline{N}_{\mathrm{cls}}(\nu)$ consists of the facilities that either are currently in $\overline{N}(\nu)$ or were obtained from splitting the facilities currently in $\overline{N}(\nu)$.)

Once all clients are exhausted, that is, each client j has r_j demands created, Phase 1 concludes. We then run Phase 2, the augmenting phase, following the same steps as in Section 4.4. For each client j and each demand $\nu \in j$ with total connection value to $\overline{N}(\nu)$ less than 1 (that is, $\sum_{\mu \in \overline{N}(\nu)} \bar{x}_{\mu\nu} < 1$), we use our AugmentToUnit() procedure to add additional facilities (possibly split, if necessary) from $\widetilde{N}(j)$ to $\overline{N}(\nu)$ to make the total connection value between ν and $\overline{N}(\nu)$ equal 1.

This completes the description of the partitioning algorithm. Summarizing, for each client $j \in \mathbb{C}$ we created r_j demands on the same point as j, and we created a number of facilities at each site $i \in \mathbb{F}$. Thus computed sets of demands and facilities are denoted

$\overline{\mathbb{C}}$ and $\overline{\mathbb{F}}$, respectively. For each facility $\mu \in i$ we defined its fractional opening value $\bar{y}_\mu$, $0 \leq \bar{y}_\mu \leq 1$, and for each demand $\nu \in j$ we defined its fractional connection value $\bar{x}_{\mu\nu} \in \{0, \bar{y}_\mu\}$. The connections with $\bar{x}_{\mu\nu} > 0$ define the neighborhood $\overline{N}(\nu)$. The facilities in $\overline{N}(\nu)$ that are closest to ν and have total connection value from ν equal $1/\gamma$ form the close neighborhood $\overline{N}_{\mathrm{cls}}(\nu)$, while the remaining facilities in $\overline{N}(\nu)$ form the far neighborhood $\overline{N}_{\mathrm{far}}(\nu)$. It remains to show that this partitioning satisfies all the desired properties.

Correctness of partitioning. We now argue that our partitioned fractional solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ satisfies all the stated properties. Properties (PS), (CO) and (NB) are directly enforced by the algorithm.

(PD'.1) holds because for each primary demand $\kappa \in p$, $\overline{N}_{\mathrm{cls}}(\kappa)$ is the same set as $\widetilde{N}_{\mathrm{cls}}(p)$ at the time when κ was created, and $\widetilde{N}_{\mathrm{cls}}(p)$ is removed from $\widetilde{N}(p)$ right after this step. Further, the partitioning algorithm makes κ a primary demand only if $\widetilde{N}_{\mathrm{cls}}(p)$ is disjoint from the set $\overline{N}(\kappa')$ of all existing primary demands κ' at that iteration, but these neighborhoods are the same as the final close neighborhoods $\overline{N}_{\mathrm{cls}}(\kappa')$.

The justification of (PD'.2) is similar to that for (PD.2) from Section 4.4. All close neighborhoods of primary demands are disjoint, due to (PD'.1), so each facility $\mu \in i$ can appear in at most one $\overline{N}_{\mathrm{cls}}(\kappa)$, for some $\kappa \in P$. Condition (CO) implies that $\bar{y}_\mu = \bar{x}_{\mu\kappa}$ for $\mu \in \overline{N}_{\mathrm{cls}}(\kappa)$. As a result, the summation on the left-hand side is not larger than $\sum_{\mu \in i} \bar{y}_\mu = y_i^*$.

Regarding (PD'.3(a)), at first glance this property seems to follow directly from the algorithm, as we only assign a demand ν to a primary demand κ when $\overline{N}(\nu)$ at that iteration overlaps with $\overline{N}(\kappa)$ (which is equal to the final value of $\overline{N}_{\mathrm{cls}}(\kappa)$). However, it is a little more subtle, as the final $\overline{N}_{\mathrm{cls}}(\nu)$ may contain facilities added to $\overline{N}(\nu)$ in Phase 2. Those facilities may turn out to be closer to ν than some facilities in $\overline{N}(\kappa) \cap \widetilde{N}(j)$ (not $\widetilde{N}_{\mathrm{cls}}(j)$) that we added to $\overline{N}(\nu)$ in Phase 1. If the final $\overline{N}_{\mathrm{cls}}(\nu)$ consists only of facilities added in Phase 2, we no longer have the desired overlap of $\overline{N}_{\mathrm{cls}}(\kappa)$ and $\overline{N}_{\mathrm{cls}}(\nu)$. Luckily this bad scenario never occurs. We postpone the proof of this property to Lemma 17. The proof of (PD'.3(b)) is similar to that of Lemma 9, and we defer it to Lemma 18.

(SI'.1) follows directly from the algorithm because for each demand $\nu \in j$, all facilities added to $\overline{N}(\nu)$ are immediately removed from $\widetilde{N}(j)$ and each facility is added to $\overline{N}(\nu)$ of exactly one demand $\nu \in j$. Splitting facilities obviously preserves (SI'.1).

The proof of (SI'.2) is similar to that of Lemma 7. If $\kappa = \nu$ then (SI'.2) follows from (SI'.1), so we can assume that $\kappa \neq \nu$. Suppose that $\nu' \in j$ is assigned to $\kappa' \in P$ and consider the situation after Phase 1. By the way we reassign facilities in Case 1, at this time we have $\overline{N}(\nu) \subseteq \overline{N}(\kappa) = \overline{N}_{\mathrm{cls}}(\kappa)$ and $\overline{N}(\nu') \subseteq \overline{N}(\kappa') = \overline{N}_{\mathrm{cls}}(\kappa')$, so $\overline{N}(\nu') \cap \overline{N}_{\mathrm{cls}}(\kappa) = \emptyset$, by (PD'.1). Moreover, we have $\widetilde{N}(j) \cap \overline{N}_{\mathrm{cls}}(\kappa) = \emptyset$ after this iteration, because any facilities that were also in $\overline{N}_{\mathrm{cls}}(\kappa)$ were removed from $\widetilde{N}(j)$ when ν was created. In Phase 2, augmentation does not change $\overline{N}_{\mathrm{cls}}(\kappa)$ and all facilities added to $\overline{N}(\nu')$ are from the set $\widetilde{N}(j)$ at the end of Phase 1, which is a subset of the set $\widetilde{N}(j)$ after this iteration, since $\widetilde{N}(j)$ can only shrink. So the condition (SI'.2) will remain true.

Lemma 17 *Property (PD'.3(a)) holds.*

Proof. Let j be the client for which $\nu \in j$. We consider an iteration when we create ν from j and assign it to κ, and within this proof, notation $\widetilde{N}_{\mathrm{cls}}(j)$ and $\widetilde{N}(j)$ will refer to the value of the sets at this particular time. At this time, $\overline{N}(\nu)$ is initialized to $\widetilde{N}(j) \cap \overline{N}(\kappa)$.

Recall that $\overline{N}(\kappa)$ is now equal to the final $\overline{N}_{\mathrm{cls}}(\kappa)$ (taking into account facility splitting). We would like to show that the set $\widetilde{N}_{\mathrm{cls}}(j) \cap \overline{N}_{\mathrm{cls}}(\kappa)$ (which is not empty) will be included in $\overline{N}_{\mathrm{cls}}(\nu)$ at the end. Technically speaking, this will not be true due to facility splitting, so we need to rephrase this claim and the proof in terms of the set of facilities obtained after the algorithm completes.

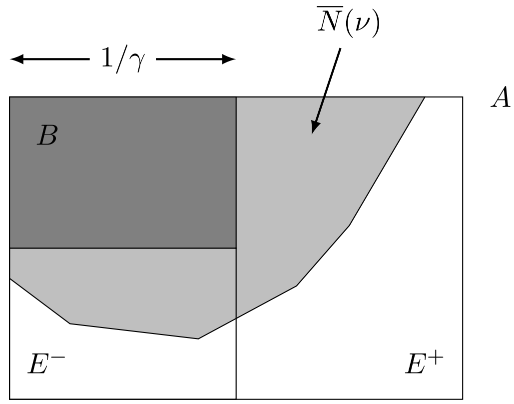


Figure 5.1: Illustration of the sets $\overline{N}(\nu)$, A, B, E^- and E^+ in the proof of Lemma 17. Let $X \Subset Y$ mean that the facility sets X is obtained from Y by splitting facilities. We then have $A \Subset \widetilde{N}(j)$, $B \Subset \widetilde{N}_{\mathrm{cls}}(j) \cap \overline{N}_{\mathrm{cls}}(\kappa)$, $E^- \Subset \widetilde{N}_{\mathrm{cls}}(j) - \overline{N}_{\mathrm{cls}}(\kappa)$, $E^+ \Subset \widetilde{N}(j) - \widetilde{N}_{\mathrm{cls}}(j)$.

We define the sets A, B, E^- and E^+ as the subsets of $\overline{\mathbb{F}}$ (the final set of facilities) that were obtained from splitting facilities in the sets $\widetilde{N}(j)$, $\widetilde{N}_{\mathrm{cls}}(j) \cap \overline{N}_{\mathrm{cls}}(\kappa)$, $\widetilde{N}_{\mathrm{cls}}(j) - \overline{N}_{\mathrm{cls}}(\kappa)$ and $\widetilde{N}(j) - \widetilde{N}_{\mathrm{cls}}(j)$, respectively. (See Figure 5.1.) We claim that at the end $B \subseteq \overline{N}_{\mathrm{cls}}(\nu)$, with the caveat that the ties in the definition of $\overline{N}_{\mathrm{cls}}(\nu)$ are broken in favor of the facilities in B. (This is the tie-breaking rule that we mentioned in the definition of $\overline{N}_{\mathrm{cls}}(\nu)$.) This will be sufficient to prove the lemma because $B \neq \emptyset$, by the algorithm.

We now prove this claim. In this paragraph $\overline{N}(\nu)$ denotes the final set $\overline{N}(\nu)$ after both phases are completed. Thus the total connection value of $\overline{N}(\nu)$ to ν is 1. Note first that

$B \subseteq \overline{N}(\nu) \subseteq A$, because we never remove facilities from $\overline{N}(\nu)$ and we only add facilities from $\widetilde{N}(j)$. Also, $B \cup E^-$ represents the facilities obtained from $\widetilde{N}_{\mathrm{cls}}(j)$, so $\sum_{\mu \in B \cup E^-} \bar{y}_\mu = 1/\gamma$. This and $B \subseteq \overline{N}(\nu)$ implies that the total connection value of $B \cup (\overline{N}(\nu) \cap E^-)$ to ν is at most $1/\gamma$. But all facilities in $B \cup (\overline{N}(\nu) \cap E^-)$ are closer to ν (taking into account our tie breaking in property (NB)) than those in $E^+ \cap \overline{N}(\nu)$. It follows that $B \subseteq \overline{N}_{\mathrm{cls}}(\nu)$, completing the proof. ■

Lemma 18 *Property (PD'.3(b)) holds.*

Proof. This proof is similar to that for Lemma 9. For a client j and demand η, we will write $\mathrm{tcc}^{\eta}_{\mathrm{cls}}(j)$ and $\mathrm{dmax}^{\eta}_{\mathrm{cls}}(j)$ to denote the values of $\mathrm{tcc}_{\mathrm{cls}}(j)$ and $\mathrm{dmax}_{\mathrm{cls}}(j)$ at the time when η was created. (Here η may or may not be a demand of client j).

Suppose $\nu \in j$ is assigned to a primary demand $\kappa \in p$. By the way primary demands are constructed in the partitioning algorithm, $\widetilde{N}_{\mathrm{cls}}(p)$ becomes $\overline{N}(\kappa)$, which is equal to the final value of $\overline{N}_{\mathrm{cls}}(\kappa)$. So we have $C^{\mathrm{avg}}_{\mathrm{cls}}(\kappa) = \mathrm{tcc}^{\kappa}_{\mathrm{cls}}(p)$ and $C^{\max}_{\mathrm{cls}}(\kappa) = \mathrm{dmax}^{\kappa}_{\mathrm{cls}}(p)$. Further, since we choose p to minimize $\mathrm{tcc}_{\mathrm{cls}}(p) + \mathrm{dmax}_{\mathrm{cls}}(p)$, we have that $\mathrm{tcc}^{\kappa}_{\mathrm{cls}}(p) + \mathrm{dmax}^{\kappa}_{\mathrm{cls}}(p) \leq \mathrm{tcc}^{\kappa}_{\mathrm{cls}}(j) + \mathrm{dmax}^{\kappa}_{\mathrm{cls}}(j)$.

Using an argument analogous to that in the proof of Lemma 8, our modified partitioning algorithm guarantees that $\mathrm{tcc}^{\kappa}_{\mathrm{cls}}(j) \leq \mathrm{tcc}^{\nu}_{\mathrm{cls}}(j) \leq C^{\mathrm{avg}}_{\mathrm{cls}}(\nu)$ and $\mathrm{dmax}^{\kappa}_{\mathrm{cls}}(j) \leq \mathrm{dmax}^{\nu}_{\mathrm{cls}}(j) \leq C^{\max}_{\mathrm{cls}}(\nu)$ since ν was created later. Therefore, we have

$$\begin{aligned} C^{\mathrm{avg}}_{\mathrm{cls}}(\kappa) + C^{\max}_{\mathrm{cls}}(\kappa) &= \mathrm{tcc}^{\kappa}_{\mathrm{cls}}(p) + \mathrm{dmax}^{\kappa}_{\mathrm{cls}}(p) \\ &\leq \mathrm{tcc}^{\kappa}_{\mathrm{cls}}(j) + \mathrm{dmax}^{\kappa}_{\mathrm{cls}}(j) \leq \mathrm{tcc}^{\nu}_{\mathrm{cls}}(j) + \mathrm{dmax}^{\nu}_{\mathrm{cls}}(j) \leq C^{\mathrm{avg}}_{\mathrm{cls}}(\nu) + C^{\max}_{\mathrm{cls}}(\nu), \end{aligned}$$

completing the proof. ■

Now we have completed the proof that the computed partitioning satisfies all the required properties.

Algorithm EBGS**.** The complete algorithm starts with solving the LP(3.1) and computing the partitioning described earlier in this section. Given the partitioned fractional solution $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}})$ with the desired properties, we start the process of opening facilities and making connections to obtain an integral solution. To this end, for each primary demand $\kappa \in P$, we open exactly one facility $\phi(\kappa)$ in $\overline{N}_{\mathrm{cls}}(\kappa)$, where each $\mu \in \overline{N}_{\mathrm{cls}}(\kappa)$ is chosen as $\phi(\kappa)$ with probability $\gamma\bar{y}_\mu$. For all facilities $\mu \in \overline{\mathbb{F}} - \bigcup_{\kappa \in P} \overline{N}_{\mathrm{cls}}(\kappa)$, we open them independently, each with probability $\gamma\bar{y}_\mu$.

We claim that all probabilities are well-defined, that is $\gamma\bar{y}_\mu \leq 1$ for all μ. Indeed, if $\bar{y}_\mu > 0$ then $\bar{y}_\mu = \bar{x}_{\mu\nu}$ for some ν, by Property (CO). If $\mu \in \overline{N}_{\mathrm{cls}}(\nu)$ then the definition of close neighborhoods implies that $\bar{x}_{\mu\nu} \leq 1/\gamma$. If $\mu \in \overline{N}_{\mathrm{far}}(\nu)$ then $\bar{x}_{\mu\nu} \leq 1 - 1/\gamma \leq 1/\gamma$, because $\gamma < 2$. Thus $\gamma\bar{y}_\mu \leq 1$, as claimed.

Next, we connect demands to facilities. Each primary demand $\kappa \in P$ will connect to the only open facility $\phi(\kappa)$ in $\overline{N}_{\mathrm{cls}}(\kappa)$. For each non-primary demand $\nu \in \overline{\mathbb{C}} - P$, if there is an open facility in $\overline{N}_{\mathrm{cls}}(\nu)$ then we connect ν to the nearest such facility. Otherwise, we connect ν to the nearest far facility in $\overline{N}_{\mathrm{far}}(\nu)$ if one is open. Otherwise, we connect ν to its *target facility* $\phi(\kappa)$, where κ is the primary demand that ν is assigned to.

Analysis. By the algorithm, for each client j, all its r_j demands are connected to open facilities. If two different siblings $\nu, \nu' \in j$ are assigned, respectively, to primary demands κ, κ' then, by Properties (SI'.1), (SI'.2), and (PD'.1) we have

$$(\overline{N}(\nu) \cup \overline{N}_{\mathrm{cls}}(\kappa)) \cap (\overline{N}(\nu') \cup \overline{N}_{\mathrm{cls}}(\kappa')) = \emptyset.$$

This condition guarantees that ν and ν' are assigned to different facilities, regardless whether they are connected to a neighbor facility or to its target facility. Therefore the computed solution is feasible.

We now estimate the cost of the solution computed by Algorithm EBGS. The lemma below bounds the expected facility cost.

Lemma 19 *The expectation of facility cost F_{EBGS} of Algorithm* EBGS *is at most γF^*.*

Proof. By the algorithm, each facility $\mu \in \overline{\mathbb{F}}$ is opened with probability $\gamma \bar{y}_\mu$, independently of whether it belongs to the close neighborhood of a primary demand or not. Therefore, by linearity of expectation, we have that the expected facility cost is

$$\mathbb{E}[F_{\text{EBGS}}] = \sum_{\mu \in \overline{\mathbb{F}}} f_\mu \gamma \bar{y}_\mu = \gamma \sum_{i \in \mathbb{F}} f_i \sum_{\mu \in i} \bar{y}_\mu = \gamma \sum_{i \in \mathbb{F}} f_i y_i^* = \gamma F^*,$$

where the third equality follows from (PS.3). ■

In the remainder of this section we focus on the connection cost. Let C_ν be the random variable representing the connection cost of a demand ν. Our objective is to show that the expectation of ν satisfies

$$\mathbb{E}[C_\nu] \leq C^{\text{avg}}(\nu) \cdot \max\left\{ \frac{1/e + 1/e^\gamma}{1 - 1/\gamma}, 1 + \frac{2}{e^\gamma} \right\}. \tag{5.9}$$

If ν is a primary demand then, due to the algorithm, we have $\mathbb{E}[C_\nu] = C^{\text{avg}}_{\text{cls}}(\nu) \leq C^{\text{avg}}(\nu)$, so (5.9) is easily satisfied.

Thus for the rest of the argument we will focus on the case when ν is a non-primary demand. Recall that the algorithm connects ν to the nearest open facility in $\overline{N}_{\text{cls}}(\nu)$ if at least one facility in $\overline{N}_{\text{cls}}(\nu)$ is open. Otherwise the algorithm connects ν to the nearest open facility in $\overline{N}_{\mathrm{far}}(\nu)$, if any. In the event that no facility in $\overline{N}(\nu)$ opens, the algorithm will connect ν to its target facility $\phi(\kappa)$, where κ is the primary demand that ν was assigned to, and $\phi(\kappa)$ is the only facility open in $\overline{N}_{\mathrm{cls}}(\kappa)$. Let Λ^{ν} denote the event that at least one facility in $\overline{N}(\nu)$ is open and $\Lambda^{\nu}_{\mathrm{cls}}$ be the event that at least one facility in $\overline{N}_{\mathrm{cls}}(\nu)$ is open. $\neg\Lambda^{\nu}$ denotes the complement event of Λ^{ν}, that is, the event that none of ν's neighbors opens. We want to estimate the following three conditional expectations:

$$\mathbb{E}[C_{\nu} \mid \Lambda^{\nu}_{\mathrm{cls}}], \quad \mathbb{E}[C_{\nu} \mid \Lambda^{\nu} \wedge \neg\Lambda^{\nu}_{\mathrm{cls}}], \quad \text{and} \quad \mathbb{E}[C_{\nu} \mid \neg\Lambda^{\nu}],$$

and their associated probabilities.

We start with a lemma dealing with the third expectation, $\mathbb{E}[C_{\nu} \mid \neg\Lambda^{\nu}] = \mathbb{E}[d_{\phi(\kappa)\nu} \mid \Lambda^{\nu}]$. The proof of this lemma relies on Properties (PD'.3(a)) and (PD'.3(b)) of modified partitioning and follows the reasoning in the proof of a similar lemma in [8, 7].

Lemma 20 *Assuming that no facility in $\overline{N}(\nu)$ opens, the expected connection cost of ν is*

$$\mathbb{E}[C_{\nu} \mid \neg\Lambda^{\nu}] \leq C^{\mathrm{avg}}_{\mathrm{cls}}(\nu) + 2C^{\mathrm{avg}}_{\mathrm{far}}(\nu). \tag{5.10}$$

Proof. It suffices to show a stronger inequality

$$\mathbb{E}[C_{\nu} \mid \neg\Lambda^{\nu}] \leq C^{\mathrm{avg}}_{\mathrm{cls}}(\nu) + C^{\mathrm{max}}_{\mathrm{cls}}(\nu) + C^{\mathrm{avg}}_{\mathrm{far}}(\nu), \tag{5.11}$$

which then implies (5.10) because $C^{\mathrm{max}}_{\mathrm{cls}}(\nu) \leq C^{\mathrm{avg}}_{\mathrm{far}}(\nu)$. The proof of (5.11) is similar to that in [7]. For the sake of completeness, we provide it here, formulated in our terminology and notation.

Assume that the event $\neg\Lambda^{\nu}$ is true, that is Algorithm EBGS does not open any facility in $\overline{N}(\nu)$. Let κ be the primary demand that ν was assigned to. Also let

$$K = \overline{N}_{\mathrm{cls}}(\kappa) \setminus \overline{N}(\nu), \quad V_{\mathrm{cls}} = \overline{N}_{\mathrm{cls}}(\kappa) \cap \overline{N}_{\mathrm{cls}}(\nu) \quad \text{and} \quad V_{\mathrm{far}} = \overline{N}_{\mathrm{cls}}(\kappa) \cap \overline{N}_{\mathrm{far}}(\nu).$$

Then $K, V_{\text{cls}}, V_{\text{far}}$ form a partition of $\overline{N}_{\text{cls}}(\kappa)$, that is, they are disjoint and their union is $\overline{N}_{\text{cls}}(\kappa)$. Moreover, we have that K is not empty, because Algorithm EBGS opens some facility in $\overline{N}_{\text{cls}}(\kappa)$ and this facility cannot be in $V_{\text{cls}} \cup V_{\text{far}}$, by our assumption. We also have that V_{cls} is not empty due to (PD'.3(a)).

Recall that $D(A, \eta) = \sum_{\mu \in A} d_{\mu\eta} \bar{y}_\mu / \sum_{\mu \in A} \bar{y}_\mu$ is the average distance between a demand η and the facilities in a set A. We shall show that

$$D(K, \nu) \leq C_{\text{cls}}^{\text{avg}}(\kappa) + C_{\text{cls}}^{\text{max}}(\kappa) + C_{\text{far}}^{\text{avg}}(\nu). \tag{5.12}$$

This is sufficient, because, by the algorithm, $D(K, \nu)$ is exactly the expected connection cost for demand ν conditioned on the event that none of ν's neighbors opens, that is the left-hand side of (5.11). Further, (PD'.3(b)) states that $C_{\text{cls}}^{\text{avg}}(\kappa) + C_{\text{cls}}^{\text{max}}(\kappa) \leq C_{\text{cls}}^{\text{avg}}(\nu) + C_{\text{cls}}^{\text{max}}(\nu)$, and thus (5.12) implies (5.11).

The proof of (5.12) is by analysis of several cases.

<u>Case 1</u>: $D(K, \kappa) \leq C_{\text{cls}}^{\text{avg}}(\kappa)$. For any facility $\mu \in V_{\text{cls}}$ (recall that $V_{\text{cls}} \neq \emptyset$), we have $d_{\mu\kappa} \leq C_{\text{cls}}^{\text{max}}(\kappa)$ and $d_{\mu\nu} \leq C_{\text{cls}}^{\text{max}}(\nu) \leq C_{\text{far}}^{\text{avg}}(\nu)$. Therefore, using the case assumption, we get $D(K, \nu) \leq D(K, \kappa) + d_{\mu\kappa} + d_{\mu\nu} \leq C_{\text{cls}}^{\text{avg}}(\kappa) + C_{\text{cls}}^{\text{max}}(\kappa) + C_{\text{far}}^{\text{avg}}(\nu)$.

<u>Case 2</u>: There exists a facility $\mu \in V_{\text{cls}}$ such that $d_{\mu\kappa} \leq C_{\text{cls}}^{\text{avg}}(\kappa)$. Since $\mu \in V_{\text{cls}}$, we infer that $d_{\mu\nu} \leq C_{\text{cls}}^{\text{max}}(\nu) \leq C_{\text{far}}^{\text{avg}}(\nu)$. Using $C_{\text{cls}}^{\text{max}}(\kappa)$ to bound $D(K, \kappa)$, we have $D(K, \nu) \leq D(K, \kappa) + d_{\mu\kappa} + d_{\mu\nu} \leq C_{\text{cls}}^{\text{max}}(\kappa) + C_{\text{cls}}^{\text{avg}}(\kappa) + C_{\text{far}}^{\text{avg}}(\nu)$.

<u>Case 3</u>: In this case we assume that neither of Cases 1 and 2 applies, that is $D(K, \kappa) > C_{\text{cls}}^{\text{avg}}(\kappa)$ and every $\mu \in V_{\text{cls}}$ satisfies $d_{\mu\kappa} > C_{\text{cls}}^{\text{avg}}(\kappa)$. This implies that $D(K \cup V_{\text{cls}}, \kappa) > C_{\text{cls}}^{\text{avg}}(\kappa) = D(\overline{N}_{\text{cls}}(\kappa), \kappa)$. Since sets K, V_{cls} and V_{far} form a partition of $\overline{N}_{\text{cls}}(\kappa)$, we obtain

that in this case V_{far} is not empty and $D(V_{\text{far}},\kappa) < C^{\text{avg}}_{\text{cls}}(\kappa)$. Let $\delta = C^{\text{avg}}_{\text{cls}}(\kappa) - D(V_{\text{far}},\kappa) > 0$. We now have two sub-cases:

Case 3.1: $D(V_{\text{far}},\nu) \le C^{\text{avg}}_{\text{far}}(\nu)+\delta$. Substituting δ, this implies that $D(V_{\text{far}},\nu)+D(V_{\text{far}},\kappa) \le C^{\text{avg}}_{\text{cls}}(\kappa)+C^{\text{avg}}_{\text{far}}(\nu)$. From the definition of the average distance $D(V_{\text{far}},\kappa)$ and $D(V_{\text{far}},\nu)$, we obtain that there exists some $\mu \in V_{\text{far}}$ such that $d_{\mu\kappa} + d_{\mu\nu} \le C^{\text{avg}}_{\text{cls}}(\kappa) + C^{\text{avg}}_{\text{far}}(\nu)$. Thus $D(K,\nu) \le D(K,\kappa) + d_{\mu\kappa} + d_{\mu\nu} \le C^{\max}_{\text{cls}}(\kappa) + C^{\text{avg}}_{\text{cls}}(\kappa) + C^{\text{avg}}_{\text{far}}(\nu)$.

Case 3.2: $D(V_{\text{far}},\nu) > C^{\text{avg}}_{\text{far}}(\nu) + \delta$. The case assumption implies that V_{far} is a proper subset of $\overline{N}_{\text{far}}(\nu)$, that is $\overline{N}_{\text{far}}(\nu) \setminus V_{\text{far}} \neq \emptyset$. Let $\hat{y} = \gamma \sum_{\mu \in V_{\text{far}}} \bar{y}_\mu$. We can express $C^{\text{avg}}_{\text{far}}(\nu)$ using $\hat{y}$ as follows

$$C^{\text{avg}}_{\text{far}}(\nu) = D(V_{\text{far}},\nu)\frac{\hat{y}}{\gamma-1} + D(\overline{N}_{\text{far}}(\nu) \setminus V_{\text{far}},\nu)\frac{\gamma-1-\hat{y}}{\gamma-1}.$$

Then, using the case condition and simple algebra, we have

$$\begin{aligned} C^{\max}_{\text{cls}}(\nu) &\le D(\overline{N}_{\text{far}}(\nu) \setminus V_{\text{far}},\nu) \\ &\le C^{\text{avg}}_{\text{far}}(\nu) - \frac{\hat{y}\delta}{\gamma-1-\hat{y}} \le C^{\text{avg}}_{\text{far}}(\nu) - \frac{\hat{y}\delta}{1-\hat{y}}, \end{aligned} \tag{5.13}$$

where the last step follows from $1 < \gamma < 2$.

On the other hand, since K, V_{cls}, and V_{far} form a partition of $\overline{N}_{\text{cls}}(\kappa)$, we have $C^{\text{avg}}_{\text{cls}}(\kappa) = (1-\hat{y})D(K \cup V_{\text{cls}},\kappa) + \hat{y}D(V_{\text{far}},\kappa)$. Then using the definition of δ we obtain

$$D(K \cup V_{\text{cls}},\kappa) = C^{\text{avg}}_{\text{cls}}(\kappa) + \frac{\hat{y}\delta}{1-\hat{y}}. \tag{5.14}$$

Now we are essentially done. If there exists some $\mu \in V_{\text{cls}}$ such that $d_{\mu\kappa} \le C^{\text{avg}}_{\text{cls}}(\kappa) + \hat{y}\delta/(1-\hat{y})$, then we have

$$
\begin{aligned}
D(K,\nu) &\le D(K,\kappa) + d_{\mu\kappa} + d_{\mu\nu} \\
&\le C^{\max}_{\text{cls}}(\kappa) + C^{\text{avg}}_{\text{cls}}(\kappa) + \frac{\hat{y}\delta}{1-\hat{y}} + C^{\max}_{\text{cls}}(\nu) \\
&\le C^{\max}_{\text{cls}}(\kappa) + C^{\text{avg}}_{\text{cls}}(\kappa) + C^{\text{avg}}_{\text{far}}(\nu),
\end{aligned}
$$

where we used (5.13) in the last step. Otherwise, from (5.14), we must have $D(K,\kappa) \le C^{\text{avg}}_{\text{cls}}(\kappa) + \hat{y}\delta/(1-\hat{y})$. Choosing any $\mu \in V_{\text{cls}}$, it follows that

$$
\begin{aligned}
D(K,\nu) &\le D(K,\kappa) + d_{\mu\kappa} + d_{\mu\nu} \\
&\le C^{\text{avg}}_{\text{cls}}(\kappa) + \frac{\hat{y}\delta}{1-\hat{y}} + C^{\max}_{\text{cls}}(\kappa) + C^{\max}_{\text{cls}}(\nu) \\
&\le C^{\text{avg}}_{\text{cls}}(\kappa) + C^{\max}_{\text{cls}}(\kappa) + C^{\text{avg}}_{\text{far}}(\nu),
\end{aligned}
$$

again using (5.13) in the last step.

This concludes the proof of (5.10). As explained earlier, Lemma 20 follows. ■

Next, we derive some estimates for the expected cost of direct connections. The next technical lemma is a generalization of Lemma 14. In Lemma 14 we bound the expected distance to the closest open facility in $\overline{N}(\nu)$, conditioned on at least one facility in $\overline{N}(\nu)$ being open. The lemma below provides a similar estimate for an arbitrary set A of facilities in $\overline{N}(\nu)$, conditioned on that at least one facility in set A is open. Recall that $D(A,\nu) = \sum_{\mu\in A} d_{\mu\nu}\bar{y}_\mu / \sum_{\mu\in A}\bar{y}_\mu$ is the average distance from ν to a facility in A.

Lemma 21 *For any non-empty set $A \subseteq \overline{N}(\nu)$, let Λ^ν_A be the event that at least one facility in A is opened by Algorithm* EBGS, *and denote by $C_\nu(A)$ the random variable representing the distance from ν to the closest open facility in A. Then the expected distance from ν to the nearest open facility in A, conditioned on at least one facility in A being opened, is*

$$
\mathbb{E}[C_\nu(A) \mid \Lambda^\nu_A] \le D(A,\nu).
$$

Proof. The proof follows the same reasoning as the proof of Lemma 14, so we only sketch it here. We start with a similar grouping of facilities in A: for each primary demand κ, if $\overline{N}_{\mathrm{cls}}(\kappa) \cap A \neq \emptyset$ then $\overline{N}_{\mathrm{cls}}(\kappa) \cap A$ forms a group. Facilities in A that are not in a neighborhood of any primary demand form singleton groups. We denote these groups $G_1, ..., G_k$. It is clear that the groups are disjoint because of (PD'.1). Denoting by $\bar{d}_s = D(G_s, \nu)$ the average distance from ν to a group G_s, we can assume that these groups are ordered so that $\bar{d}_1 \leq ... \leq \bar{d}_k$.

Each group can have at most one facility open and the events representing opening of any two facilities that belong to different groups are independent. To estimate the distance from ν to the nearest open facility in A, we use an alternative random process to make connections, that is easier to analyze. Instead of connecting ν to the nearest open facility in A, we will choose the smallest s for which G_s has an open facility and connect ν to this facility. (Thus we selected an open facility with respect to the minimum $\bar{d}_s$, not the actual distance from ν to this facility.) This can only increase the expected connection cost, thus denoting $g_s = \sum_{\mu \in G_s} \gamma \bar{y}_\mu$ for all $s = 1, \ldots, k$, and letting $\mathbb{P}[\Lambda^\nu_A]$ be the probability that A has at least one facility open, we have

$$\begin{aligned}
\mathbb{E}[C_\nu(A) \mid \Lambda^\nu_A] &\leq \frac{1}{\mathbb{P}[\Lambda^\nu_A]}(\bar{d}_1 g_1 + \bar{d}_2 g_2 (1-g_1) + \ldots + \bar{d}_k g_k (1-g_1)\ldots(1-g_{k-1})) && (5.15)\\
&\leq \frac{1}{\mathbb{P}[\Lambda^\nu_A]} \frac{\sum_{s=1}^k \bar{d}_s g_s}{\sum_{s=1}^k g_s}(1 - \prod_{s=1}^k (1-g_s)) && (5.16)\\
&= \frac{\sum_{s=1}^k \bar{d}_s g_s}{\sum_{s=1}^k g_s} = \frac{\sum_{\mu \in A} d_{\mu\nu} \gamma \bar{y}_\mu}{\sum_{\mu \in A} \gamma \bar{y}_\mu}\\
&= \frac{\sum_{s=1}^k d_{\mu\nu} \bar{y}_\mu}{\sum_{\mu \in A} \bar{y}_\mu} = D(A, \nu).
\end{aligned}$$

Inequality (5.16) follows from inequality (A.3) in A.2. The rest of the derivation follows from $\mathbb{P}[\Lambda_A^\nu] = 1 - \prod_{s=1}^k (1 - g_s)$, and the definition of $\bar{d}_s$, g_s and $D(A, \nu)$. ■

A consequence of Lemma 21 is the following corollary which bounds the other two expectations of C_ν, when at least one facility is opened in $\overline{N}_{\text{cls}}(\nu)$, and when no facility in $\overline{N}_{\text{cls}}(\nu)$ opens but a facility in $\overline{N}_{\text{far}}(\nu)$ is opened.

Corollary 22 (a) $\mathbb{E}[C_\nu \mid \Lambda_{\text{cls}}^\nu] \le C_{\text{cls}}^{\text{avg}}(\nu)$, *and* (b) $\mathbb{E}[C_\nu \mid \Lambda^\nu \wedge \neg\Lambda_{\text{cls}}^\nu] \le C_{\text{far}}^{\text{avg}}(\nu)$.

Proof. When there is an open facility in $\overline{N}_{\text{cls}}(\nu)$, the algorithm connect ν to the nearest open facility in $\overline{N}_{\text{cls}}(\nu)$. When no facility in $\overline{N}_{\text{cls}}(\nu)$ opens but some facility in $\overline{N}_{\text{far}}(\nu)$ opens, the algorithm connects ν to the nearest open facility in $\overline{N}_{\text{far}}(\nu)$. The rest of the proof follows from Lemma 21. By setting the set A in Lemma 21 to $\overline{N}_{\text{cls}}(\nu)$, we have

$$\mathbb{E}[C_\nu \mid \Lambda_{\text{cls}}^\nu] \le D(\overline{N}_{\text{cls}}(\nu), \nu), = C_{\text{cls}}^{\text{avg}}(\nu),$$

proving part (a), and by setting the set A to $\overline{N}_{\text{far}}(\nu)$, we have

$$\mathbb{E}[C_\nu \mid \Lambda^\nu \wedge \neg\Lambda_{\text{cls}}^\nu] \le D(\overline{N}_{\text{far}}(\nu), \nu) = C_{\text{far}}^{\text{avg}}(\nu),$$

which proves part (b). ■

Given the estimate on the three expected distances when ν connects to its close facility in $\overline{N}_{\text{cls}}(\nu)$ in (5.3), or its far facility in $\overline{N}_{\text{far}}(\nu)$ in (5.3), or its target facility $\phi(\kappa)$ in (5.10), the only missing pieces are estimates on the corresponding probabilities of each event, which we do in the next lemma. Once done, we shall put all pieces together and proving the desired inequality on $\mathbb{E}[C_\nu]$, that is (5.9).

The next Lemma bounds the probabilities for events that no facilities in $\overline{N}_{\text{cls}}(\nu)$ and $\overline{N}(\nu)$ are opened by the algorithm.

Lemma 23 (a) $\mathbb{P}[\neg\Lambda^{\nu}_{\text{cls}}] \le 1/e$, *and* (b) $\mathbb{P}[\neg\Lambda^{\nu}] \le 1/e^{\gamma}$.

Proof. (a) To estimate $\mathbb{P}[\neg\Lambda^{\nu}_{\text{cls}}]$, we again consider a grouping of facilities in $\overline{N}_{\text{cls}}(\nu)$, as in the proof of Lemma 21, according to the primary demand's close neighborhood that they fall in, with facilities not belonging to such neighborhoods forming their own singleton groups. As before, the groups are denoted $G_1, \dots, G_k$. It is easy to see that $\sum_{s=1}^{k} g_s = \sum_{\mu \in \overline{N}_{\text{cls}}(\nu)} \gamma \bar{y}_{\mu} = 1$. For any group G_s, the probability that a facility in this group opens is $\sum_{\mu \in G_s} \gamma \bar{y}_{\mu} = g_s$ because in the algorithm at most one facility in a group can be chosen and each is chosen with probability $\gamma \bar{y}_{\mu}$. Therefore the probability that no facility opens is $\prod_{s=1}^{k}(1 - g_s)$, which is at most $e^{-\sum_{s=1}^{k} g_s} = 1/e$. Therefore we have $\mathbb{P}[\neg\Lambda^{\nu}_{A}] \le 1/e$.

(b) This proof is similar to the proof of (a). The probability $\mathbb{P}[\neg\Lambda^{\nu}]$ is at most $e^{-\sum_{s=1}^{k} g_s} = 1/e^{\gamma}$, because we now have $\sum_{s=1}^{k} g_s = \gamma \sum_{\mu \in \overline{N}(\nu)} \bar{y}_{\mu} = \gamma \cdot 1 = \gamma$. ■

We are now ready to bound the overall connection cost of Algorithm EBGS, namely inequality (5.9).

Lemma 24 *The expected connection of* ν *is*

$$\mathbb{E}[C_{\nu}] \le C^{\text{avg}}(\nu) \cdot \max\left\{\frac{1/e + 1/e^{\gamma}}{1 - 1/\gamma}, 1 + \frac{2}{e^{\gamma}}\right\}.$$

Proof. Recall that, to connect ν, the algorithm uses the closest facility in $\overline{N}_{\text{cls}}(\nu)$ if one is opened; otherwise it will try to connect ν to the closest facility in $\overline{N}_{\text{far}}(\nu)$. Failing that, it will connect ν to $\phi(\kappa)$, the sole facility open in the neighborhood of κ, the primary demand ν was assigned to. Given that, we estimate $\mathbb{E}[C_{\nu}]$ as follows:

$$
\begin{aligned}
\mathbb{E}[C_\nu] \;&=\; \mathbb{E}[C_\nu \mid \Lambda^\nu_{\text{cls}}] \cdot \mathbb{P}[\Lambda^\nu_{\text{cls}}] \;+\; \mathbb{E}[C_\nu \mid \Lambda^\nu \;\wedge \neg\Lambda^\nu_{\text{cls}}] \cdot \mathbb{P}[\Lambda^\nu \;\wedge \neg\Lambda^\nu_{\text{cls}}] \\
&\qquad +\; \mathbb{E}[C_\nu \mid \neg\Lambda^\nu] \cdot \mathbb{P}[\neg\Lambda^\nu] \\
&\le\; C^{\text{avg}}_{\text{cls}}(\nu) \cdot \mathbb{P}[\Lambda^\nu_{\text{cls}}] \;+\; C^{\text{avg}}_{\text{far}}(\nu) \cdot \mathbb{P}[\Lambda^\nu \;\wedge \neg\Lambda^\nu_{\text{cls}}] \qquad (5.17) \\
&\qquad +\; [\, C^{\text{avg}}_{\text{cls}}(\nu) + 2C^{\text{avg}}_{\text{far}}(\nu)\,] \cdot \mathbb{P}[\neg\Lambda^\nu] \\
&=\; [\, C^{\text{avg}}_{\text{cls}}(\nu) + C^{\text{avg}}_{\text{far}}(\nu)\,] \cdot \mathbb{P}[\neg\Lambda^\nu] \;+\; [\, C^{\text{avg}}_{\text{far}}(\nu) - C^{\text{avg}}_{\text{cls}}(\nu)\,] \cdot \mathbb{P}[\neg\Lambda^\nu_{\text{cls}}] \;+\; C^{\text{avg}}_{\text{cls}}(\nu) \\
&\le\; [\, C^{\text{avg}}_{\text{cls}}(\nu) + C^{\text{avg}}_{\text{far}}(\nu)\,] \cdot \frac{1}{e^\gamma} \;+\; [\, C^{\text{avg}}_{\text{far}}(\nu) - C^{\text{avg}}_{\text{cls}}(\nu)\,] \cdot \frac{1}{e} \;+\; C^{\text{avg}}_{\text{cls}}(\nu) \qquad (5.18) \\
&=\; \Big(1 - \frac{1}{e} + \frac{1}{e^\gamma}\Big) \cdot C^{\text{avg}}_{\text{cls}}(\nu) \;+\; \Big(\frac{1}{e} + \frac{1}{e^\gamma}\Big) \cdot C^{\text{avg}}_{\text{far}}(\nu).
\end{aligned}
$$

Inequality (5.17) follows from Corollary 22 and Lemma 20. Inequality (5.18) follows from Lemma 23 and $C^{\text{avg}}_{\text{far}}(\nu) - C^{\text{avg}}_{\text{cls}}(\nu) \ge 0$.

Now define $\rho = C^{\text{avg}}_{\text{cls}}(\nu)/C^{\text{avg}}(\nu)$. It is easy to see that ρ is between 0 and 1. Continuing the above derivation, applying (5.8), we get

$$
\begin{aligned}
\mathbb{E}[C_\nu] \;&\le\; C^{\text{avg}}(\nu) \cdot \left((1-\rho)\frac{1/e + 1/e^\gamma}{1 - 1/\gamma} + \rho(1 + \frac{2}{e^\gamma}) \right) \\
&\le C^{\text{avg}}(\nu) \cdot \max\left\{ \frac{1/e + 1/e^\gamma}{1 - 1/\gamma}, 1 + \frac{2}{e^\gamma} \right\},
\end{aligned}
$$

and the proof is now complete. ■

With Lemma 24 proven, we are now ready to bound our total connection cost. For any client j we have

$$
\begin{aligned}
\sum_{\nu \in j} C^{\text{avg}}(\nu) &= \sum_{\nu \in j} \sum_{\mu \in \overline{\mathbb{F}}} d_{\mu\nu} \bar{x}_{\mu\nu} \\
&= \sum_{i \in \mathbb{F}} d_{ij} \sum_{\mu \in i} \sum_{\nu \in j} \bar{x}_{\mu\nu} = \sum_{i \in \mathbb{F}} d_{ij} x^*_{ij} = C^*_j.
\end{aligned}
$$

Summing over all clients j we obtain that the total expected connection cost is

$$
\mathbb{E}[C_{\text{EBGS}}] \le C^* \max\left\{ \frac{1/e + 1/e^\gamma}{1 - 1/\gamma}, 1 + \frac{2}{e^\gamma} \right\}.
$$

Recall that the expected facility cost is bounded by γF^*, as argued earlier. Hence the total expected cost is bounded by $\max\{\gamma, \frac{1/e+1/e^\gamma}{1-1/\gamma}, 1+\frac{2}{e^\gamma}\} \cdot \mathrm{LP}^*$. Picking $\gamma = 1.575$ we obtain the desired ratio.

Theorem 25 *Algorithm* EBGS *is a* 1.575*-approximation algorithm for* FTFP*.*

Chapter 6

Primal-dual Algorithms

In this chapter, we present the results on primal-dual algorithms. Unlike the LP-rounding algorithms in Chapter 5, primal-dual algorithms do not require solving the LP explicitly and are computationally more efficient. Primal-dual algorithms work by making simultaneous updates to a primal solution, which is integral, and a dual solution, which may be fractional, and eventually arriving at a feasible primal solution and a feasible dual solution. It is then possible to compare the primal solution's cost to the optimal value. Assuming that the primal problem is a minimization problem, the optimal value of the primal program is lower bounded by the cost of any feasible dual solution.

We present a natural greedy algorithm and employ a technique called *dual-fitting* to analyze the approximation ratio. We also give an example showing one possible limitation of this analysis. In Section 6.1, we review the related work by Jain *et al.* [26] about the greedy algorithm and the dual-fitting analysis for UFL. In Section 6.2, we explain how a similar algorithm can be used to solve FTFP, and derive an approximation ratio using the dual-fitting analysis. Lastly, in Section 6.3, we provide an example illustrating the difference between UFL and FTFP when the dual-fitting analysis is used to derive the approximation ratio.

6.1 Greedy Algorithm and Dual-fitting Analysis for UFL

Jain *et al.* [26] analyzed a greedy algorithm with a ratio of 1.861 for UFL using dual-fitting. The algorithm works by repeatedly picking the most cost-effective star until all clients are connected. A star consists of a facility i and a set of clients C'. The cost of the star (i, C') is $f_i + \sum_{j \in C'} d_{ij}$. The cost-effectiveness, or average-cost, is the cost of the star divided by the number of clients in that star. Clients in the just-selected star are connected to the facility, and the opening cost of that facility is then set to zero. The greedy algorithm can be interpreted as an equivalent algorithm that grows a dual solution and updates the corresponding primal solution. Each client j is associated with a dual variable α_j, and α_j is fixed to the average cost of the star when j gets connected. Clearly, the sum of α_j for all clients j is equal to the cost of the primal solution, which is the cost to open facilities and the cost to make connections.

The next step is to find a suitable common factor γ to make the dual solution $\{\alpha_j/\gamma\}$ feasible. For this purpose, Jain *et al.* derived an upper bound on the optimal values of a series of linear programs, which are used to capture the hardest instance for the algorithm. These linear programs are called the *factor-revealing LPs*, and the supremum of their optimal values is shown to be equal to γ, which is an upper bound on the approximation ratio of the algorithm.

The greedy algorithm, as stated, is very flexible and can be applied with only minor modifications to solve problems with fault-tolerant requirements, for example the FTFL problem and the FTFP problem. However, it seems rather difficult to generalize the analysis to the more general problems, and we have to settle for a much worse approximation ratio except for some special cases. In the literature, there is no published result on the approximation ratio of the greedy algorithm for FTFL, although an H_n-approximation ratio [1] is not difficult to obtain. For the uniform demands case, Swamy and Shmoys [46] showed that the dual-fitting analysis for UFL can be generalized for FTFL to obtain the same ratio of 1.52. For the FTFP problem studied in this thesis, the uniform demands case is trivial, as it is nothing but a UFL problem. We have the following results in the next two sections: for the general demands case, we show a logarithmic ratio for the greedy algorithm using the dual-fitting analysis; we then present an example illustrating the difficulty in obtaining an $O(1)$-approximation ratio for the FTFP problem when dual-fitting is used.

6.2 Greedy algorithm for FTFP with Ratio $O(\log n)$

6.2.1 Greedy Algorithm for FTFP

In this section, we show that a greedy algorithm for FTFP similar to the one for UFL in Section 6.1 gives an approximation ratio of $H_n \approx \ln(n)$, where $n = |\mathbb{C}|$ is the number of clients. The greedy algorithm proceeds by picking a star and connecting member clients to the site of that star. A star is a site i together with a subset of clients C'. The cost of a star will be given as we present the greedy algorithm below.

[1] The term H_n is the n^{th} harmonic number, defined as $H_n = 1 + 1/2 + 1/3 + \ldots + 1/n \approx \ln n$.

Call a client j *fully-connected* if j has made r_j connections. Let U be the set of not fully-connected clients. The greedy algorithm for FTFP iterates until all clients are fully-connected. In one iteration, let y_i be the number of facilities opened at site i, and x_{ij} be the number of connections between site i and client j. Then, the following stars are available to the algorithm:

- For every site $i \in \mathbb{F}$, and every nonempty subset of clients $C' \subseteq U$, we have one star $S = (i, C')$ with a cost $c(S) = f_i + \sum_{j \in C'} d_{ij}$. We call this type of star a *paid* star.
- For every site $i \in \mathbb{F}$, and every nonempty subset of clients $C' \subseteq U$ such that $x_{ij} < y_i$ for every $j \in C'$, we have one star $S = (i, C')$ with a cost $c(S) = \sum_{j \in C'} d_{ij}$. In other words, the facility is *free* in this star, and we call this type of star a *free* star.

The average cost of a star $S = (i, C')$ is always $c(S)/|C'|$, that is, the cost of the star divided by the number of clients in that star. The greedy algorithm then picks some star $S = (i, C')$ with the minimum average cost. If this star is a paid star, the algorithm opens one more facility at the site i. Regardless of whether the star is paid or free, the algorithm make one more connection from every client j in C' to the site i.

To see that the algorithm can be implemented to run in polynomial time, we have two observations to allow multiple iterations to be completed in one step and the number of steps are polynomial in the problem size.

Our first observation is that once a star becomes the best star, it remains the best until we can no longer select the same star. The latter might happen when one of the two cases hold:

- Case 1: The star is a free star. Then, without loss of generality, we may assume that this star contains exactly one client. This star will not be available to the algorithm after the client j has connected to all free facilities at that site i, that is, $x_{ij} = y_i$.

- Case 2: The star is a paid star. Then the star cannot be selected once one or more of the member clients become fully-connected and are removed from the set U.

Our second observation is that, although the number of possible stars is exponential, the best paid star can be identified by considering each site with the nearest $1, 2, \ldots, k$ clients for some integer k, where adding the $(k+1)^{th}$ client would increase the average cost. For free stars, since the greedy algorithm has not specified how to break ties when multiple stars have the same average cost, we may assume that the best free star is a star with just one client. Therefore, we can find the best free star by considering each (i, j) pair for every site i and every client j with $x_{ij} < y_i$. The overall best star is then the better of the best paid star and the best free star. It follows that we can accomplish multiple iterations in a single step, and the number of steps is polynomially bounded by $|\mathbb{F}| \cdot |\mathbb{C}|$. Therefore, the greedy algorithm runs in polynomial time.

6.2.2 Analysis

We now derive an approximation ratio of the greedy algorithm just described. For this purpose, we adapt the dual-fitting analysis for UFL to FTFP, though we have to settle for a much less satisfying ratio of H_n.

When we run the greedy algorithm, we associate each demand of a client j with a number α_j^l, which is the average cost of the star when the l^{th} demand of client j is connected.

Let $\alpha_j = \alpha_j^{r_j}$, and order the clients non-decreasingly by their α_j values, that is,

$$\alpha_1 \leq \alpha_2 \leq \ldots \leq \alpha_n.$$

For the α_j values ordered non-decreasingly, one invariant that the greedy algorithm maintains is: for every $j = 1, \ldots, n$, where $n = |\mathbb{C}|$ is the number of clients, we have

$$\sum_{l=j}^{n} (\alpha_j - d_{il})_+ \leq f_i \quad \forall i \in \mathbb{F}.$$

The notation $(\cdot)_+$ means taking the maximum of the value or 0. The reason is that, when the last demand of client j is connected, all the clients $j+1, \ldots, n$ are still active so their total contribution to the site i cannot exceed f_i.

Let us take a closer look at the numbers $\{\alpha_j\}$. We know that the algorithm's total cost is exactly $\sum_{j=1}^{n} \sum_{l=1}^{r_j} \alpha_j^l$, which is not more than $\sum_{j=1}^{n} r_j \alpha_j$, because we take α_j to be $\alpha_j^{r_j}$. If we can show that $\sum_{j=1}^{n} r_j \alpha_j$ is not more than $\gamma \cdot \mathrm{OPT}$, where OPT is the cost of an integral optimal solution, then we claim our algorithm returns an integral solution within a factor of γ from the optimal value OPT.

We show that $\sum_{j=1}^{n} r_j \alpha_j$ is within a factor of γ from OPT by showing that $\{\alpha_j/\gamma\}$ is a feasible dual solution to the following program, which is a different way to write the dual program of the LP (3.1).

$$\begin{aligned} \text{maximize} \ & \sum_j r_j \alpha_j \\ \text{subject to} \ & \sum_{j=1}^{n} (\alpha_j - d_{ij})_+ \leq f_i \quad \forall i \in \mathbb{F} \end{aligned} \tag{6.1}$$

We repeat the LP (3.1) below for the reader's convenience.

$$
\begin{array}{lll}
\text{minimize} & \sum_{i \in \mathbb{F}} f_i y_i + \sum_{i \in \mathbb{F}, j \in \mathbb{C}} d_{ij} x_{ij} & \\
\text{subject to} & y_i - x_{ij} \geq 0 & \forall i \in \mathbb{F}, j \in \mathbb{C} \\
& \sum_{i \in \mathbb{F}} x_{ij} \geq r_j & \forall j \in \mathbb{C} \\
& x_{ij} \geq 0, y_i \geq 0 & \forall i \in \mathbb{F}, j \in \mathbb{C}
\end{array}
$$

Obviously, we want such a value γ to be as small as possible, because in the dual-fitting analysis, we take γ as our approximation ratio and we want the ratio to be small. To find the minimum γ that would always shrink $\{\alpha_j\}$ to be dual feasible, we need to find a worst case instance that maximizes γ. Moreover, the worst case instance must contain a star whose feasibility requirement realizes the value of γ. It is also clear that this star must also be the worst star in that instance.

As a first step, we can drop the $\max\{0, \cdot\}$ operation, because we can always find a new star by dropping those clients j whose $(\alpha_j - d_{ij})$ terms are negative. That new star would still be a worst case star. Suppose that the worst case star has k clients, and contains a site i. We then have

$$\sum_{j=1}^{k} \alpha_j - d_{ij} \leq f_i.$$

In the summation above, we rename the clients in the new star to be $1, \ldots, k$. These clients are still ordered non-decreasingly by their α_j values. Now, our goal is to find the supremum of the following program for all possible values of $k = 1, 2, \ldots$.

$$
\begin{array}{ll}
\text{maximize} & \dfrac{\sum_{j=1}^{k} \alpha_j}{f_i + \sum_{j=1}^{k} d_{ij}} \\
\text{subject to} & \sum_{l=j}^{k} (\alpha_l - d_{il})_+ \leq f_i \quad \text{for } j = 1, \ldots, k
\end{array}
$$

Because we are dealing with a particular star, we can abstract away the site i to obtain the following program:

$$\begin{aligned} &\text{maximize } \frac{\sum_{j=1}^{k} \alpha_j}{f + \sum_{j=1}^{k} d_j} \\ &\text{subject to } \sum_{l=j}^{k} (\alpha_j - d_l)_+ \leq f \quad \text{for } j = 1, \ldots, k \end{aligned} \tag{6.2}$$

Because (6.2) is a maximization program, we claim that we can again drop the $\max\{0, \cdot\}$ operation: doing so would relax the constraints, and therefore can only make the optimal value larger. The real optimal value of (6.2) is then upper bounded by the optimal value of the relaxed program (6.3).

$$\begin{aligned} &\text{maximize } \frac{\sum_{j=1}^{k} \alpha_j}{f + \sum_{j=1}^{k} d_j} \\ &\text{subject to } \sum_{l=j}^{k} (\alpha_j - d_l) \leq f \quad \text{for } j = 1, \ldots, k \end{aligned} \tag{6.3}$$

For each $j = 1, \ldots, k$, the corresponding constraint in (6.3) can be rewritten as:

$$(k - j + 1)\alpha_j \leq f + \sum_{l=j}^{k} d_l \leq f + \sum_{l=1}^{k} d_l. \tag{6.4}$$

The first inequality is a rewrite of the constraint in (6.3), and the second is straightforward. We now have $\alpha_j \leq (1/(k - j + 1))(f + \sum_{j=1}^{k} d_j)$, and it is then easy to see that

$$\sum_{j=1}^{n} \alpha_j/(f + \sum_{j=1}^{k} d_j) \leq (1/k + 1/(k-1) + \ldots + 1) = H_k \leq H_n. \tag{6.5}$$

Thus, we can take $\gamma = H_n$, and we have that $\{\alpha_j/\gamma\}$ is a feasible dual solution to (6.1). It follows that the greedy algorithm is an H_n-approximation.

The H_n-approximation is hardly the best possible approximation ratio for the greedy algorithm. In deriving the H_n ratio, we have not even used the triangle inequality, though we are working on metric FTFP. On the other hand, similar attempts made by other researchers to obtain a sub-logarithmic ratio for FTFL were not successful, as described in Section 6.1. Although FTFP seems to be easier to approximate than FTFL when LP-rounding algorithms are used, as we have shown in Chapter 5, it seems that the fault-tolerant requirement in both problems presents a hurdle for primal-dual based algorithms. In the following section, we provide an example illustrating some of the difficulties when adapting the dual-fitting analysis to the FTFP problem.

6.3 Limitation of Dual-fitting for FTFP

For FTFP, the greedy algorithm that repeatedly picks the best star until all clients become fully-connected can be implemented in polynomial time. In Section 6.2, we showed that this algorithm is an H_n-approximation where $n = |\mathcal{C}|$ is the number of clients. Since the same greedy algorithm is shown to have an $O(1)$-approximation ratio for UFL [36], a natural question is whether the greedy algorithm can be shown to have an $O(1)$ approximation ratio. Here, we give an example that hints at a negative answer.

We assume that the greedy algorithm is analyzed using the dual-fitting technique, which associates every client j with a number α_j, interpreted as a dual solution to the LP (3.2). However, the dual solution $\{\alpha_j\}$ in general may not be feasible. The dual-fitting technique aims at finding the smallest possible number γ such that, after the dual solution $\{\alpha_j\}$ is shrunk (divided) by γ, all dual constraints are satisfied, that is,

$$\sum_{j\in\mathcal{C}}(\alpha_j/\gamma - d_{ij})_+ \leq f_i \qquad \text{for all } i \in \mathcal{F}.$$

The number γ is taken as the approximation ratio.

In the greedy algorithm, a star with the minimum average cost is picked at each iteration, and each member client of that star then gets one more connection. The dual value α_j associated with each client j in the dual-fitting analysis can be seen as a way to charge the cost to individual clients. Although charging each member client an equal share works well for UFL, the same may not be true for FTFP. In general, the dual-fitting analysis does have the freedom of distributing the cost of f_i into member clients. Nonetheless, we assume that the cost of f_i is distributed among members only, and not to clients outside this star. We call this assumption the *local charging* assumption. Our second assumption is that the proposed dual solution α_j is taken as the average of individual α_j^l values, that is, $\alpha_j = \sum_{l=1}^{r_j} \alpha_j^l / r_j$. Suppose that the l^{th} demand of j is satisfied while j is in a star with a site i; then $\alpha_j^l = d_{ij} + f_i^{j,l}$, where $f_i^{j,l}$ is the portion of f_i attributed to j in the analysis. Taking the average implies the computed α_j values make $\sum_{j \in \mathbb{C}} r_j \alpha_j$ equal to the cost of the integral solution produced by the greedy algorithm. Under these two assumptions, the example we give below shows that the dual-fitting analysis cannot show an approximation ratio better than $O(\log n / \log \log n)$. In particular, this analysis cannot be used to show an approximation ratio of $O(1)$.

We now give our example, illustrated in Figure 6.1. Our example has one site and k groups of clients. Opening one facility at that site costs f_1. The first group has n_1 clients, each with a demand of r_1, and all those clients are at distance $d_1 = 0$ from the site. The distances from each of the groups to the site are listed below. Distances that are not shown explicitly are assumed to follow the triangle inequality. Notice that the distances are chosen

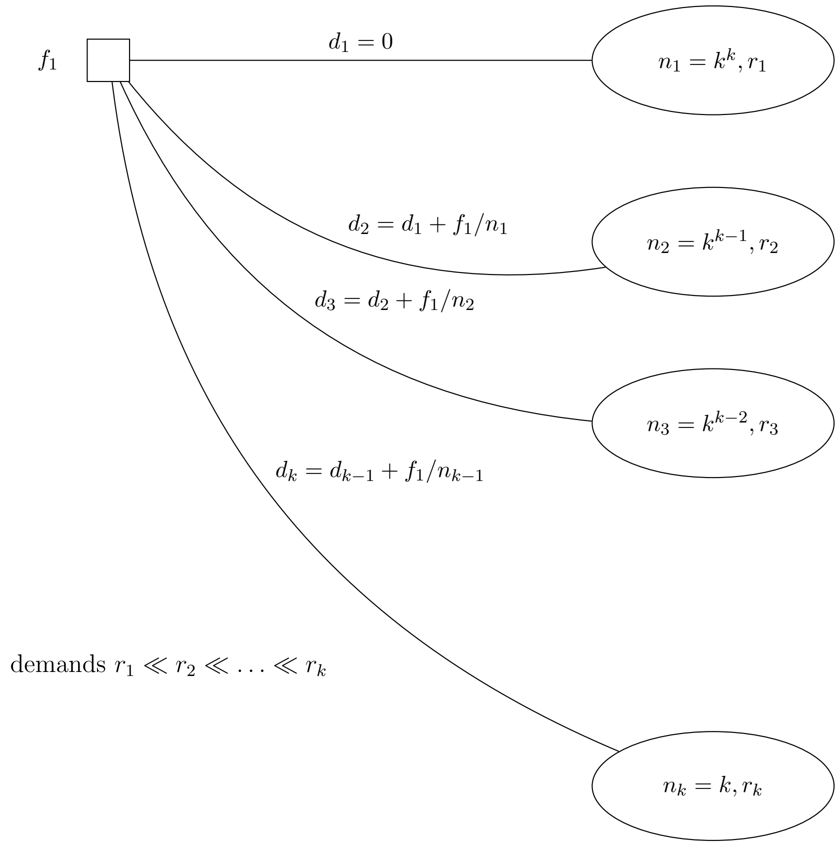


Figure 6.1: An example showing that the greedy algorithm for FTFP analyzed using dual-fitting, could give a ratio of $\Omega(\log n/\log\log n)$, assuming that the facility cost of a star can only be charged to clients within the star.

in a way to force a particular choice of the best stars.

$$
\begin{aligned}
d_1 &= 0 \\
d_2 &= f_1/n_1 \\
d_3 &= f_1/n_2 + d_2 = f_1/n_2 + f_1/n_1 = f_1(\frac{1}{n_2} + \frac{1}{n_1}) \\
&\ldots \\
d_k &= f_1/n_{k-1} + d_{k-1} = f_1(\frac{1}{n_{k-1}} + \ldots + \frac{1}{n_1})
\end{aligned}
$$

For the numbers, we need $r_1 \ll r_2 \ll \ldots \ll r_k$, and $n_1 = u^k, n_2 = u^{k-1}, \ldots, n_k = u^1 = u$ for some number u. We actually take $u = k$ in the end, and this may not be the best possible choice.

Call a star with facility cost zero *trivial*. A star is *non-trivial* if the facility has non-zero cost. Now the greedy algorithm executes like this: first we have a non-trivial star of (f_1, n_1) with r_1 replicas. Then we have a trivial star of a zero-cost facility and all n_2 clients in group 2 with r_1 replicas. The second non-trivial star is (f_1, n_2), with $r_2 - r_1$ replicas. Notice that the r_1 replicas of trivial stars with group 2 satisfy the first r_1 demand of the n_2 clients in that group. After that, the n_2 clients of group 2 each have a residual demand of $r_2 - r_1 \approx r_2$ (since $r_2 \gg r_1$), and are then satisfied by the $r_2 - r_1$ replicas of the (f_1, n_2) stars. The process repeats until the k^{th} group finishes with r_k new facilities.

According to our local charging assumption, we have $\alpha_1 = f_1$, with α_1 defined as the total dual value of the n_1 clients in group 1, regardless of how the analysis distributes the facility cost within that group. Similarly, $\alpha_2 = f_1 + n_2 d_2$, and so on. Substituting in the numbers, we have

$$\begin{aligned}
&\alpha_1 = f_1 \\
&\alpha_2 = f_1 + n_2 d_2 = f_1 + f_1/n_1 \cdot n_2 = f_1(1 + n_2/n_1) \\
&\alpha_3 = f_1 + n_3 d_3 = f_1 + f_1(\frac{1}{n_2} + \frac{1}{n_1})n_3 = f_1(1 + \frac{n_3}{n_2} + \frac{n_3}{n_1}) \\
&\dots \\
&\alpha_k = f_1 + n_k d_k = f_1 + f_1 n_k(\frac{1}{n_{k-1}} + \dots \frac{1}{n_1})
\end{aligned}$$

Notice that $r_1 \ll r_2 \ll \ldots \ll r_k$ implies that α_j is decided by the max among α_j^l, so in the following calculation we ignore the terms involving trivial stars.

Now going back to the dual constraint. The constraint requires the shrinking factor γ to satisfy the following inequality:

$$\frac{\alpha_1}{\gamma} - d_1 + \frac{\alpha_2}{\gamma} - d_2 + \ldots + \frac{\alpha_k}{\gamma} - d_k \leq f_1, \tag{6.6}$$

which we use to derive an upper bound on γ. Substituting in the α_j values derived above, we have

$$\begin{aligned}
\gamma &\geq (\sum_{j=1}^{k} \alpha_j)/(f_1 + \sum_{j=1}^{k} d_j) \\
&\geq \frac{f_1 + n_1 d_1 + f_1 + n_2 d_2 + f_1 + n_3 d_3 + \ldots + f_1 + n_k d_k}{f_1 + n_1 d_1 + n_2 d_2 + \ldots + n_k d_k} \\
&= 1 + (k-1) f_1 / (f_1 + n_1 d_1 + n_2 d_2 + \ldots + n_k d_k) \\
&= 1 + (k-1) f_1 / \left(f_1 + n_2 f_1 / n_1 + \ldots + n_k f_1 (\frac{1}{n_{k-1}} + \frac{1}{n_{k-2}} + \ldots + \frac{1}{n_1}) \right) \\
&= 1 + (k-1) / \left(1 + n_2/n_1 + \ldots + n_k (\frac{1}{n_{k-1}} + \frac{1}{n_{k-2}} + \ldots + \frac{1}{n_1}) \right) \\
&= 1 + (k-1) / \left(1 + 1/u + \ldots + (\frac{1}{u} + \ldots + \frac{1}{u^{k-1}}) \right) \\
&= 1 + (k-1) / \left(1 + k/u + (k-1)/u^2 + \ldots + 1/u^{k-1} \right) \\
&\geq 1 + (k-1) / \left(1 + k/u + k/u^2 + \ldots + k/u^{k-1} \right) \\
&= 1 + (k-1) / \left(1 + 1 + 1/k + \ldots + 1/k^{k-2} \right) \\
&\approx k/2
\end{aligned}$$

Thus, for k groups we can force a shrinking factor γ as big as $k/2$. Earlier we have shown that the approximation ratio of the greedy algorithm is not more than H_n. Is this a contradiction? No, because we have the number of clients $n = k^k + k^{k-1} + \ldots + 1 \approx k^k$, and hence $k = O(\log n / \log\log n)$. Therefore, the example shows that the dual-fitting analysis with the local-charging assumption cannot hope to get a ratio better than $O(\log n / \log\log n)$. Notice this example is similar in spirit to Mahdian *et al.*'s [26] $\Omega(\log n / \log\log n)$ example for Hochbaum's algorithm for the UFL problem. In both their example and ours, we have the numbers of clients across groups decrease exponentially; while the distances are increasing at about the same rate. This choice seems to strike the right balance when devising bad examples for the greedy algorithm. The reason is that, in these examples, we want a fast growing sequence; while on the other hand, we also need the sequence to be not too short.

Notice that our example only shows that a particular type of analysis, that is dual-fitting with the local charging assumption, cannot be used to show an $O(1)$ approximation ratio for the greedy algorithm. Our example did not rule out the possibility to analyze the greedy algorithm using a different technique and give an $O(1)$ ratio. This concludes our discussions on primal-dual algorithm and this chapter.

Chapter 7

Conclusion

In this thesis we have studied the Fault-Tolerant Facility Placement problem (FTFP), a generalization of the well-known Uncapacitated Facility Location problem (UFL). We showed that the known LP-rounding algorithms for UFL can be adapted to FTFP while preserving the approximation ratio. To accomplish this reduction, we developed two techniques, namely demand reduction and adaptive partitioning, which could be of more general interest. Our results show that FTFP seems easier in terms of approximation, compared to another related problem, FTFL. It would be interesting to see if similar ideas can be used to design an LP-rounding algorithm for FTFL with a matching ratio.

We have also studied the primal-dual approaches, and provided a possible explanation of the difficulty in obtaining a constant approximation ratio using those techniques.

In anticipating future research, we agree with Byrka *et al.* [10] who remarked that both UFL and FTFL are likely to have approximation algorithms with a ratio matching the 1.463 lower bound. Due to our results by demand reduction, if FTFL can be approximated with a ratio of 1.463 with respect to the optimal fractional solution, then the same ratio holds for FTFP as well.

Bibliography

[1] Karen Aardal, Fabián Chudak, and David Shmoys. A 3-approximation algorithm for the k-level uncapacitated facility location problem. *Inf. Process. Lett.*, 72:161–167, 1999.

[2] Karen Aardal, Martine Labb, Janny Leung, and Maurice Queyranne. On the two-level uncapacitated facility location problem. *Informs J. Comput.*, 8:289–301, 1996.

[3] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristic for k-median and facility location problems. In *Proc. 33rd Symp. Theory of Computing (STOC)*, pages 21–29, 2001.

[4] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM J. Comput.*, 33(3):544–562, 2004.

[5] M.L. Balinski. On finding integer solutions to linear programs. In *Proceedings of the IBM Scientific Computing Symposium on Combinatorial Problems*, pages 225–248, 1966.

[6] Jaroslaw Byrka. An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. In *Proc. 10th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 29–43, 2007.

[7] Jaroslaw Byrka and Karen Aardal. An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. *SIAM J. Comput.*, 39(6):2212–2231, 2010.

[8] Jaroslaw Byrka, MohammadReza Ghodsi, and Aravind Srinivasan. LP-rounding algorithms for facility-location problems. *CoRR*, abs/1007.3611, 2010.

[9] Jaroslaw Byrka and Bartosz Rybicki. Improved lp-rounding approximation algorithm for k-level uncapacitated facility location. In *Proc. 39rd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 157–169, 2012.

[10] Jaroslaw Byrka, Aravind Srinivasan, and Chaitanya Swamy. Fault-tolerant facility location, a randomized dependent LP-rounding algorithm. In *Proc. 14th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 244–257, 2010.

[11] Moses Charikar and Sudipto Guha. Improved combinatorial algorithms for facility location problems. *SIAM J. Comput.*, 34(4):803–824, 2005.

[12] Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. A constant-factor approximation algorithm for the k-median problem. *J. Comput. Syst. Sci.*, 65(1):129–149, 2002.

[13] Moses Charikar, Samir Khuller, David Mount, and Giri Narasimhan. Algorithms for facility location problems with outliers. In *Proc. 12th Symp. on Discrete Algorithms (SODA)*, pages 642–651, 2001.

[14] Moses Charikar and Shi Li. A dependent lp-rounding approach for the k-median problem. In *Proc. 39rd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 194–205, 2012.

[15] Fabián Chudak and David Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. *SIAM J. Comput.*, 33(1):1–25, 2004.

[16] Fabián A. Chudak. Improved approximation algorithms for uncapitated facility location. In *Proc. 6th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 180–194, 1998.

[17] Vašek Chvátal. *Linear Programming.* W. H. Freeman and Company, 1983.

[18] David Shmoys David Williamson. *The Design of Approximation Algorithms.* Cambridge University Press, 2011.

[19] Dimitris Fotakis. On the competitive ratio for online facility location. *Algorithmica*, 50(1):1–57, 2007.

[20] Dimitris Fotakis. A primal-dual algorithm for online non-uniform facility location. *J. of Discrete Algorithms*, 5(1):141–148, 2007.

[21] Sudipto Guha and Samir Khuller. Greedy strikes back: improved facility location algorithms. In *Proc. 9th Symp. on Discrete Algorithms (SODA)*, pages 649–657, 1998.

[22] Sudipto Guha, Adam Meyerson, and Kamesh Munagala. A constant factor approximation algorithm for the fault-tolerant facility location problem. *J. Algorithms*, 48(2):429–440, 2003.

[23] Anupam Gupta. Lecture notes: CMU 15-854b, spring 2008, 2008.

[24] G.H. Hardy, John Littlewood, and George Pólya. *Inequalities.* Cambridge University Press, 1988.

[25] Dorit Hochbaum. Heuristics for the fixed cost median problem. *Math. Program.*, 22:148–162, 1982.

[26] Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *J. ACM*, 50(6):795–824, 2003.

[27] Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In *Proc. 34th Symp. Theory of Computing (STOC)*, pages 731–740, 2002.

[28] Kamal Jain and Vijay Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *J. ACM*, 48(2):274–296, 2001.

[29] Kamal Jain and Vijay Vazirani. An approximation algorithm for the fault tolerant metric facility location problem. *Algorithmica*, 38(3):433–439, 2003.

[30] Madhukar Korupolu, C. Greg Plaxton, and Rajmohan Rajaraman. Analysis of a local search heuristic for facility location problems. In *Proc. 9th Symp. on Discrete Algorithms (SODA)*, pages 1–10, 1998.

[31] Ravishankar Krishnaswamy and Maxim Sviridenko. Inapproximability of the multi-level uncapacitated facility location problem. In *Proc. 23th Symp. on Discrete Algorithms (SODA)*, pages 718–734, 2012.

[32] Alfred Kuehn and Michael Hamburger. A heuristic program for locating warehouses. *Management Science*, 9(4):643–666, 1963.

[33] Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. In *Proc. 38rd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 77–88, 2011.

[34] Kewen Liao and Hong Shen. Unconstrained and constrained fault-tolerant resource allocation. In *Proc. 17th Annual International Conference on Computing and Combinatorics (COCOON)*, pages 555–566, 2011.

[35] Jyh-Han Lin and Jeffrey Vitter. e-approximations with minimum packing constraint violation (extended abstract). In *Proc. 24th Symp. Theory of Computing (STOC)*, pages 771–782, 1992.

[36] Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay Vazirani. A greedy facility location algorithm analyzed using dual fitting. In *Proc. 4th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 127–137, 2001.

[37] Mohammad Mahdian, Yinyu Ye, and Jiawei Zhang. Approximation algorithms for metric facility location problems. *SIAM J. Comput.*, 36(2):411–432, 2006.

[38] Alan Manne. Plant location under economies-of-scale-decentralization and computation. *Management Science*, 11(2):213–235, 1964.

[39] Adam Meyerson. Online facility location. In *Proc. 42st Symp. Foundations of Computer Science (FOCS)*, pages 426–431, 2001.

[40] Martin Pál, Éva Tardos., and Tom Wexler. Facility location with nonuniform hard capacities. In *Proc. 42st Symp. Foundations of Computer Science (FOCS)*, pages 329–338, 2001.

[41] Richard Francis Pitu Mirchandani. *Discrete Location Theory.* Wiley-Interscience, 1990.

[42] David Shmoys. The design and analysis of approximation algorithms: Facility location as a case study. In *AMS Proceedings of Symposia in Applied Mathematics*, volume 61, pages 85–97, 2004.

[43] David Shmoys, Éva Tardos, and Karen Aardal. Approximation algorithms for facility location problems (extended abstract). In *Proc. 29th Symp. Theory of Computing (STOC)*, pages 265–274, 1997.

[44] David B. Shmoys. Approximation algorithms for facility location problems. In *Proc. 3th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 27–33, 2000.

[45] Maxim Sviridenko. An improved approximation algorithm for the metric uncapacitated facility location problem. In *Proc. 9th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 240–257, 2002.

[46] Chaitanya Swamy and David Shmoys. Fault-tolerant facility location. *ACM Trans. Algorithms*, 4(4):1–27, 2008.

[47] Jens Vygen. Approximation algorithms for facility location problems (lecture notes). Technical Report No. 05950, Research Institute for Discrete Mathematics, University of Bonn, 2005.

[48] Shihong Xu and Hong Shen. The fault-tolerant facility allocation problem. In *Proc. 20th International Symp. on Algorithms and Computation (ISAAC)*, pages 689–698, 2009.

[49] Li Yan and Marek Chrobak. Approximation algorithms for the fault-tolerant facility placement problem. *Inf. Process. Lett.*, 111(11):545–549, 2011.

[50] Jiawei Zhang. Approximating the two-level facility location problem via a quasi-greedy approach. *Math. Program.*, 108(1):159–176, 2006.

[51] Jiawei Zhang, Bo Chen, and Yinyu Ye. A multiexchange local search algorithm for the capacitated facility location problem. *Math. Oper. Res.*, 30(2):389–403, 2005.

Appendix A

Technical Background

A.1 Integer Programming and Linear Programming

In this section, we give a short introduction to Integer Programming and Linear Programming, with an emphasis on their application to the design and analysis of approximation algorithms for optimization problems. Figure A.1 gives an overview of the discussions below.

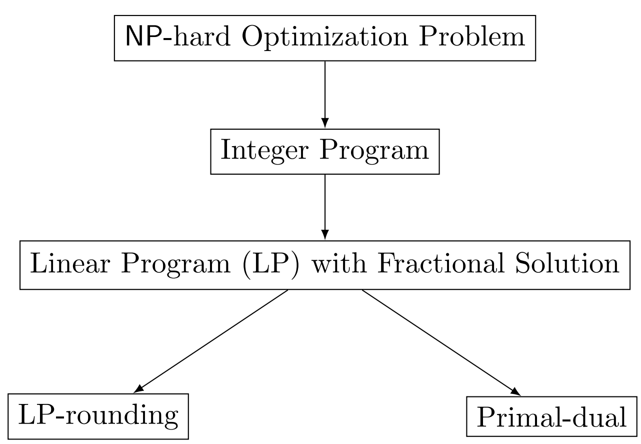


Figure A.1: An overview of application of Integer Programming and Linear Programming for NP-hard optimization problems.

A.1.1 Optimization and Integer Programming

Most optimization problems have a natural integer program in which we use variables to describe the solution that we seek, and write the constraints imposed by the feasibility requirements. The objective function is the cost function of the solution. Both the feasibility requirements and the cost function are specified by the problem. For example, in the Vertex Cover problem, we are given a graph $G = (V, E)$ and we are to find a subset W of V, such that every edge $e \in E$ has at least one endpoint in W; we want such a set W to have the minimum size. To formulate this problem as an integer program, we use $x_v \in \{0, 1\}$ to denote whether a node $v \in V$ is in W or not. The constraint is that, for every edge $e = (u, v)$, we have $x_u + x_v \geq 1$. The objective is to minimize $\sum_{v \in V} x_v$. The integer program for Vertex Cover is written as

$$\boxed{\begin{array}{lll} \text{minimize} & \displaystyle\sum_{v \in V} x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \forall (u, v) \in E \\ & x_v \in \{0, 1\} & \forall v \in V \end{array}}$$

In general, an integer program cannot be solved exactly in polynomial time, as Integer Programming is NP-hard. However, if we relax the integrality constraint and allow the variables to take fractional values, we then obtain a Linear Program (LP). LP is polynomially solvable, for example, using the ellipsoid method or the interior point method. Thus, we can first solve the LP optimally, obtaining a fractional optimal solution. The value of the fractional optimal solution is then a lower bound on the value of the integral optimal solution, assuming a minimization problem. Our next step is then to round the fractional solution appropriately, so that we maintain feasibility, while keeping the cost from increasing too much. The exact rounding procedure is problem specific and we shall not delve into this topic here. The rounding relevant to the FTFP problem is presented in detail in Chapter 5.

A.1.2 Linear Programming, Duality and Complementary Slackness Conditions

We now give a brief overview of linear programming; see [17] for an introductory book on this topic. A general Linear Program can be written as

$$\boxed{\begin{array}{lll} \text{minimize} & \displaystyle\sum_{j=1}^{n} c_j x_j & \\ \text{subject to} & \displaystyle\sum_{j=1}^{n} a_{ij} x_j \geq b_i, & \text{for } i = 1, \ldots, m \\ & x_j \geq 0 & \text{for } j = 1, \ldots, n \end{array}} \tag{A.1}$$

For the LP above, we can take its dual as

$$\boxed{\begin{array}{lll} \text{maximize} & \displaystyle\sum_{i=1}^{m} b_i y_i & \\ \text{subject to} & \displaystyle\sum_{i=1}^{m} a_{ij} y_i \leq c_j & \text{for } j = 1, \ldots, n \\ & y_i \geq 0 & \text{for } i = 1, \ldots, m \end{array}} \tag{A.2}$$

The LP (A.1) is called the primal program and the LP (A.2) is called the dual program. Regarding the objective function value of the two programs, we have the Weak Duality Theorem:

Theorem 26 *For every feasible solution* $\boldsymbol{x}$ *to the primal (A.1) and* $\boldsymbol{y}$ *to the dual (A.2), we have that* $\boldsymbol{c}^T\boldsymbol{x} \geq \boldsymbol{b}^T\boldsymbol{y}$.

The Strong Duality Theorem is that:

Theorem 27 *If both the primal (A.1) and the dual (A.2) are feasible, then both of them have optimal solution* $\boldsymbol{x}^*$ *and* $\boldsymbol{y}^*$ *and their objective function values are equal, that is* $\boldsymbol{c}^T\boldsymbol{x}^* = \boldsymbol{b}^T\boldsymbol{y}^*$.

One way to characterize optimal primal and dual solutions is the Complementary Slackness Conditions. The complementary slackness conditions says that:

Theorem 28 *Two feasible solutions* $\boldsymbol{x}$ *and* $\boldsymbol{y}$ *are both optimal to LP (A.1) and (A.2) respectively, if and only if, for every primal variable* x_j*, either* $x_j = 0$ *or the corresponding constraint in the dual is tight, that is* $\sum_{i=1}^{m} a_{ij}y_i = c_j$*; and for every dual variable* y_i*, either* $y_i = 0$ *or the corresponding constraint in the primal is tight, that is* $\sum_{j=1}^{n} a_{ij}x_j = b_i$.

The complementary slackness conditions provide a simple way to validate the optimality when one is presented with a primal solution and a dual solution that are claimed to be optimal. In addition, the complementary slackness conditions play a crucial role in the design and analysis of approximation algorithms. For example, suppose we have an algorithm that computes a feasible integral solution $\boldsymbol{x}$ to the primal program (A.1) and a feasible fractional solution to the dual program (A.2). Moreover, we know that the two solutions satisfy a relaxed version of the complementary slackness conditions, that for some numbers γ and ρ, we have

$$\begin{array}{llll} \text{either } y_i = 0 & \text{or} & b_i \le \sum_j a_{ij}x_j \le \gamma\, b_i, & \text{for } i = 1, \ldots, m; \\ \text{either } x_j = 0 & \text{or} & \rho\, c_j \le \sum_i a_{ij}y_i \le c_j, & \text{for } j = 1, \ldots, n. \end{array}$$

Then the integral solution $\boldsymbol{x}$ has a cost of not more than γ/ρ times the optimal value. In particular, we have $\sum_j c_j x_j \le \gamma/\rho \sum_i b_i y_i$ and the value of a feasible dual solution, namely

$\sum_i b_i y_i$, is a lower bound on the optimal value of the primal program.

To show an application of the complementary slackness conditions, we look at their use in the design and analysis of algorithms for the Uncapacitated Facility Location problem (UFL). Recall that we define the neighborhood $N(j)$ of a client j as the set of facilities with $x^*_{ij} > 0$, where $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is some fractional optimal solution to the LP (2.1) and $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$ is some optimal fractional dual solution to the LP (2.2). Given a client j, the complementary slackness conditions give an upper bound of α^*_j on d_{ij} for every facility $i \in N(j)$. To see this bound, notice that the dual constraint says $\alpha_j - \beta_{ij} \leq d_{ij}$. If the primal solution has $x^*_{ij} > 0$, then the inequality is actually an equality and we have $\alpha^*_j - \beta^*_{ij} = d_{ij}$. Together with $\beta^*_{ij} \geq 0$, we have $\alpha^*_j \geq d_{ij}$ for every facility i such that $x^*_{ij} > 0$. By definition, those are facilities in the neighborhood $N(j)$. Therefore we have $d_{ij} \leq \alpha^*_j$ for every $i \in N(j)$.

An important application of using the relaxed complementary slackness conditions is demonstrated by Jain and Vazirani [28] with their algorithm for the UFL problem. Their algorithm that outputs an integral solution $(\boldsymbol{x}, \boldsymbol{y})$ to the primal program (2.1) and a feasible (possibly fractional) solution $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ to the dual program (2.2). Moreover, the two solutions satisfy the conditions that

$$\begin{aligned}
&\text{either } \sum_j \beta_{ij} = f_i \quad \text{or} \quad y_i = 0;\\
&\text{either } (1/3)\, d_{ij} \leq \alpha_j - \beta_{ij} \leq d_{ij} \quad \text{or} \quad x_{ij} = 0.
\end{aligned}$$

The solution $(\boldsymbol{x}, \boldsymbol{y})$ is then a 3-approximation.

A.2 Proof of Inequality (5.3)

In Sections 5.2 and 5.3 we make use of the following inequality to derive the approximation ratio.

$$\bar{d}_1 g_1 + \bar{d}_2 g_2 (1-g_1) + \ldots + \bar{d}_k g_k (1-g_1)(1-g_2)\ldots(1-g_k) \\ \leq \frac{1}{\sum_{s=1}^{k} g_s} \left(\sum_{s=1}^{k} \bar{d}_s g_s\right) \left(\sum_{t=1}^{k} g_t \prod_{z=1}^{t-1}(1-g_z)\right). \tag{A.3}$$

for $0 < \bar{d}_1 \leq \bar{d}_2 \leq \ldots \leq \bar{d}_k$, and $0 < g_1, ..., g_s \leq 1$.

Here we give a new proof of this inequality, much simpler than the existing proof in [15], and also simpler than the argument by Sviridenko [45]. We derive this inequality from the following generalized version of the Chebyshev Sum Inequality:

$$\sum_i p_i \sum_j p_j a_j b_j \leq \sum_i p_i a_i \sum_j p_j b_j, \tag{A.4}$$

where each summation runs from 1 to l and the sequences (a_i), (b_i) and (p_i) satisfy the following conditions: $p_i \geq 0, a_i \geq 0, b_i \geq 0$ for all i, $a_1 \leq a_2 \leq \ldots \leq a_l$, and $b_1 \geq b_2 \geq \ldots \geq b_l$. Given the inequality (A.4), we can obtain our inequality (A.3) by simple substitution

$$p_i \leftarrow g_i, \quad a_i \leftarrow \bar{d}_i, \quad b_i \leftarrow \Pi_{s=1}^{i-1}(1-g_s),$$

for $i = 1, ..., k$.

For the sake of completeness, we include the proof of inequality (A.4), due to Hardy, Littlewood and Polya [24]. The idea is to evaluate the following sum:

$$
\begin{aligned}
S &= \textstyle\sum_i p_i \sum_j p_j a_j b_j - \sum_i p_i a_i \sum_j p_j b_j \\
&= \textstyle\sum_i \sum_j p_i p_j a_j b_j - \sum_i \sum_j p_i a_i p_j b_j \\
&= \textstyle\sum_j \sum_i p_j p_i a_i b_i - \sum_j \sum_i p_j a_j p_i b_i \\
&= \textstyle\frac{1}{2} \cdot \sum_i \sum_j (p_i p_j a_j b_j - p_i a_i p_j b_j + p_j p_i a_i b_i - p_j a_j p_i b_i) \\
&= \textstyle\frac{1}{2} \cdot \sum_i \sum_j p_i p_j (a_i - a_j)(b_i - b_j) \leq 0.
\end{aligned}
$$

The last inequality holds because $(a_i - a_j)(b_i - b_j) \leq 0$, since the sequences (a_i) and (b_i) are ordered oppositely.