LP-based Approximation Algorithms for the Capacitated Facility Location Problem.

by

Marco David Blanco Sandoval

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

Waterloo, Ontario, Canada, 2012

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

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

Abstract

The capacitated facility location problem is a well known problem in combinatorial optimization and operations research. In it, we are given a set of clients and a set of possible facility locations. Each client has a certain demand that needs to be satisfied from open facilities, without exceeding their capacity. Whenever we open a facility we incur in a corresponding opening cost. Whenever demand is served, we incur in an assignment cost; depending on the distance the demand travels. The goal is to open a set of facilities that satisfy all demands while minimizing the total opening and assignment costs.

In this thesis, we present two novel LP-based approximation algorithms for the capacitated facility location problem.

The first algorithm is based on LP-rounding techniques, and is designed for the special case of the capacitated facility location problem where capacities are uniform and assignment costs are given by a tree metric.

The second algorithm follows a primal-dual approach, and works for the general case.

For both algorithms, we obtain an approximation guarantee that is linear on the size of the problem. To the best of our knowledge, there are no LP-based algorithms known, for the type of instances that we focus on, that achieve a better performance.

Acknowledgements

I would first like to thank my supervisor Ricardo Fukasawa, for making my stay in this university possible, for all the time he dedicated to me, and for all his valuable comments and ideas.

I also thank Jochen Könemann, who collaborated with us through most of the research that led to this thesis. His final comments as a reader were very useful.

I am equally thankful to Zac Friggstad, who joined our research team later and contributed with some great ideas for the design and analysis of the algorithm in Chapter 4.

Finally, I thank the second reader, Chaitanya Swamy, for his valuable comments.

Table of Contents

Chapter 1

Introduction and Preliminaries

In this chapter, we start by recalling some basic definitions and results in optimization and linear programming. Thus, this first section can be safely skipped by a reader who is familiar with these topics. Then, we introduce the capacitated facility location problem, and give a short review of the background in finding approximation algorithms for it. We conclude by explaining the motivation for choosing this topic, and describing the outline of this thesis.

1.1 Basic Concepts

Let us consider a feasible, bounded mixed integer program (MIP) of the following general form:

$$(P_I) \qquad \begin{array}{rl} \min & c^T x \\ \text{s.t} & Ax \geq b \\ & x \in \mathbb{Z}^r \times \mathbb{R}^{n-r} \\ & x \geq 0, \end{array}$$

where $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^m$, $c \in \mathbb{Q}_+$ and $0 \leq r \leq n$. Let z^* be its optimal value. Note that by nonnegativity of c and x, we have $z^* \geq 0$. This will later be important when we analyze the performance of our algorithms.

Let (P) be the relaxation we obtain by removing the integrality constraints on x. That is,

$$(P) \qquad \begin{aligned} \min \quad & c^T x \\ \text{s.t} \quad & Ax \geq b \\ & x \in \mathbb{R}^n \\ & x \geq 0. \end{aligned}$$

Let z_P^* be the its optimal value. Since (P) is a relaxation of (P_I), which is a minimization problem, we know that $z_P^* \leq z^*$. By the assumption on c made above, we have that z_P^* is also nonnegative.

The *dual* of (P) is defined as

$$(D) \qquad \begin{aligned} \max \quad & b^T y \\ \text{s.t} \quad & y^T A \leq c \\ & y \in \mathbb{R}^m \\ & y \geq 0, . \end{aligned}$$

Let z_D^* be its optimal value. The property given by the following theorem is called *weak duality*, and is a very important and classic result in linear programming.

Theorem 1.1.1 *Let* (P), (D), z_P^* *and* z_D^* *be as above. Then,*

$$z_D^* \leq z_P^*.$$

One of the many uses of weak duality is in the design and analysis of primal-dual algorithms, as we shall see in Chapter 4. The proof of this theorem is omitted, it can be found in almost any linear programming book.

We define the (multiplicative) *integrality gap* of the problem as $\max\limits_{(P)\in\mathcal{I}}\{\frac{z^*}{z_P^*}\}$, where $\mathcal{I}$ is the set of instances of the problem. By weak duality, this is at most $\frac{z^*}{z_D^*}$. Let us see why the integrality gap is important to us.

We say an algorithm (which takes (P_I) as input) has an *approximation guarantee of* α (we also say it is an α-*approximation algorithm*) if it runs on polynomial time and outputs a feasible solution (for P_I) of value z such that $z \leq \alpha z^*$. Here, α may depend on the size of the input, but we omit that from the notation. If α is independent of the instance and its size, we call it a *constant factor* approximation algorithm. As the name says, *LP-based approximation algorithms* are algorithms that use linear programming techniques in their design and/or analysis. These concepts are of particular importance to our work, because we will describe and analyze LP-based approximation algorithms in the three main chapters.

By the relation between z^* and z_P^* noticed above, if an algorithm gives us a feasible solution of value z such that $z \leq \alpha z_P^*$, we know that it is an α-approximation algorithm. By weak duality, an algorithm satisfying $z \leq \alpha z_D^*$ is also an α-approximation algorithm. In general, LP-based approximation algorithms prove the approximation guarantee by using one or both of these properties. Clearly, the approximation guarantee of such an LP-based algorithm cannot be smaller than the integrality gap of the problem. In particular, if the integrality gap of a relaxation is unbounded, a constant factor approximation on these lines is not possible. As we will see, this property is what makes it so difficult to find good LP-based approximation algorithms for the capacitated facility location problem.

In this work, we will focus on two types of LP-based approximation algorithms. The first type are *LP-rounding algorithms*. These take an optimal (or just feasible) solution to an LP relaxation and round it to a feasible solution to the original problem. The algorithms we describe in Chapters 2 and 3 belong to this category. The second type are *primal-dual* algorithms. These iteratively construct pairs of feasible dual solutions and infeasible primal solutions. As the algorithm progresses, the cost of each solution increases; and the algorithm stops when the primal solution becomes feasible. An advantage of primal-dual algorithms over LP-rounding algorithms is that they do not require to solve the LP relaxation. In particular, even if the LP cannot be solved in polynomial time because of the number of constraints is too large, one may still design an efficient primal-dual algorithm. We will work with primal-dual algorithms in Chapter 4, and we will make use of this property.

A disadvantage of the relaxations we will introduce in the following chapters is that they have an exponential number of constraints, which makes them hard to solve to optimality. However, we can make use of the equivalence between separation and optimization over polyhedra, which is another classical result in mathematical optimization. For that, we make the following definitions,

Let $K \in \mathbb{R}^n$ be a rational, bounded polyhedron; and let $c \in \mathbb{Q}^n$.

The *optimization problem* corresponding to c and K is the following:

- Find $y \in K$ maximizing $\{c^T x | x \in K\}$ or show $K = \emptyset$.

The *separation problem* corresponding to K is the following:

- Given $\bar{x} \in \mathbb{R}^n$, is $\bar{x} \in K$? If not, find $\alpha \in \mathbb{R}^n$, $\beta \in \mathbb{R}$ such that $\alpha^T \bar{x} > \beta$ and $\alpha^T x \leq \beta$ for every $x \in K$.

The next theorem is due to Grötschel, Lovász and Schrijver, and can be found in [7]. The theorem is valid for more general sets K, but this simpler version suffices for us.

Theorem 1.1.2 *Given c and K as above, the separation problem can be solved in polynomial time if and only if the optimization problem can be solved in polynomial time.*

As mentioned above, the importance of this result for us is that if the separation problem corresponding to an LP can be solved efficiently (as will be the case in Chapters 2 and 3) for every point in $\mathbb{R}^n$, then the LP can be solved to optimality in polynomial time, even if the number of constraints is exponential.

1.2 The Capacitated Facility Location Problem

We are now ready to introduce the *capacitated facility location* problem, which is the main problem we will focus on in this work. It is defined as follows:

Consider a set of facilities $\mathcal{F}$ and a set of clients $\mathcal{C}$. Each facility i has an opening cost f_i, and a capacity u_i. Every client j has a demand d_j that needs to be satisfied from open facilities. The cost of transporting a unit of product from facility i to client j is given by c_{ij}. A facility cannot operate unless its opening cost is paid, and once it is open, it cannot exceed its capacity. The goal is to satisfy all demands while minimizing the total cost.

We will assume that demands are splittable. This means that a single facility can be served by several clients.

This problem is of a great importance to operations research, since it can be used to model a huge number of real-life problems, ranging from telecommunications to mail delivery or supply chain management. Besides that, it is also of great interest in combinatorial optimization, and there are still several open questions concerning it.

As with most optimization problems, there are several variants of this problem that have been considered in the literature. In general, we will work with *metric* assignment costs and *hard* capacities. By this, we mean that the facilities and clients are points in a metric space with distances given by the assignment costs; and that we can open each facility at most once (as opposed to the *soft* capacities case, where we are allowed to open several copies of each facility).

So, unless otherwise specified, we will make these assumptions throughout the text, and we will refer to the problem as CFL.

We can easily formulate this program as an MIP. For every $i \in \mathcal{F}, j \in \mathcal{C}$, let x_{ij} denote the fraction of the demand d_j satisfied from facility i; and let y_i equal 1 when facility i is open, and 0 otherwise. Then, the formulation is the following:

$$
\text{(MIP-CFL)} \quad
\begin{array}{lll}
\min & \sum_{i\in\mathcal{F}} f_i y_i + \sum_{i\in\mathcal{F}}\sum_{j\in\mathcal{C}} d_j x_{ij} & \\
\text{s.t} & \sum_{i\in\mathcal{F}} x_{ij} \geq 1 & \text{for every } j \in \mathcal{C}, \quad (1.1)\\
 & \sum_{j\in\mathcal{C}} d_j x_{ij} \leq u_i y_i & \text{for every } i \in \mathcal{F}, \quad (1.2)\\
 & y_i \leq 1 & \text{for every } i \in \mathcal{F},\\
 & y_i \geq 0 & \text{for every } i \in \mathcal{F},\\
 & x_{ij} \geq 0 & \text{for every } i \in \mathcal{F}, j \in \mathcal{C},\\
 & y_i \in \mathbb{Z} & \text{for every } i \in \mathcal{F}.
\end{array}
$$

Here, constraints (1.1) tell us that all demands must be satisfied. Constraints (1.2) say that the total amount of demand served by a facility i cannot exceed its capacity.

In the literature, the formulation of this problem often includes the following set of constraints as well:

$$x_{ij} \leq y_i \qquad \text{for every } i \in \mathcal{F}, j \in \mathcal{C}.$$

These constraints say that a facility cannot serve demand unless it is open. They are trivially implied by (1.2) for feasible points. One reason why it makes sense to include them is that they are essential in the formulation of the uncapacitated facility location problem. Another reason is that they are not redundant after removing the integrality constraint. However, they are not necessary for our analysis, so we will ignore them.

Throughout the work, we will refer to the LP obtained by relaxing the integrality constraints of (MIP-CFL) as the *natural* LP relaxation of (MIP-CFL). This convention is useful because later we will consider more sophisticated relaxations and we will compare them to this one. As we will see now, the natural LP relaxation of the capacitated facility location problem has an arbitrarily large integrality gap, which is why we will need to close that gap by considering stronger relaxations. In fact, as the example will show, the gap remains even if we add the constraints $x_i \geq y_i$.

Consider an instance with two facilities and one client. Let $f_1 = M$, $f_2 = 0$, $u_1 = M$, $u_2 = M$, $d_1 = M + 1$ and $c_{11} = c_{21} = 0$. In other words, one facility can be built for free while the other is very expensive; and each one has a little less capacity than necessary to satisfy the whole demand. Clearly, all feasible solutions (in particular the optimal solutions) open both facilities, so the optimal solution has cost M. On the other hand, the optimal solution of the natural LP relaxation is the following: $y_1 = \frac{1}{M}$, $y_2 = 1$, $x_{11} = \frac{1}{M+1}$, $x_{21} = \frac{M}{M+1}$. That is, we open the free facility completely and the expensive one partially, just enough to satisfy the demand. This solution has cost 1. So, the integrality gap of the problem is at least $\frac{M}{1} = M$. We can make M arbitrarily large, so the integrality gap of the natural LP relaxation of (MIP-CFL) is unbounded. Since we know that the integrality gap is a lower bound on the approximation guarantee of any LP-based algorithm, this tells us that it is impossible to get a constant factor approximation algorithm based on the natural relaxation of (MIP-CFL).

1.2.1 Review of Previous Results

CFL is a generalization of the facility location problem (where all facilities have an infinite capacity), which has been extensively studied. The facility location problem is not only NP-hard; it is known that, unless each problem in NP can be solved in $O(n^{O(\log \log n)})$ time, then we cannot find an approximation algorithm for the facility location problem with a guarantee better than 1.463. This can be done via a reduction from the unweighted set cover problem. The original proof is due to Guha and Khuller ([8]), it can also be found in [20].

Since facility location is a particular case of CFL, this hardness result is also valid for the problem we will study.

The first constant factor approximation for the metric CFL problem with hard capacities was given by Korupolu et al. ([11]) in 1998, for the special case of uniform capacities (which we will also consider, in Chapters 2 and 3). This was improved in 1999 to a guarantee of 5.83 by Chudak & Williamson in [4], for the same special case. The state of the art for uniform capacities is a 3-approximation by Aggarval et al., in [2].

The first such algorithm for the case of non-uniform capacities was given in 2001 by Pál et al. ([15]). This was subsequently improved by Mahdian and Pál ([14]) and then by Zhang et al. ([21]) in 2004, to match the 5.83 guarantee of the uniform capacities case. This is currently the best guarantee known.

All the results mentioned above are based on local search algorithms. The only efficient LP-based algorithm known for this problem is due to Levi, Shmoys and Swamy ([13]). This algorithm uses LP-rounding and clustering techniques and gives a 5-approximation for the case of uniform opening costs. This last assumption is very important for the analysis of their algorithm, since they use the natural LP relaxation of (MIP-CFL), which (as we have seen) has an unbounded integrality gap in the more general case.

1.3 Motivation

As mentioned before, all known approximation algorithms for the general case are based on local search. It is thus natural to wonder whether a constant factor LP-based approximation algorithm can be found for the general case. Beside the desire to satisfy this curiosity, LP-based algorithms have some advantages over local search algorithms.

One of these advantages is that LP-based algorithms are in general easy to implement (although they may have long running times). The second is that, while local search algorithms offer an *a priori* approximation guarantee, the guarantee given by LP-based algorithms is *a fortiori*. Let us elaborate on this:

Given a local search α-approximation algorithm for a minimization problem, let z be the value returned. If z^* is the optimal value, the only thing we know is $z^* \leq z \leq \alpha z^*$; but since in general we do not know the value z^*, there is no way of knowing where z lies in the interval $[z^*, \alpha z^*]$.

On the other hand, suppose an LP-based α-approximation algorithm for the same problem outputs a solution with value $\bar{z}$. In general, we can find the optimal value of the LP relaxation easily, suppose in this case it is z^*_{LP}. We know $z^*_{LP} \leq z^* \leq \bar{z} \leq \alpha z^*_{LP} \leq \alpha z^*$. Now, we can easily compute $\beta = \dfrac{\bar{z}}{z^*_{LP}} \leq \alpha$; and by the chain of inequalities above, we can say that the algorithm actually gives us a β-approximation for this instance.

This does not necessarily mean that the LP-based algorithm will have a better performance than the local search algorithm; but we have a better performance measure for it.

Finding this constant factor LP-based algorithm (or even an LP relaxation with a bounded integrality gap) is one of the most prominent open problems in the field of approximation algorithms (see the open problems section in [20]).

Because the integrality gap of the standard LP relaxation of the problem is unbounded, the only way of obtaining an LP-based constant factor approximation algorithm must be closing this gap through strengthening the LP relaxation with valid inequalities for (MIP-CFL).

This work was mostly motivated by the algorithm that Levi, Lodi and Sviridenko presented in [12]. In this paper, which we will review in detail in Chapter 2, the authors design an LP-rounding algorithm for the lot-sizing problem that gives a 2-approximation. As we will see, this problem is very closely related to CFL, and it even shares the problem of having a bad integrality gap.

The question of whether the techniques used in that paper could be adapted to CFL is what inspired this research.

1.4 Outline and Results

In Chapter 2, we define the multi-item lot-sizing problem with uniform capacities, and we present in detail the results in [12], by Retsef Levi, Andrea Lodi and Maxim Sviridenko; where the authors give a 2-approximate LP-rounding algorithm for this problem. This algorithm uses a family of inequalities, known as flow-cover, to obtain an LP relaxation with a bounded integrality gap; and is one of the main inspirations behind our work.

In Chapter 3, we consider the problem of facility location with uniform capacities and assignment costs corresponding to a tree metric, and generalize the techniques of Levi et al. to obtain an approximation algorithm for this special case. Our approximation guarantee is constant when the number of leaves of the tree (or the number of facilities) is upper-bounded by a constant. We show that the same algorithm is valid for a slightly more general framework, which we can prove to be strongly NP-hard.

In Chapter 4, we start by presenting a result by Tim Carnes and David Shmoys ([3]), where they give a primal-dual algorithm for the single-client capacitated facility location problem, which is based on a different type of flow-cover inequalities. Then, we generalize these techniques and obtain a primal-dual approximation algorithm for the capacitated facility location problem. The approximation guarantee of this algorithm is constant if the number of clients is upper-bounded by a constant.

Then, in Chapter 5 we present the conclusions to our work.

Finally, in the Appendix, we prove some basic results that are needed for Chapters 2 and 3 but are not included in these chapters. More specifically, we prove that flow-cover inequalities can be separated in polynomial time under some conditions; and we prove NP-hardness of the multi-item lot-sizing problem.

Chapter 2

An LP-Rounding Algorithm for the Multi-Item Lot-Sizing Problem with Uniform Capacities

The first approach we take to find approximation algorithms for the CFL problem is inspired on [12]. In this paper, the authors develop an LP-rounding algorithm for the *multi-item lot-sizing problem with uniform capacities*. As we will see, this problem is very similar to CFL. It also shares the issue of having a natural LP formulation with an unbounded integrality gap, which is why the result in the paper is so important. To solve this issue, the authors strengthen the natural LP relaxation by adding a set of inequalities called *flow-cover* to it.

In this chapter, we will present the results in [12]. First, we will introduce the problem and review its computational complexity. Next, we define flow-cover inequalities, and show how certain subsets of them can be separated in polynomial time. Then, we describe a randomized algorithm based on the strengthened LP relaxation we obtain by adding these inequalities. Finally, we present the analysis of the algorithm to show it gives a 2-approximation guarantee. We also shortly describe a derandomization of the algorithm and an *on the fly* variant of it.

2.1 The Problem

The problem we will study is the following. We consider N types of items, and a set of T time periods. For every item of type i and every time period t, there is a demand $d_{it} \in \mathbb{R}_+$ of such items that needs to be satisfied in the given period (we will call (i, t) a *demand point*). At the beginning of every period s, we may choose to pay a fixed ordering cost $f_s \in \mathbb{R}_+$, that will allow us to "open" the order corresponding to that period. Each order has an associated capacity $u_s \geq 0$; that is, the maximum number of items we can produce in that time period if we open the order (we incur in no extra cost for ordering items, just for opening the order). This capacity bounds the total number of items ordered at this time point, regardless of their type. If we order an item in some period s, we may choose to use it to satisfy a demand in the same period s, or hold it to satisfy a demand later in time. However, by doing that we incur in a holding cost. The cost for holding a unit of item i from period s to period t (with $s < t$) is $c^i_{st} \in \mathbb{R}_+$ (if we use an item to satisfy a demand right in the moment it was ordered, we pay no holding cost). As in capacitated facility location, we will assume that demand points can be served by several orders. The objective is to satisfy all the demands at a minimal cost. After describing the LP-formulation, we will give a small example.

In this chapter, we will focus on the case of uniform capacities (That is, $u_s = u$ for every s) and monotone holding costs (c^i_{st} is non-increasing in s for fixed i, t). So, when we refer to the lot-sizing problem, we will mean this special case.

The following notation conventions will be useful. Let $s, t \in \mathbb{Z}_+$ with $s \leq t$. Since we will be dealing with discrete time events, by $[s, t]$ we mean $\{s, s+1, \ldots, t-1, t\}$, by $(s, t]$ we mean $\{s+1, \ldots, t-1, t\}$, and similarly for other intervals in $\mathbb{R}$.

2.1.1 Complexity

Just like for CFL, several variants of this problem (uniform vs. non-uniform capacities/costs/demands, single-item vs. multi-item, monotone vs. linear vs. concave vs. general holding costs, etc.) have been studied in the literature, and in general they belong to different complexity classes.

Just to mention some examples, the case with a single item ($N = 1$) and non-uniform capacities is weakly NP-hard, but there exists a FPTAS for it (see [9]). The single-item case with uniform capacities can be solved in polynomial time (See [5]).

However, the variant we will be working on; i.e., multi-item lot-sizing with uniform capacities; is strongly NP-hard. The proof of this is given in Appendix A.

2.1.2 Formulation

A natural MIP formulation for the multi-item lot-sizing problem with uniform capacities is the following:

$$
\text{(MIP-LS)} \quad
\begin{aligned}
\min \quad & \sum_{s=1}^{T} f_s y_s + \sum_{i=1}^{N} \sum_{t=1}^{T} d_{it} \sum_{s=1}^{t} c_{st}^i x_{st}^i \\
\text{s.t} \quad & \sum_{s \le t} x_{st}^i = 1 && \forall i = 1, \dots, N;\ t = 1, \dots T, && (2.1) \\
& \sum_{i=1}^{N} \sum_{t \ge s} d_{it} x_{st}^i \le u y_s && \forall s = 1, \dots, T, && (2.2) \\
& x_{st}^i \ge 0 && \forall i = 1, \dots, N; s, t = 1, \dots, T; s \le t, \\
& y_s \in \{0, 1\} && \forall s = 1, \dots, T.
\end{aligned}
$$

Here, the variables y_s are indicator variables whose value is 1 if we open an order in period s and 0 otherwise. x_{st}^i is the fraction of the demand d_{it} that is satisfied by an order placed in period s. That is, $d_{it} x_{st}^i$ is the amount of demand of item i in period t served from order s. We define x_{st}^i **only** for $s \le t$. We sometimes include $s \le t$ in the restrictions to emphasize that we can only serve demand forward in time, but this is not necessary.

As we can see, this formulation is almost identical to (MIP-CFL).

Similarly to what happens with CFL, the natural LP relaxation of this MIP has an unbounded integrality gap. The example that shows this is essentially the same one we presented in the introduction. That is, let $M > 1$, and consider an instance with $T = 2$, $N = 1$, $d_1^1 = 0$, $d_2^1 = M + 1$, $u = M$, $f_1 = 0$, $f_2 = M$, $c_{12}^1 = 0$. The reasoning is then analogous to what we did in the introduction.

2.2 Strengthening the Formulation

Knowing that the formulation described above is weak, the goal is to find inequalities that are valid for the integer solutions but cut off unwanted fractional solutions. We will do that by using inequalites called *flow-covers*.

2.2.1 Flow-Cover Inequalities

Before stating the inequalities, we make the following definitions. Given a subset of demand points $A \subseteq \{(i,t) | i = 1, \ldots, N; t = 1, \ldots, T\}$, we define

$$\begin{aligned} D(A) &= \sum_{(i,t) \in A} d_{it}, \\ \ell_A &= \lceil D(A)/u \rceil, \\ R_A &= D(A) - (\ell_A - 1)u, \\ r_A &= R_A/u. \end{aligned}$$

These parameters will later be key in the analysis of the algorithm. Intuitively, $D(A)$ is the total demand of the demand points in A. ℓ_A is the minimum number of orders we need to open to satisfy the demand in A. R_A is the demand in A we still have to satisfy if we open one less order than strictly necessary and completely use the open orders. Note that $0 < R_A \leq u$, making $0 < r_A \leq 1$.

Proposition 2.2.1 *Let A be a set of demand points and F a set of orders. The following inequality is valid for all feasible points of* (MIP-LS).

$$D(A) - \sum_{(i,t) \in A} \sum_{s \in F} d_{it} x^i_{st} \geq R_A \left(\ell_A - \sum_{s \in F} y_s \right). \tag{2.3}$$

Proof: The authors of [12] prove the validity of (2.3) by showing it can be derived from an MIR (Mixed Integer Rounding) inequality. Here, we will give a simpler and more intuitive proof.

Let us reformulate inequality (2.3) as follows.

$$\sum_{s \in F} y_s \geq \ell_A - \frac{\sum_{(i,t) \in A} \sum_{s \notin F} d_{it} x^i_{st}}{R_A}.$$

Here, we used the definition of $D(A)$ and constraint (2.1).

Let (x, y) be a feasible point. If $\sum_{s \in F} y_s \geq \ell_A$, then the inequality is trivially satisfied. Otherwise, we have $\sum_{s \in F} y_s = \ell_A - k$, for some $k \in \{1, \ldots, \ell_A\}$. This means that there are exactly $\ell_A - k$ open orders in F. Therefore, these orders are serving at most $(\ell_A - k)u$ units of demand. Since we need a total of $(\ell_A - 1)u + R_A$ units to satisfy all the demand in A (by definition of R_A), then at least $u(k-1)+R_A$ of them will have to come from orders outside F.

That is,

$$\sum_{(i,t)\in A}\sum_{s\notin F} d_{it}x^i_{st} \geq u(k-1)+R_A \geq R_A(k-1)+R_A = R_A k.$$

Thus,

$$-k \geq -\frac{\sum_{(i,t)\in A}\sum_{s\notin F} d_{it}x^i_{st}}{R_A},$$

and adding ℓ_A on both sides, the inequality we want follows. ■

From this proof, we get the following observation, which will be useful to prove separation:

Lemma 2.2.2 *Let (x,y) be a fractional solution to the LP relaxation of* (MIP-LS) *that violates a flow-cover inequality for some sets A and F. Then,*

$$\ell_A - 1 < \sum_{s\in F} y_s < \ell_A. \tag{2.4}$$

Proof: In the previous proposition, we proved that if y is feasible for (MIP-LS) and does not satisfy (2.4), then it satisfies the flow-cover inequality. Here, we will see that fractional solutions of the natural LP relaxation of (MIP-LS) that do not satisfy (2.4), do satisfy the flow-cover inequality. The proof is analogous to the previous one. We only need to check that the integrality of the y_s variables or k is not needed in the argument.

As before, if $\sum_{s\in F} y_s \geq \ell_A$, then the inequality is clearly satisfied. Now, if $\sum_{s\in F} y_s \leq \ell_A - k$ (where $k \geq 1$ but not necessarily an integer), we can no longer argue that there are $\ell_A - k$ open orders in $\mathcal{F}$. However, we know that the total capacity available is at most $u(\ell_A - k)$, and the rest follows as before. ■

To the best of our knowledge (and to the best of the knowledge of the authors of [12] as well), the complexity of separating flow-cover inequalities is unknown. However, the next theorem (of which the proof will be given in Appendix B) gives a partial result, which will be enough for our purposes.

Theorem 2.2.3 *For a fixed set of orders $\boldsymbol{F}$, the inequalities* (2.3) *can be separated in time $O(NT^2)$.*

The importance of this theorem is that if we choose a polynomial (in T) number of sets of orders and add all the corresponding flow-cover inequalities to the LP relaxation, by the equivalence of separation and optimization stated in the introduction, we are going to be able to solve the LP in polynomial time, even though the number of constraints is exponential.

Keeping this in mind, let $\mathcal{I} := \{[s,t] : 1 \leq s \leq t \leq T\}$ be the collection of sets of orders corresponding to intervals $\{s, s+1, \ldots, t-1, t\}$, and let $C(A)$ be the set of covers of A for every A. Let us consider the following LP:

$$
\text{(LP-LS)} \qquad
\begin{array}{lll}
\min & \displaystyle\sum_{s=1}^{T} f_s y_s + \sum_{i=1}^{N}\sum_{s=1}^{T}\sum_{t=s}^{T} c_{st}^i d_{it} x_{st}^i & \\
\text{s.t} & \displaystyle\sum_{s\leq t} x_{st}^i \geq 1 & \forall i = 1, \ldots, N;\ t = 1, \ldots T \\
& \displaystyle\sum_{i=1}^{N}\sum_{t\geq s} d_{it} x_{st}^i \leq u y_s & \forall s = 1, \ldots, T, \\
& \displaystyle D(A) - \sum_{(i,t)\in A}\sum_{s\in F} d_{it} x_{st}^i \geq R_A \left(\ell_A - \sum_{s\in F} y_s\right) & \forall F \in \mathcal{I}, A \text{ s.t. } F \in C(A). \\
& x_{st}^i \geq 0 & \forall i = 1, \ldots, N, \\
& 0 \leq y_s \leq 1 & \forall s = 1, \ldots, T.
\end{array}
$$

That is, we look at the LP relaxation we had before and add all flow-cover inequalities corresponding to sets of orders in $\mathcal{I}$. Note that $|\mathcal{I}| = O(T^2)$. Thus, by Theorem 2.2.3, this new LP relaxation can be solved in polynomial time.

2.3 The Algorithm

In this section, we describe the algorithm. First we solve (LP-LS). Let $(\widehat{x}, \widehat{y})$ be an optimal solution. We will round $(\widehat{x}, \widehat{y})$ to a feasible solution $(\tilde{x}, \tilde{y})$ of (MIP-LS).

The algorithm runs in two phases. In the first phase (called random-shift procedure), we round $\widehat{y}$, and in the second phase (median assignment procedure) we round $\widehat{x}$.

2.3.1 The Random-Shift Procedure

In this phase, we just focus on $\widehat{y} = (\widehat{y}_1, \dots, \widehat{y}_T)$, and describe how to derive from this fractional vector an integral solution that represents a suitable choice of periods in which orders will be placed.

For every s, let $\overline{y}_s := \min\{2\widehat{y}_s, 1\}$. Let $W = \left\lceil \sum_{s=1}^{T} \overline{y}_s \right\rceil$, and α a random number chosen uniformly in $(0, 1]$. For every period $r \in \{1, \dots, T\}$, there is either none or exactly one number of the form $w + \alpha$ in the interval $(\sum_{s=1}^{r-1} \overline{y}_s, \sum_{s=1}^{r} \overline{y}_s]$, with $w \in \{0, 1, \dots, W\}$. Let $r_1, \dots, r_Q$ be the orders such that there is such a point in the corresponding interval, and such that $r_1 < r_2 < \dots < r_Q$. We call them the *opened orders*. We define $\tilde{y}_{r_m} = 1$ for every $m = 1, \dots, Q$, and $\tilde{y}_s = 0$ for every other component of $\tilde{y}$.

To make the algorithm clear, let us look at an example.
Say we we get $\widehat{y} = (0.2, 0.4, 0.9, 0.5, 0.1, 0.15)$ as an optimal solution to (LP-LS). Then, by definition, $\bar{y} = (0.4, 0.8, 1, 1, 0.2, 0.3)$. Now, if $\alpha = 0.3$, for example, the algorithm is going to output $\tilde{y} = (1, 0, 1, 1, 1, 0)$. That is, orders 1, 3, 4 and 5 are opened, orders 2 and 6 are kept closed. On the other hand, if $\alpha = 0.9$, we get $\tilde{y} = (0, 1, 1, 1, 0, 0)$. We can see it graphically in Figure 2.1 and Figure 2.2, respectively.

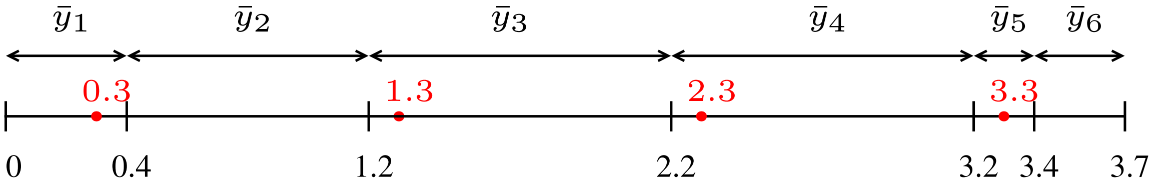

Figure 2.1: Random-shift procedure for $\widehat{y} = (0.2, 0.4, 0.9, 0.5, 0.1, 0.15)$ with $\alpha = 0.3$. The intervals that contain a point of the form $w + \alpha$ (with $w \in \mathbb{Z}$) correspond to $\bar{y}_1, \bar{y}_3, \bar{y}_4$ and $\bar{y}_5$.

The following lemma gives us a bound on the cost incurred by this rounding.

Lemma 2.3.1 *The total expected ordering cost of the solution obtained by the random-shift procedure is at most* $2\sum_{s=1}^{T} \widehat{y}_s f_s$.

Proof: Clearly, the probability of placing an order in period s is exactly $\overline{y}_s$, which by definition is at most $2\widehat{y}_s$. Let K be the ordering cost of the rounded solution. We have:

$$\mathbb{E}(K) = \sum_{s=1}^{T} \mathbb{P}(\tilde{y}_s = 1) f_s = \sum_{s=1}^{T} \bar{y}_s f_s \leq \sum_{s=1}^{T} 2\widehat{y}_s f_s = 2\sum_{s=1}^{T} \widehat{y}_s f_s.$$

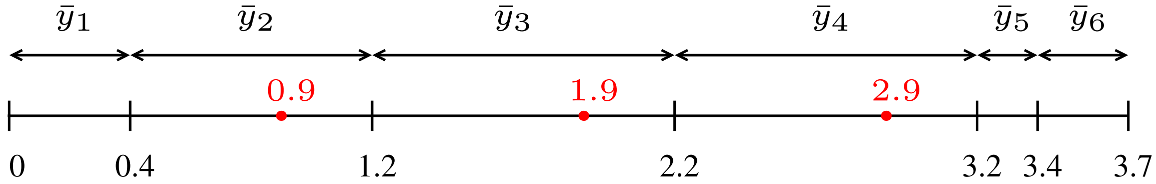

Figure 2.2: Random-shift procedure for $\widehat{y} = (0.2, 0.4, 0.9, 0.5, 0.1, 0.15)$ with $\alpha = 0.9$. The intervals that contain a point of the form $w + \alpha$ (with $w \in \mathbb{Z}$) correspond to $\bar{y}_2, \bar{y}_3$ and $\bar{y}_4$.

■

This randomized rounding technique is very similar to the standard randomized rounding used, for example, in the well known randomized approximation algorithm for the set cover problem (see [19], section 14.2). While the basic idea is that we open an order s with probability $2\widehat{y}_s$ (or 1 if this number exceeds 1), the main difference here is that the events $\{$*order s is opened*$\}$ are not independent. As seen above, this is not a problem when dealing with the objective function, thanks to the linearity of expectation.
On the other hand, this will result in a very powerful tool as we will see below. Intuitively, by doing this kind of rounding, we are making sure that if there is a group of consecutive orders whose $\widehat{y}$-values are not necessarily large but add up to something "large" (for example 1/2), then with high probability (or probability 1 if the sum exceeds 1/2) we are going to open at least one order from the group. Thus we make sure that for every demand point, with probability 1 there is an open order "not too far away" from it that can serve it. These notions will be formalized in the analysis of the algorithm.

2.3.2 The Median Assignment Procedure

In the second phase of the algorithm, we will work with the fixed set of open orders obtained in the first phase. We will construct $\tilde{x}$ such that $(\tilde{x}, \tilde{y})$ is a feasible solution and we will prove that the total holding cost of this new solution is at most twice the total holding cost paid by the fractional solution $\widehat{x}$ (more precisely, we will first construct a "solution" whose cost is clearly bounded, and then we will prove it is feasible). Along with Lemma 2.3.1, this will imply a 2-approximation.

We construct $\tilde{x}$ as follows.

Let (i, t) be a demand point, and let $s_1 < \cdots < s_G$ be the (possibly) fractional orders that serve this demand point in the optimal LP solution $(\widehat{x}, \widehat{y})$, that is, $\widehat{x}^i_{s_g,t} > 0$ for every

$g = 1, \ldots, G$, and $\sum_{g=1}^{G} \widehat{x}^i_{s_g,t} = 1$. Let s_M be the *median order* of (i,t), that is, the latest point in time such that at least half of the demand d_{it} is satisfied from orders within $[s_M, t]$. Formally, $M = \max\{m : \sum_{g=m}^{G} \widehat{x}^i_{s_g,t} \geq 0.5\}$. Now, we define the *flow requirement of* (i,t) *that is due to* s_g as

$$z^i_{s_g,t} := \begin{cases} 2\widehat{x}^i_{s_g,t} d_{it} & \text{for } g = 1, \ldots, M-1, \\ 1 - \sum_{q=1}^{M-1} 2\widehat{x}^i_{s_q,t} & \text{for } g = M, \\ 0 & \text{for } g = M+1, \ldots, G. \end{cases}$$

Next, we will construct a feasible assignment such that for every demand point (i,t) and for every order s_g, at least $z^i_{s_g,t}$ units of d_{it} are satisfied from orders within the interval $[s_g, t]$. If we can find such an assignment for $z^i_{s_g,t}$, we say that the flow requirement $z^i_{s_g,t}$ is satisfied. In Figure 2.3, we can see an example of the flow requirements of some demand point (i,t). As we can see, they add up to $10 = d_{it}$. We will find an assignment such that the open facilities in the interval $[s_1, t]$ serve at least 3 units of item i to the demand point (i,t); the open facilities in the interval $[s_2, t]$ serve at least two units of demand to (i,t) (not counting the demand considered for the previous flow requirement), etc. When doing this, it is important to note that every unit of demand served is associated to a unique flow-requirement. This is in order to ensure that the total demand served when satisfying the flow-requirements is d_{it}.

Intuitively, what we do is the following. Let us imagine for a moment that we are not aiming at a 2-approximation but that we want an exact algorithm. We look at a fixed demand point (i,t) and an order s_g that serves it in the optimal LP solution. By the monotonicity of the holding costs, we know that if we are able to find an assignment such that the demand served by s_g is now served by the newly opened orders **located in time between** s_g **and** t and do the same for every demand point and every order that serves it, we will find a feasible solution that pays at most the same total holding cost as the LP solution.

However, this assignment may not always be possible. Thus, we will relax this condition a little. We look at a fixed demand point (i,t) and all the orders that serve it in the LP solution. Let us say the orders are partitioned in two sets, S_1 and S_2, such that each set serves half of the demand d_{it} and the orders in S_2 are all located later in time than the orders in S_1. We will "move" the demand from the orders in S_2 to the orders in S_1, so that now for every order $s_g \in S_1$ we want to satisfy twice the demand it was originally serving, from newly opened orders located between s_g and t. We ignore the orders in S_2.

The definition of $z^i_{s_g,t}$ and the monotonicity of the holding costs imply that the total

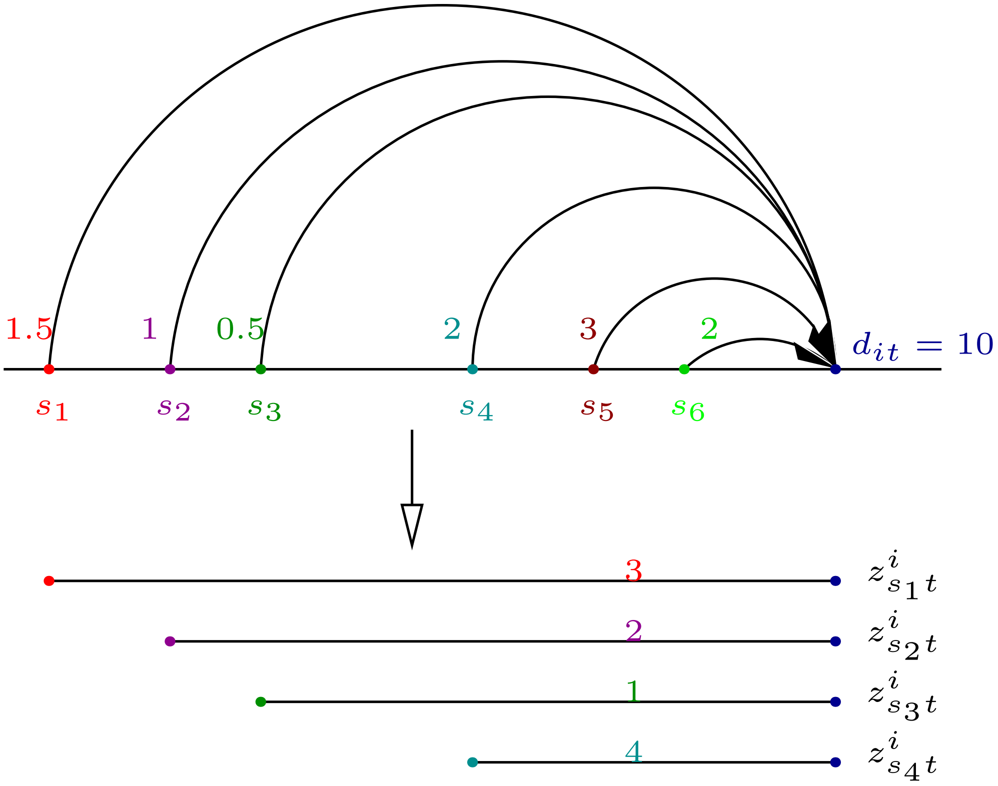

Figure 2.3: This is an example of flow requirements. The numbers by the curved arrows are $d_{it}x^i_{st}$. The positive flow requirements of (i, t) are $z^i_{s_1 t}, z^i_{s_2 t}, z^i_{s_3 t}, z^i_{s_4 t}$, and they are represented by the numbers above the intervals.

holding cost incurred by the solution $\tilde{x}$ we just found is at most $2\sum_{g=1}^{G} \widehat{x}^i_{s_g,t} c^i_{s_g,t} d_{it}$. That is, at most twice the optimal cost incurred by $\widehat{x}$.

Thus, if we are able to complete this assignment (called the Median Assignment) successfully, we will automatically obtain a feasible solution that will give us the desired approximation guarantee.

Now, we will describe the median assignment procedure.

We will define an order $\prec$ on the set of positive flow requirements according to the following rules. Let $z^{i_1}_{s_{g_1},t_1}, z^{i_2}_{s_{g_2},t_2}$ be two positive flow requirements.

- If $s_{g_1} < s_{g_2}$, then $z^{i_1}_{s_{g_1},t_1} \prec z^{i_2}_{s_{g_2},t_2}$

- If $s_{g_1} = s_{g_2}$ and $t_1 < t_2$, then $z^{i_1}_{s_{g_1},t_1} \prec z^{i_2}_{s_{g_2},t_2}$
- If $s_{g_1} = s_{g_2}$ and $t_1 = t_2$ and $i_1 < i_2$, then $z^{i_1}_{s_{g_1},t_1} \prec z^{i_2}_{s_{g_2},t_2}$.

Recall that $\{r_1, r_2, \ldots, r_Q\}$ is the set of open orders obtained in the first phase of the algorithm, and let $z_1, \ldots, z_J$ be all flow requirements in increasing order (the notation z_j is not consistent with what we had before, but it makes the description of the algorithm much easier).

Once we have that, the median assignment procedure is very easy. We describe it formally in Algorithm 1.

Algorithm 1 Median Assignment

```
Initialize $\tilde{x}^i_{s,t} = 0$ for every $i, s, t$.
for $j = J$ to 1 do
  (Say $z_j = z^i_{s,t}$)
  for $q = Q$ to 1 do
    if $s \leq r_q \leq t$ then
      Serve demand from $r_q$ to $z^i_{s,t}$ until either all the capacity of $r_q$ is used or the flow requirement $z^i_{s,t}$ is satisfied.
      Update $z^i_{s,t}$ and $\tilde{x}^i_{r_q,t}$ accordingly.
    end if
  end for
end for
output $\tilde{x}$.
```

We can see an example in Figure 2.4. In each step, we look at the largest (with respect to $\prec$) flow requirement that is not satisfied yet, and we serve it greedily from the available open orders from right to left.

2.4 Analysis

We will prove that after termination of the median assignment procedure, all the flow requirements are satisfied. This will immediately lead to the randomized 2-approximation.

We will start by making a small observation that will be very useful throughout the whole analysis. Recall that $(\widehat{x}, \widehat{y})$ is the optimal LP solution. By construction of the

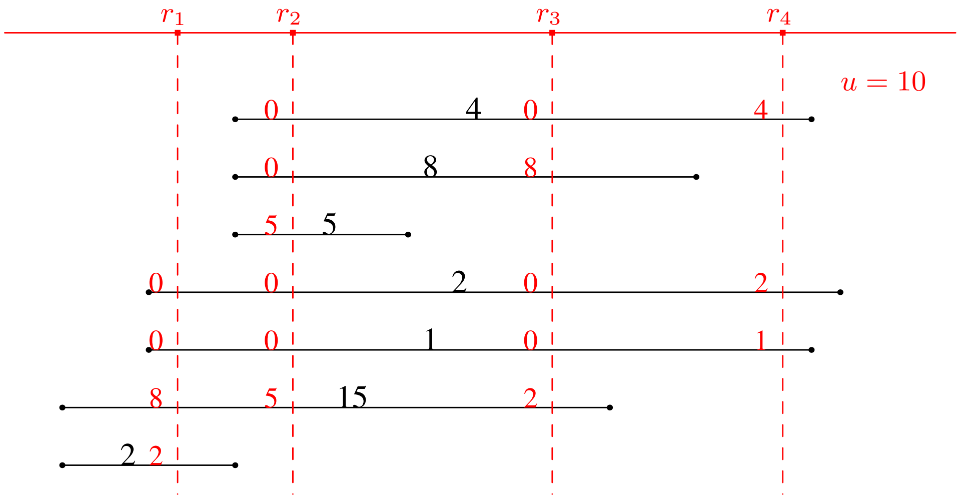

Figure 2.4: A sample run of the median assignment procedure. The flow requirements are in decreasing order (downwards), and are represented by the numbers above the intervals. Each interval corresponds to the two time periods that determine the corresponding flow-requirement. The red numbers next to the dotted lines correspond to how much demand is served from the corresponding facility to the corresponding client.

random-shift procedure, if for some interval $[s,t]$ the total number of fractionally open facilities in this optimal LP solution is at least $(\ell-1) \in \mathbb{Z}$, then the algorithm will open at least that many orders in the same interval. That is, if $\sum_{v=s}^{t} \widehat{y}_v \geq (\ell-1)$ and ℓ is integer, then

$$\sum_{v=s}^{t} \tilde{y}_v \geq \ell - 1. \tag{2.5}$$

In fact, this observation can be made a little stronger.

Lemma 2.4.1 *Let $1 \leq s \leq t \leq T$. Let $\sum_{v=s}^{t} \widehat{y}_v = (\ell-1)+\beta$, where $\ell \in \mathbb{Z}$ and $0 < \beta \leq 1$. Then, if $\beta \geq 0.5$, the number of orders opened in the first phase of the algorithm is at least ℓ. That is, $\sum_{v=s}^{t} \tilde{y}_v \geq \ell$.*

Proof: By construction of the random-shift procedure, the number of open orders is at least $\lfloor \sum_{v=s}^{t} \overline{y}_v \rfloor$. Thus, it will suffice to prove $\sum_{v=s}^{t} \overline{y}_v \geq \ell$.

Let $\mathcal{O}$ be the set of periods $v \in \{s, \ldots, t\}$ such that $\widehat{y}_v \geq 0.5$. Again by construction of the random-shift procedure, this means $\overline{y}_v = \begin{cases} 1 & \text{if } v \in \mathcal{O} \\ 2\widehat{y}_v & \text{if } v \in [s,t] \backslash \mathcal{O} \end{cases}$. That is, $v \in \mathcal{O}$

implies $\overline{y}_v = 1$, and thus $\tilde{y}_v = 1$. So, if $|\mathcal{O}| \geq \ell$, there will be at least ℓ open orders and we are done.

Suppose then that $|\mathcal{O}| \leq \ell - 1$. By all the above,

$$\begin{aligned}
\sum_{v=s}^{t} \overline{y}_v &= \sum_{v \in \mathcal{O}} \overline{y}_v + \sum_{v \in [s,t] \setminus \mathcal{O}} \overline{y}_v = |\mathcal{O}| + 2 \sum_{v \in [s,t] \setminus \mathcal{O}} \widehat{y}_v = |\mathcal{O}| + 2 \left(\sum_{v=s}^{t} \widehat{y}_v - \sum_{v \in \mathcal{O}} \widehat{y}_v \right) \\
&= |\mathcal{O}| + 2 \left((\ell - 1) + \beta - \sum_{v \in \mathcal{O}} \widehat{y}_v \right) \geq |\mathcal{O}| + 2\left((\ell - 1) + 0.5 - |\mathcal{O}|\right) \\
&= 2(\ell - 1) - |\mathcal{O}| + 1 = \ell + (\ell - 1 - |\mathcal{O}|) \geq \ell.
\end{aligned}$$

In the first inequality we used $\sum_{v \in \mathcal{O}} \widehat{y}_v \leq \sum_{v \in \mathcal{O}} 1 = |\mathcal{O}|$ and $\beta \geq 0.5$. In the second inequality, we used $|\mathcal{O}| \leq \ell - 1$. This completes the proof. ■

Lemma 2.4.2 *After termination of the median assignment procedure, all the positive flow requirements are satisfied.*

Proof: Let us assume for contradiction that there exists a flow requirement $z^i_{\tau,\bar{t}}$ that is never satisfied. We consider the instant when the algorithm terminates the iteration corresponding to that flow requirement. That is, the moment when we realize that the flow requirement $z^i_{\tau,\bar{t}}$ will not be satisfied, even if we go on with the algorithm. By construction of the algorithm, all the open orders in the interval $[\tau, \bar{t}]$ are fully used by the current solution $(\tilde{x}, \tilde{y})$. In other words, $\sum_{i=1}^{N} \sum_{v=r}^{T} \tilde{x}^i_{r,v} d_{iv} = u$ for every open order r in the interval $[\tau, \bar{t}]$.

Let $\bar{r}$ be the earliest open order in the interval $(\bar{t}, T]$ that still has free capacity (under $(\tilde{x}, \tilde{y})$), or $T + 1$ if no such order exists.

Let F be the set of all orders in the interval $[\tau, \bar{r})$, regardless if they are open or not. Let A be the set of demand points (i, t) with $t \in [\tau, \bar{r})$ that have positive flow requirements that are due within $[\tau, t]$. That is, $A = \{(i, t) : t \in [\tau, \bar{r}), \sum_{s=\tau}^{t} z^i_{st} > 0\}$. In Figure 2.5 we can see an example.

From this point on, we will focus on the sets F and A and derive a contradiction using the corresponding flow-cover inequality. One of the key observations we will need is the following. Since the order $\bar{r}$ still has free capacity; by construction of the median assignment procedureit follows that all the positive flow requirements that are due within $[\bar{r}, T]$ are fully satisfied by orders in that interval. This means that no open order $r_q < \bar{r}$ is serving demands points in the interval $[\bar{r}, T]$. That is, $\tilde{x}^i_{st} = 0$ for every $t \geq \bar{r}, s < \bar{r}$. Thus, all the capacity from the open orders in the interval $[\tau, \bar{r})$ is fully used to serve demand points

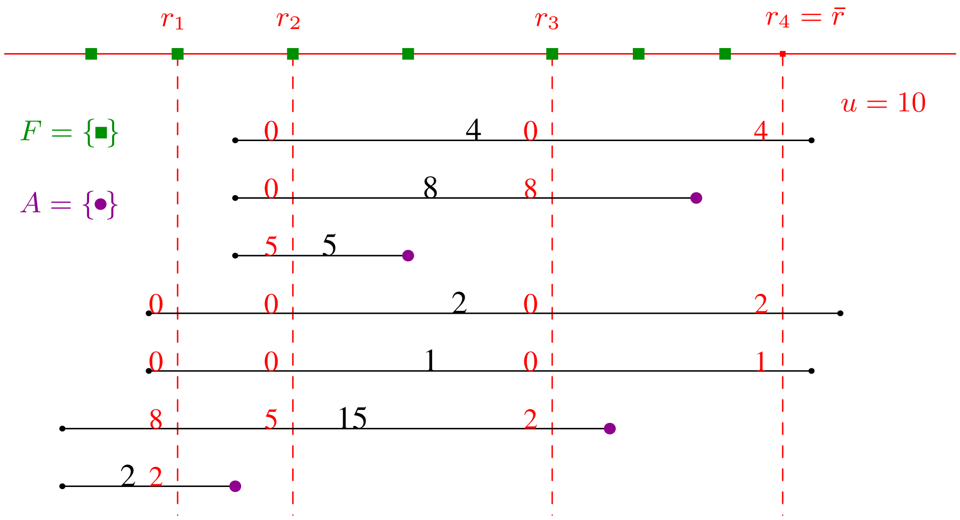

Figure 2.5: In this example, the second-to-last flow requirement is not satisfied by the algorithm. We define F and A as described. As we can see, all the capacity of open orders in F (that is, r_1, r_2 and r_3) is being used to satisfy demand only in A.

(i,t) with $t \in [\tau, \bar{r})$. Furthermore, let (i,t) be a demand point with $t \in [\tau, \bar{r})$ and $(i,t) \notin A$. By the definition of the order $\prec$, all positive flow requirements of (i,t) are smaller than the flow requirements that define A, and by construction of the Median Assignment, these flow requirements have not been considered yet. Thus, all the capacity of open orders in F is being fully used to serve demand points in A.

By the assumption we are making, all this capacity is not enough to satisfy all the flow requirements of demand points in A that are due within the interval F. That is,

$$\sum_{v \in F} \tilde{y}_v u < \sum_{(i,t) \in A} \sum_{v \in F} z^i_{v,t}. \tag{2.6}$$

Now, using the same notation we introduced before defining the flow-cover inequalities, we express the total demand over A as $D(A) = (\ell_A - 1)u + R_A$, with $\ell_A \in \mathbb{Z}$ and $0 < R_A \leq u$. Similarly, we express the total capacity of the fractionally open orders over F in the optimal LP solution as $\sum_{v \in F} \widehat{y}_v u = (\ell - 1)u + R$, with $\ell \in \mathbb{Z}$ and $0 < R \leq u$.

The rest of the proof is structured as follows: First we prove $\ell \leq \ell_A$, then $R/u < 0.5$ and then $\ell = \ell_A$. After that, we use these facts to derive a contradiction from the flow-cover inequality corresponding to F and A.

The intuition behind this is the following. We know that the fractional solution opens enough capacity in F to serve A. The random-shift procedure ensures that the capacity in F in the rounded solution is also enough to satisfy this demand. That is, there are as many open facilities in F as are needed. In other words, $\ell = \ell_A$. Since we are assuming that the median assignment procedure could not be completed, this will give us a contradiction.

$\ell \leq \ell_A$ is true because

$$(\ell - 1)u \leq \sum_{v \in F} \tilde{y}_v u < \sum_{(i,t) \in A} \sum_{v \in F} z^i_{v,t} \leq D(A) \leq \ell_A u.$$

The first inequality follows from the definition of ℓ, the fact that F is a set of facilities in an interval, and from inequality (2.5). The second inequality comes from inequality (2.6). The last two follow from the definitions of the flow requirements and of ℓ_A, respectively. Since ℓ and ℓ_A are integers, $\ell - 1 < \ell_A$ implies $\ell \leq \ell_A$.

Next, we prove $R/u < 0.5$.
Suppose $R/u \geq 0.5$. By definition of ℓ and R, we have that $\sum_{v \in F} \widehat{y}_v \geq (\ell - 1) + 0.5$. From Lemma 2.4.1, the random-shift procedure opens at least ℓ orders in F. In other words, $\sum_{v \in F} \tilde{y}_v u \geq \ell u$. We have the following chain of inequalities, which contradicts (2.6).

$$\sum_{v \in F} \sum_{(i,t) \in A} z^i_{v,t} \leq \sum_{v \in F} \sum_{(i,t) \in A} \widehat{x}^i_{v,t} d_{it} \leq \sum_{v \in F} \widehat{y}_v u = (\ell - 1)u + R \leq \ell u \leq \sum_{v \in F} \tilde{y}_v u.$$

The first inequality follows from the definition of flow requirements, the second from the capacity restrictions (2.2) in the LP. The last inequality is the observation we just made above. Thus, $R/u < 0.5$.

Now, we prove $\ell = \ell_A$. Suppose $\ell_A - 1 \geq \ell$.
The following observation is very important. Let $(i,t) \in A$. By definition of A, (i,t) has positive flow requirements over F. Again by definition, F is an interval, and by definition of flow requirements, more than half of the demand of (i,t) is satisfied from within F in the optimal LP solution (if it were not so, all flow requirements of (i,t) over F would be zero). That is, $\sum_{v \in F} \widehat{x}^i_{v,t} > 0.5$. Thus, we have

$$\sum_{v \in F} z^i_{v,t} = 2 d_{it} \left(\sum_{v \in F} \widehat{x}^i_{v,t} - 0.5 \right).$$

The -0.5 corresponds to the flow requirements that are zero, all of which are within the interval F.

Adding over all demand points in A:

$$\begin{aligned}\sum_{(i,t)\in A}\sum_{v\in F} z^i_{v,t} &= 2\left(\sum_{(i,t)\in A}\sum_{v\in F}\widehat{x}^i_{v,t}d_{it} - 0.5D(A)\right) \leq 2\left(\sum_{v\in F}\widehat{y}_v u - 0.5D(A)\right)\\ &= 2(\ell-1)u + 2R - (\ell_A - 1)u - R_A \leq 2(\ell-1)u + u - \ell u + 0\\ &= (2\ell - 2 + 1 - \ell)u = (\ell-1)u \leq \sum_{v\in F}\tilde{y}_v u.\end{aligned}$$

The first inequality follows from the LP restrictions (2.2). The second inequality comes from the assumptions $\ell_A - 1 \geq \ell$ and $2R < u$, as well as from the trivial fact $R_A \geq 0$. The last inequality comes from (2.5).
Again, this contradicts (2.6), so we can conclude $\ell = \ell_A$.

Thus, let us assume $R/u < 0.5$ and $\ell_A = \ell$. By definition of ℓ and R, we know that the LP opened (fractionally) a total of $(\ell - 1) + R > \ell - 1$ orders in F. This means there are at least ℓ orders in F, and thus $|F| \geq \ell = \ell_A$. So, by definition, F is a cover of A. Furthermore, F is the set of all orders in an interval; that is, it is of the form $[s,t]$. Thus, we know that the corresponding flow-cover inequality is valid for (MIP-LS) and is thus included in the restrictions of (LP-LS). Thus, the fractional solution $(\widehat{x}, \widehat{y})$ must satisfy it:

$$\begin{aligned}D(A) - \sum_{(i,t)\in A}\sum_{v\in F}\widehat{x}^i_{v,t}d_{it} &\geq R_A\left(\ell_A - \sum_{v\in F}\widehat{y}_v\right) = R_A(\ell_A - (\ell-1) - R/u)\\ &= R_A(1 - R/u) \geq 0.5R_A.\end{aligned}$$

Here, we used the flow-cover inequality, the definition of L and R, and the facts we proved above.

Thus, we have $2(D(A) - \sum_{(i,t)\in A}\sum_{v\in F}\widehat{x}^i_{v,t}d_{it}) \geq R_A$. Reformulating,

$$2\left(\sum_{(i,t)\in A}\sum_{v\in F}\widehat{x}^i_{v,t}d_{it} - 0.5D(A)\right) \leq D(A) - R_A.$$

But in the proof of $\ell = \ell_A$ we saw that the left hand side is equal to $\sum_{(i,t)\in A}\sum_{v\in F} z^i_{v,t}$, and we know the right hand side is equal to $(\ell_A - 1)u$ by definition of R_A.
So, using this and inequality (2.5) once more,

$$\sum_{(i,t)\in A}\sum_{v\in F} z^i_{v,t} \leq (\ell_A - 1)u = (\ell-1)u \leq \sum_{v\in F}\tilde{y}_v u.$$

Again, this contradicts (2.6), and so we are done. ■

We can now state the main theorem:

Theorem 2.4.3 *The random-shift with median assignment procedure is a randomized 2-approximation algorithm for the multi-item lot-sizing problem with uniform capacities.*

Proof: Because the median assignment procedure completed the assignment successfully and by definition of flow requirements, we have

$$\sum_{i=1}^{N}\sum_{s=1}^{T}\sum_{t=s}^{T} c^i_{st} d_{it} \tilde{x}^i_{st} \le \sum_{i=1}^{N}\sum_{s=1}^{T}\sum_{t=s}^{T} c^i_{st} z^i_{st} \le 2\sum_{i=1}^{N}\sum_{s=1}^{T}\sum_{t=s}^{T} c^i_{st} d_{it} \widehat{x}^i_{st}.$$

Let OPT_{LP} be the optimal LP value, and let Z be the objective value of the rounded solution. By Lemma 2.3.1 and by the inequality above, we have

$$\mathbb{E}[Z] = \mathbb{E}\left[\sum_{s=1}^{T} f_s \tilde{y}_s + \sum_{i=1}^{N}\sum_{s=1}^{T}\sum_{t=s}^{T} c^i_{st} d_{it} \tilde{x}^i_{st}\right] \le 2\sum_{s=1}^{T} f_s \widehat{y}_s + 2\sum_{i=1}^{N}\sum_{s=1}^{T}\sum_{t=s}^{T} c^i_{st} d_{it} \widehat{x}^i_{st} = 2\cdot OPT_{LP}.$$

■

One should note that the expected value is needed to bound the ordering costs, but not the holding costs. The beauty of the algorithm is that no matter what the outcome of the first phase is, the second phase is guaranteed to find a not too expensive assignment of orders.

2.5 Derandomization and On-the-Fly Algorithm

The algorithm can be easily derandomized, by noting that even though α has an infinite number of possible values, the possible outputs of the first phase are polynomial. In fact, there are $O(T)$ different sets of orders that could be chosen by the random-shift procedure. So, one can consider all these sets of orders, and the cheapest is guaranteed to be bounded by $2\sum_{s=1}^{t} \widehat{y}_s$. Then, we can keep that set of open orders and proceed to the second phase. Thus, we get a deterministic 2-approximation.

Another way of approaching the problem is through an *on-the-fly* variant. The on-the-fly algorithm consists in not adding all constraints corresponding to $\mathcal{I}$ at once to the LP formulation, and instead trying to solve the median assignment. If the assignment is possible, we are done. Otherwise, we have found a violated flow-cover inequality (as seen in the proof of correctness of the algorithm), which we add to the restrictions, and we repeat the process until the median assignment is possible. Clearly, every such step (that is, running the median assignment procedure to find a violated inequality) takes polynomial time. However, we may have to repeat that procedure an exponential number of times. The advantage of this variant is that it is much easier to implement, but for theoretical purposes it offers no significant advantages.

Finally, we can also note that the description and analysis of the median assignment procedure is necessary to prove that the orders that were opened by the random-shift procedure are enough to guarantee a good solution. However, once we know that, we could simply describe the algorithm as follows:

1. Solve the strengthened LP relaxation, let $(\widehat{x}, \widehat{y})$ be an optimal solution.
2. Run the random-shift procedure, let $\tilde{y}$ be its output.
3. Solve the induced transportation problem. That is, fix $y = \tilde{y}$ in (MIP-LS) (this will give us an LP with variables x) and solve it to obtain $\tilde{x}$. By all the analysis above, we know that this resulting problem will always be feasible and will give us a good approximation, so $(\tilde{x}, \tilde{y})$ is a randomized 2-approximation.

This way of stating the algorithm is much simpler than the original, and it should in general be more effective than the median assignment procedurewhen implementing it; but it is not obvious how to prove the 2-approximation guarantee (or a better guarantee) for it.

One can note that the 2 in the approximation guarantee comes from the definition of both the random-shift procedure (where we multiply the y-values by 2) and from the median assignment procedure(where we double the demand that the longest intervals need to serve). The choice of the number 2 seems rather arbitrary. In the next chapter, when we generalize these ideas, we will see that 2 is the best possible ratio.

Chapter 3

An LP-Rounding Algorithm for the Capacitated Facility Location Problem with Uniform Capacities

In this chapter, we adapt to CFL with uniform capacities the ideas used in the previous chapter. We will first work with a special case of the problem in which the assignment costs are given by a tree metric. Adapting Levi et al.'s algorithm to the even more particular case where the tree is a path is very easy. Generalizing to trees is not straightforward, and requires several new ideas.

Our main result in this section is an approximation algorithm for this problem that has a constant factor approximation guarantee if the number of leaves of the tree is bounded. At the end of the chapter, we show how the same algorithm works for a slightly more general case, which we know is NP hard. We also very briefly discuss how we can use the techniques of probabilistic embedding of metric spaces on tree metrics to obtain an approximation algorithm for general CFL.

3.1 The Problem

Consider the following special case of the capacitated facility location problem. Let $\mathcal{F}$ be a set of facilities with uniform capacities u and $\mathcal{C}$ a set of clients, each client j with a demand d_j. Let $T = (\mathcal{F} \cup \mathcal{C}, E)$ be a tree and $c : E \to \mathbb{R}_+$. For any two vertices u and v, let $c(u, v)$ (or c_{uv}, we use both indistinctly) denote the tree distance from u to v (that is, the sum of the weights of the edges on the unique u, v-path in T). The assignment costs are determined by this distance. Let l be the number of leaves of this tree.

The following proposition allows us to simplify the problem.

Proposition 3.1.1 *Without loss of generality, we may assume that all leaves of T are facilities.*

Proof: Let T' be the smallest subgraph of T that is also a tree and contains all facilities as vertices. Clearly, all leaves of T' are facilities. Let u be a vertex of $T \backslash T'$ corresponding to a client j, and suppose there is no facility at u. Let v be the unique closest vertex to u in T'. Consider a modified problem where the client j is now located at v instead of u. Clearly, there is a bijection between all solutions to the original problem and all solutions to the new problem, and the objective values differ by a constant (namely, the cost of sending d_j units of demand from v to u). We can add this constant to the objective function of the new problem, so that the objective values of equivalent solutions of both problems are the same. So, if we do the same thing for all clients not in T' and solve the resulting problem, we get a solution of the same cost in the original problem. ■

It is clear that this assumption can be made in terms of the optimal value. However, it is less clear why this still holds in an approximation setting. Using the notation from above, suppose that the original problem has an optimal value $z+c$, where c is the constant arising from connecting every client in $T \backslash T'$ to the closest node in T'. After we modify the instance to make all leaves contain facilities, the new objective value is z'. Suppose we can find an approximate solution to this new problem of value $\bar{z} \leq \alpha z'$. Clearly, $z' = z$, so we have $\bar{z} \leq \alpha z$. Modifying the approximate solution to serve clients outside of T', we get a solution of cost $\bar{z} + c \leq \alpha z + c \leq \alpha(z + c)$. So, we obtained an α-approximation for the original problem.

For two vertices u and v of the tree, let $p(u, v)$ denote the set of vertices on the unique u, v-path. Let us consider the following set:

$$\mathcal{S} = \{F \subseteq \mathcal{F} | p(i_1, i_2) \cap \mathcal{F} \subseteq F \quad \forall i_1, i_2 \in F\}.$$

That is, $\mathcal{S}$ is the family of sets of facilities corresponding to connected subgraphs (or *subtrees*) of T. The sets in $\mathcal{S}$ generalize intervals of facilities from the line case.

3.2 The Algorithm

In this section, we give an LP-rounding algorithm that is strongly inspired on the algorithm described and analyzed in Chapter 2. It is not a generalization in the strictest sense of the word, because CFL does not generalize the lot-sizing problem. However, the algorithm (and analysis) of the case when the tree is just a path is almost identical to Levi et al's algorithm. Therefore, there will be a lot of similarities as well with the more general case we will present.

Our algorithm also relies on the use of flow-cover inequalities. In the previous chapter, we proved the validity of these for the lot-sizing problem. They are equally valid for CFL with uniform capacities, and the proof is identical, so it will be omitted. As in the previous chapter, let us define $D(A) = \sum_{j \in A} d_j$, $\ell_A = \lceil D(A)/u \rceil$ and $R_A = D(A) - (\ell_A - 1)u$. So, the flow-cover inequality corresponding to sets $F \subseteq \mathcal{F}$, $A \subseteq \mathcal{C}$ is the following:

$$D(A) - \sum_{j \in A} \sum_{i \in F} x_{i,j} d_j \geq R_A \left(L_A - \sum_{i \in F} y_i \right).$$

Using these inequalities, we obtain the following relaxation (notice that we include only flow-cover inequalities corresponding to sets of facilities in $\mathcal{S}$

$$\text{(LP-CFL)} \quad \begin{array}{lll} \min & \sum_{i \in \mathcal{F}} f_i y_i + \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{C}} d_j x_{ij} & \\ \text{s.t} & \sum_{i \in \mathcal{F}} x_{ij} \geq 1 & \text{for every } j \in \mathcal{C}, \quad (3.1) \\ & \sum_{j \in \mathcal{C}} d_j x_{ij} \leq u y_i & \text{for every } i \in \mathcal{F}, \quad (3.2) \\ & D(A) - \sum_{j \in A} \sum_{i \in F} x_{i,j} d_j \geq R_A \left(L_A - \sum_{i \in F} y_i \right) & \text{for every } A \subseteq \mathcal{C}, F \in \mathcal{S}, \\ & y_i \leq 1 & \text{for every } i \in \mathcal{F}, \\ & y_i \geq 0 & \text{for every } i \in \mathcal{F}, \\ & x_{ij} \geq 0 & \text{for every } i \in \mathcal{F}, j \in \mathcal{C}, \end{array}$$

The algorithm we will present is divided in two phases as before:

Extended Random-Shift Procedure:

1. Solve (LP-CFL), and let $(\widehat{x}, \widehat{y})$ be an optimal solution found.[1]

[1] In Section 3.6 we will see under what conditions we can do this efficiently.

2. Partition $\mathcal{F}$ into sets from $\mathcal{S}$ such that for every set B in the partition, the smallest connected subgraph of T that contains all vertices corresponding to facilities in B is a path, and all facilities in this path belong to B. Intuitively, we decompose the tree into paths that may intersect at client-vertices but not at facility-vertices. Let p be the cardinality of the partition. (Note: We can always find a trivial partition of size $l-1$. On the other hand, for every partition we have $p \geq \frac{l}{2}$, since a path can contain at most two leaves. We can not always have $p = \ell/2$, for example if ℓ is odd.). We can see an example in Figure 3.1

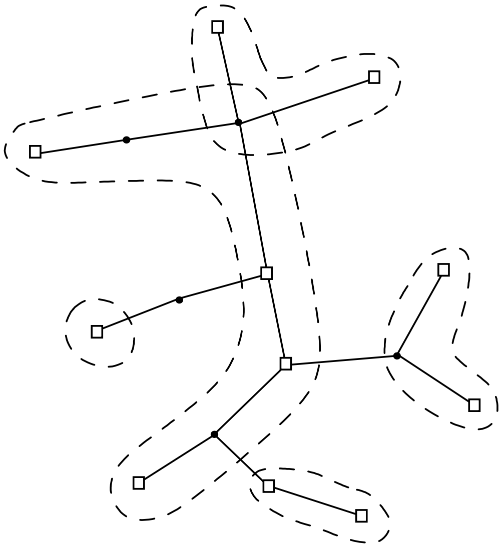

Figure 3.1: In this instance, squares represent facilities and circles represent clients. A possible partition of the facilities into paths is shown. This partition gives us $p = 5$.

3. Run Levi et al.'s random-shift procedure on each of these paths using the factor $p+1$ instead of 2, and open the selected facilities. That is, for every $i \in \mathcal{F}$, define $\bar{y}_i = \min\{(p+1)\widehat{y}_i, 1\}$, and then round as before, opening each facility i with probability $\bar{y}_i$. For every $i \in \mathcal{F}$, let $\tilde{y}_i = 1$ if i is open, $\tilde{y}_i = 0$ otherwise.

Extended Median Assignment: Let us first define flow requirements in a similar fashion to those in the previous chapter. Because in this chapter we are not aiming at a 2-approximation, we will use $p+1$ and not 2 in the definition.[2]

[2] The factor $p+1$ already appeared in the definition of the extended random-shift algorithm. We will justify the choice of that number in Section 3.5.

For every client j, let let $i_1, \dots, i_m$ be all facilities in $\mathcal{F}$ that fractionally serve j in the LP solution, labeled such that $c(i_1, j) \leq c(i_2, j) \leq \cdots \leq c(i_m, j)$. Let M (both the facilities $i_1, \dots, i_m$ and M are in fact dependent on j, but we omit j from the notation for clarity of exposition) be the smallest index such that $\sum_{g=1}^{M} x_{i_g,j} \geq \frac{p}{p+1}$. We call i_M the median facility (We are abusing the word *median* here, since we no longer look for the facility "in the middle". However, we will keep using that name for simplicity). For every g, the *flow requirement of j from i_g* is defined as

$$z_{i_g,j} := \begin{cases} (p+1)\widehat{x}_{i_g,j} d_j & \text{for } g = M+1, \dots, m, \\ (1 - \sum_{q=M+1}^{m} (p+1)\widehat{x}_{i_q,j}) d_j & \text{for } g = M, \\ 0 & \text{for } g = 1, \dots, M-1. \end{cases}$$

By this definition, we have $\sum_{i \in \mathcal{F}} z_{ij} = d_j$ for every client j. That is, we decompose each demand d_j into several flow requirements z_{ij}, which we will require to be satisfied from facilities within the ball centered at j with radius $c(i, j)$. Let F_{ij} be the set of facilities in this ball. We will call F_{ij} a *cluster* centered around j. An example is shown in Figure 3.2. Note that, by the monotonicity of the assignment costs, $F_{ij} \in \mathcal{S}$ (that it, F_{ij} is the set of facilities in a subtree). We will use the notation F_{ij} when we want to emphasize the client and facility that determine this cluster. If this is not important, we will usually write F' or F^s for some s.

Above, we mentioned the notion of "satisfying" flow requirements. Here we will formalize that notion, which is in fact analogous to that in Chapter 2. Suppose that in the final solution, we have some $i \in \mathcal{F}$, $j \in \mathcal{C}$ such that $\tilde{x}_{ij} > 0$. Let $F^1, F^2, \dots, F^t$ be all clusters centered around j that contain i. A *decomposition* of $\tilde{x}_{ij}$ is a set of variables $\tilde{x}_{ij}(F^1), \dots, \tilde{x}_{ij}(F^t)$ such that for every $s \in 1, \dots, t$ we have $\tilde{x}_{ij}(F^s) \geq 0$ and $\sum_{s=1}^{t} \tilde{x}_{ij}(F^s) = \tilde{x}_{ij}$. Given i, j, a (not necessarily feasible) solution $(\tilde{x}, \tilde{y})$ and a decomposition of $\tilde{x}_{i'j}$ for every $i' \in F_{ij}$, we say the flow requirement z_{ij} is *satisfied* if $z_{ij} = \sum_{i' \in F_{ij}} \tilde{x}_{i'j}(F_{ij})$. We say that facility i' serves a demand $\tilde{x}_{i'j}(F_{ij})$ to flow requirement F_{ij}. To avoid unnecessary notation, we will simply use expressions like "we serve demand from i' to flow requirement F_{ij}" meaning that we increase the variable $\tilde{x}_{i'j}(F_{ij})$.

In the first part of the algorithm, we obtained an integral vector $\tilde{y}$. In the extended median assignment procedure, we will construct a demand assignment $\tilde{x}$ such that not only $(\tilde{x}, \tilde{y})$ is feasible, but $\tilde{x}$ satisfies all flow requirements. That is, we can partition all components of $\tilde{x}$ such that all flow requirements are satisfied. We will achieve this by iteratively increasing the variables $\tilde{x}_{ij}(F^s)$.

We are now ready to state the procedure:

1. Initialize $\tilde{x} = 0$. Define flow requirements for every client as described above.

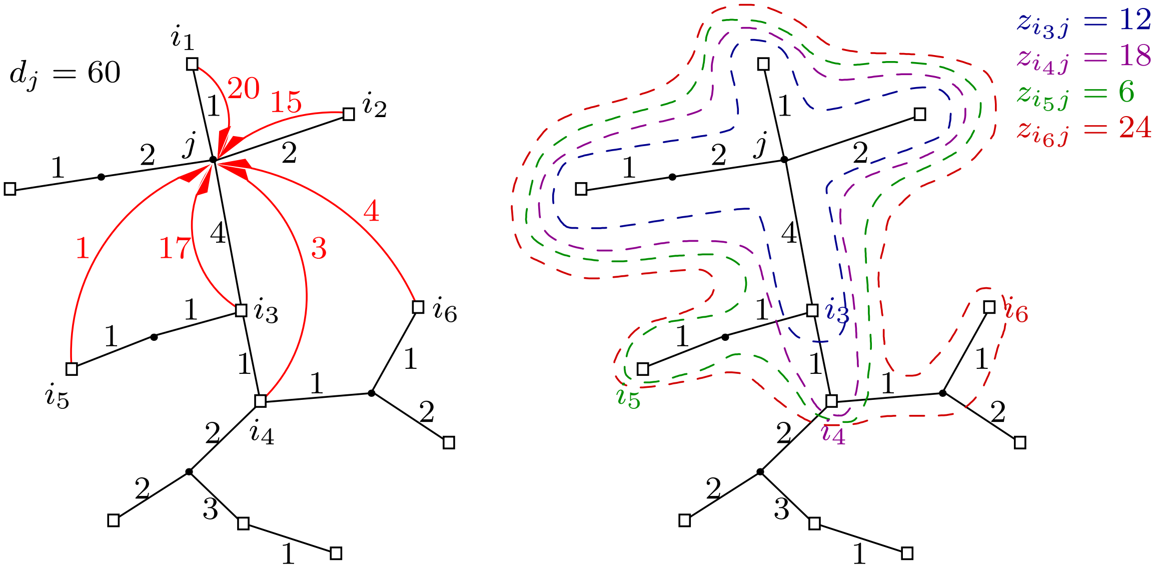

Figure 3.2: On the left, we see a possible assignment of demands to client j, represented by the red arrows. On the right, we can see the values of the nonnegative flow requirements corresponding to j and their respective clusters (assuming $p = 5$). Here, i_3 is the median facility.

2. Serve the flow requirements from the open facilities greedily, in any order, until each flow requirement either is satisfied or cannot be satisfied because all facilities in its cluster are fully used (in this case, we say the flow requirement is *frustrated*). Update $\tilde{x}$ accordingly. If all flow requirements are satisfied, stop.

3. Otherwise, choose an arbitrary frustrated flow requirement, and *transfer* capacity to it from other facilities (we will define this notion of transferring capacity shortly) until it is satisfied. Update $\tilde{x}$. Repeat for all frustrated flow requirements.

Now, we define the process of capacity transferring. Given a cluster F^0 corresponding to a frustrated flow requirement, for every facility $i' \in \mathcal{F} \backslash F^0$ we say i' is *connected* to F^0 if there exist clusters $F^1, \ldots, F^q \in \mathcal{S}$ and facilities $i_0, i_1, \ldots, i_q = i' \in \mathcal{F}$ such that the following conditions hold:

- For every $j < q$, $i_j \in F^j \cap F^{j+1}$ and i_j serves a positive demand to the flow requirement corresponding to F^{j+1}
- $i_q \in F^q$.

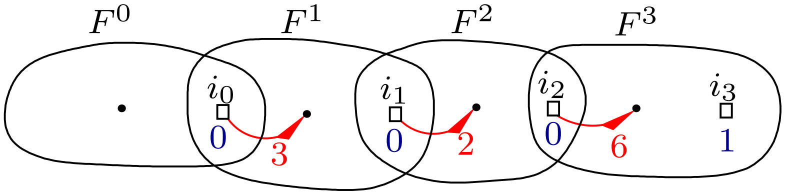

Figure 3.3: Here, we can see that i_0 serves F^1, i_1 serves F^2 and i_2 serves F^3. The numbers below the facilities represent the current available capacity. According to our definition, i_3 is connected to F^0.

Figure 3.3 shows an example.

We define the *connected component of* F^0 as $F^0 \cup \{i \in \mathcal{F} | i \text{ is connected to } F^0\}$. Note that by the definition above, all facilities in $F^1 \cup \cdots \cup F^q$ are connected to F^0. Since we also have $F^j \cap F^{j+1} \neq \emptyset$ for $j = 0, \dots, q-1$, this means that the connected component of F^0 is in $\mathcal{S}$.

Consider the following procedure:

If there exists a facility with free capacity connected to F^0, let us call it $i' = i_q$. Also, let the sets F^j and facilities i_j be as above. We use the free capacity of i_q to replace (possibly a fraction of) the demand served to F^q by the facilities in $F^{q-1} \cap F^q$ that serve F^q (this set is non empty, since it contains i_{q-1}). These facilities now have some free capacity, we use it to serve F^{q-1} while freeing capacity in facilities in $F^{q-2} \cap F^{q-1}$, and so on. We move along the chain of clusters and repeat this procedure until we reach F^0. This procedure is illustrated in Figure 3.4 If this facility does not exist, we terminate the algorithm and declare that the extended median assignment got stuck.

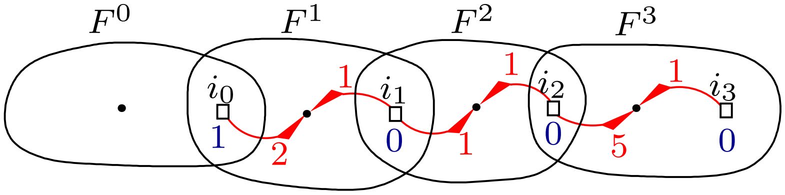

Figure 3.4: Continuing with the previous example, we notice that i_3 is connected to F^0 and has 1 unit of capacity available. We update the demand assignments as shown and thus transfer the unit of free capacity to i_0. Now, we can use it to serve demand to F^0.

At this point, at least one facility in F^0 will have a positive amount of free capacity, which we can use to further serve the corresponding flow requirement. Note that if this procedure lets us free more capacity in F^0 than needed, we can always regulate it to free exactly the amount of capacity we need, and this is what we will do. This way, all facilities use as much capacity as they were using before, except for i'. Furthermore, all flow requirements that were being satisfied are still satisfied. We refer to this procedure as *transferring* capacity from i' to F^0.

The motivation for doing this is simple, even if a flow requirement is frustrated, we can use capacity available at neighbouring facilities to help satisfy it. Furthermore, if the extended median assignment gets stuck, we can obtain a set of facilities F and a set of clients A such that open facilities in F serve only clients in A but do not have enough capacity to satisfy the flow requirements corresponding to clients in A and facilities in F. If a flow requirement is still frustrated after transferring as much capacity as possible, we know that all facilities connected to it are using their full capacity, and as we will see, this will yield sets A and F with the properties we seek.

3.3 Analysis

The following two lemmas prove the approximation guarantee of the algorithm.

Lemma 3.3.1 *The total expected opening cost of the solution obtained by the extended random-shift procedure is at most* $(p+1)\sum_{i\in\mathcal{F}}\widehat{y}_i f_i$.

Proof: The probability of opening facility i is exactly $\overline{y}_i$, which by definition is at most $(p+1)\widehat{y}_i$. Let K be the opening cost of the rounded solution. We have:

$$\mathbb{E}(K) = \sum_{i\in\mathcal{F}} \mathbb{P}(\tilde{y}_i = 1) f_i = \sum_{i\in\mathcal{F}} \overline{y}_i f_i \le \sum_{i\in\mathcal{F}} (p+1)\widehat{y}_i f_i = (p+1)\sum_{i\in\mathcal{F}} \widehat{y}_i f_i.$$

■

Lemma 3.3.2 *If the extended median assignment procedure is completed, the total assignment cost of the solution obtained by it is at most* $(p+1)\sum_{i\in\mathcal{F}}\sum_{j\in\mathcal{C}} c(i,j) d_j \widehat{x}_{ij}$.

Proof:

Since for every $i\in\mathcal{F}, j\in\mathcal{C}$ we satisfy the flow requirement z_{ij} from facilities with distance at most $c(i,j)$ from j, we have

$$\sum_{i\in\mathcal{F}} c(i,j) d_j \tilde{x}_{ij} \le \sum_{i\in\mathcal{F}} c(i,j) z_{ij}.$$

So,

$$\sum_{i\in\mathcal{F}}\sum_{j\in\mathcal{C}} c(i,j)d_j\tilde{x}_{ij} \leq \sum_{i\in\mathcal{F}}\sum_{j\in\mathcal{C}} c(i,j)z_{ij} \leq \sum_{i\in\mathcal{F}}\sum_{j\in\mathcal{C}}(p+1)c(i,j)d_j\widehat{x}_{ij}.$$

The last inequality follows from the definition of flow requirements. ■

Thus, to prove the approximation guarantee, all we need to do is prove that the extended median assignment procedure can be completed successfully. We do that in the following theorem.

Theorem 3.3.3 *The extended median assignment procedure terminates successfully. That is, together with the extended random-shift procedure, it gives us a randomized $(p+1)$-approximation to the capacitated facility location problem with uniform capacities on trees.*

Before we prove the theorem, we need the following two lemmas, which are inspired by Lemma 2.4.1.

Lemma 3.3.4 *Consider $i_1,\ldots,i_m$ consecutive facilities along a path, on which we run the random-shift procedure. If $\widehat{y}_{i_1}+\cdots+\widehat{y}_{i_m} = \ell + r$, with $r \geq \frac{1}{p+1}$, then $\bar{y}_{i_1}+\cdots+\bar{y}_{i_m} \geq \ell+1$.*

Proof: Let $\mathcal{O} = \{s|\widehat{y}_{i_s} \geq \frac{1}{p+1}\}$. By construction of the random-shift procedure, we have
$\bar{y}_i = \begin{cases} 1 & \text{if } i \in \mathcal{O} \\ (p+1)\widehat{y}_i & \text{if } i \notin O \end{cases}$. Clearly, if $|\mathcal{O}| \geq \ell+1$, we are done.

So, assume $|\mathcal{O}| \leq \ell$.

$$\begin{aligned}
\sum_{s=1}^{m}\bar{y}_{i_s} &= \sum_{s\in\mathcal{O}}\bar{y}_{i_s} + \sum_{s\notin\mathcal{O}}\bar{y}_{i_s} = |\mathcal{O}| + (p+1)\left(\sum_{i=1}^{m}\widehat{y}_{i_s} - \sum_{s\in\mathcal{O}}\widehat{y}_{i_s}\right) \\
&= |\mathcal{O}| + (p+1)\left(\ell + r - \sum_{s\in\mathcal{O}}\widehat{y}_{i_s}\right) = |\mathcal{O}| + (p+1)\ell + (p+1)r - (p+1)\sum_{s\in\mathcal{O}}\widehat{y}_{i_s} \\
&\geq |\mathcal{O}| + (p+1)\ell + 1 - (p+1)|\mathcal{O}| = \ell + 1 + \ell p - |\mathcal{O}|p \\
&= \ell + 1 + (\ell - \mathcal{O})p \geq \ell + 1.
\end{aligned}$$

In the first inequality we used $r \geq \frac{1}{p+1}$ and $\bar{y}_i \leq 1$ for every i. In the second one, we used $|\mathcal{O}| \leq \ell$. ■

Lemma 3.3.5 *For every $F \in \mathcal{S}$, let $\sum_{i\in F}\widehat{y}_i = L + R$, with $L \in \mathbb{Z}, R \in (0,1]$. The following are true:*

(a) $\sum_{i \in F} \tilde{y}_i \geq L$

(b) If $R \geq \frac{p}{p+1}$, *then* $\sum_{i \in F} \tilde{y}_i \geq L + 1$.

Proof: Let $P_1, P_2, \ldots, P_p$ be the partition of the facilities in F induced by the original partition of the tree (some of the P_j may be empty). If $p = 1$, we know the result is true by the previous lemma, so let us assume $p > 1$.

For every $j = 1, \ldots, p$, let us write $\sum_{i \in P_j} \widehat{y}_i = \ell_j + r_j$, with $\ell_j \in \mathbb{Z}, r_j \in [0, 1)$. Without loss of generality, say $r_1 \geq r_2 \geq \ldots r_p$.

Clearly,

$$L + R = \sum_{i \in F} \widehat{y}_i = \sum_{j=1}^{p} \ell_j + \sum_{j=1}^{p} r_j.$$

Let $k = L - \sum_{j=1}^{p} \ell_j = \sum_{j=1}^{p} r_j - R$. If we prove that there exist at least k paths P_j such that $\sum_{i \in P_j} \bar{y}_i \geq \ell_j + 1$, we are done; because by observation (2.5), in each of these paths there will be at least $\ell_j + 1$ facilities with $\tilde{y}_i = 1$, so $\sum_{i \in F} \tilde{y}_i \geq \sum_{j=1}^{p} \ell_j + k = L$. By Lemma 3.3.4, if we prove that $r_1, \ldots, r_k \geq \frac{1}{p+1}$ (note that if $k = 0$ we are done, so we may assume without loss of generality that $k \geq 1$), we are done. In fact, we will prove something slightly stronger: $r_1, \ldots, r_k \geq \frac{1}{p}$.

To prove this, suppose otherwise. Then, $r_j < \frac{1}{p}$ for every $j = k, \ldots, p$. Adding all these inequalities, we obtain

$$r_k + \cdots + r_p < (p - k + 1)\frac{1}{p} \leq 1. \tag{3.3}$$

We also know $r_j < 1$ for every $j = 1, \ldots, k - 1$. This implies

$$r_1 + \cdots + r_{k-1} < k - 1. \tag{3.4}$$

Adding (3.3) and (3.4), we get $\sum_{j=1}^{p} r_j < k - 1 + 1 = k = \sum_{j=1}^{p} r_j - R$, which is a contradiction.

This proves (a).

Similarly, for (b) we can notice that if $\sum_{j=1}^{p} r_j \geq k + \frac{p}{p+1}$, then $r_1, \ldots, r_{k+1} \geq \frac{1}{p+1}$, which implies the result using Lemma 3.3.4 again. If this was not the case, we would have

$$r_{k+1} + \cdots + r_p < (p - k)\frac{1}{p+1} = \frac{p}{p+1} - \frac{k}{p+1} = \frac{p-k}{p+1} < \frac{p}{p+1}$$

and

$$r_1 + \cdots + r_k < k.$$

Adding both inequalities, we obtain $\sum_{j=1}^{p} r_j < k + \frac{p}{p+1}$, which is again a contradiction. So, the result follows. ■

As in the previous chapter, the proof of the main theorem is achieved by assuming that the extended median assignment procedure cannot be completed, and identifying sets F and A that violate a flow-cover inequality. We construct and analyze these sets next.

If the extended median assignment procedure gets stuck when trying to satisfy the flow requirement corresponding to cluster F^0, define F as the connected component of F^0, and A as the set of clients currently being served by a facility in F. Sets A and F satisfy the following properties:

(i) For every $i \in F$, $j \notin A$, we have $\tilde{x}_{ij} = 0$. That is, F serves demand only to clients in A.

(ii)

$$\sum_{i \in F} \tilde{y}_i u < \sum_{j \in A} \sum_{i \in F} z_{i,j} \tag{3.5}$$

(Recall that $(\widehat{x}, \widehat{y})$ is the initial fractional solution and $(\tilde{x}, \tilde{y})$ is the solution obtained when the extended median assignment procedure gets stuck.)

(i) follows immediately from the definition of A. (ii) is also easy, since by construction of the extended median assignment procedure (specifically, by definition of the capacity transferring process), if the procedure gets stuck, then we know that all facilities in F are using their full capacity. By (i), they use it only to serve clients in A and it is not enough to satisfy the demand in A, so the inequality follows.

As we can see, sets A and F satisfy the same properties as the equivalent sets in the previous chapter. These properties will be key in the proof of the main theorem.

Intuitively, this is saying that before rounding, there was enough capacity in F to satisfy all the flow requirements corresponding to F and A , but after rounding that is no longer true. However, Lemma 3.3.5 tells us that the capacity available in F before and after rounding is roughly the same. From there, we will derive the contradiction.

We are finally able to prove the main theorem. As we will see, the proof is very similar to that by Levi et al.

Proof of Theorem 3.3.3: Suppose for contradiction that the extended median assignment procedure gets stuck. Let F and A be the sets defined above.

Let us recall that, by definition, $D(A) = (L_A - 1)u + R_A$, with $L_A \in \mathbb{Z}$ and $0 < R_A \leq u$. Similarly, we express the total capacity of the fractionally open facilities over F in the optimal LP solution as $\sum_{i \in F} \widehat{y}_i u = (L-1)u + R$, with $L \in \mathbb{Z}$ and $0 < R \leq u$.

The rest of the proof is structured as follows: First we prove $L \leq L_A$, then $R/u < 0.5$ and then $L = L_A$. After that, we use these facts to derive a contradiction from the flow-cover inequality corresponding to F and A.

Lemma 3.3.6 $L \leq L_A$

Proof:

$$(L-1)u \leq \sum_{i \in F} \tilde{y}_i u < \sum_{j \in A} \sum_{i \in F} z_{i,j} \leq D(A) \leq L_A u.$$

The first inequality follows from the definition of L and from Property (a). The second inequality comes from inequality (3.5). The last two follow from the definitions of flow requirements and of L_A, respectively. Since L and L_A are integers, $L - 1 < L_A$ implies $L \leq L_A$. ■

Lemma 3.3.7 $R/u < \frac{p}{p+1}$.

Proof: Suppose $R/u \geq \frac{p}{p+1}$. By definition of L and R, we have that $\sum_{i \in F} \widehat{y}_i \geq (L - 1) + \frac{p}{p+1}$. From Property (b) of Lemma 3.3.5, the extended random-shift procedure opens at least L facilities in F. In other words, $\sum_{i \in F} \tilde{y}_i u \geq Lu$. We have the following chain of inequalities, which contradicts (3.5).

$$\sum_{i \in F} \sum_{j \in A} z_{i,j} \leq \sum_{i \in F} \sum_{j \in A} \widehat{x}_{i,j} d_j \leq \sum_{i \in F} \widehat{y}_i u = (L-1)u + R \leq Lu \leq \sum_{i \in F} \tilde{y}_i u.$$

The first inequality follows from the definition of flow requirements, the second from the capacity constraints 3.2. The last inequality is the observation we just made above.
Thus, $R/u < \frac{p}{p+1}$. ■

Lemma 3.3.8 $L = L_A$.

Proof: By Lemma 3.3.6, we can suppose for contradiction that $L_A - 1 \geq L$.
The following observation is very important. Let $j \in A$. By definition of A, j has at least one positive flow requirement whose corresponding cluster is a subset of F. By definition

of flow requirements, more than a fraction of $\frac{p}{p+1}$ of the demand of j is satisfied from within F in the optimal LP solution (if it were not so, all flow requirements of j over F would be zero). That is, $\sum_{i\in F}\widehat{x}_{i,j} > \frac{p}{p+1}$. Thus, we have

$$\sum_{i\in F} z_{i,j} = (p+1)d_j\left(\sum_{i\in F}\widehat{x}_{i,j} - \frac{p}{p+1}\right). \tag{3.6}$$

The $-\frac{p}{p+1}$ corresponds to the flow requirements that are zero, all of which are within the interval F.
Adding over all clients in A:

$$\begin{aligned}
\sum_{j\in A}\sum_{i\in F} z_{i,j} &= (p+1)\left(\sum_{j\in A}\sum_{i\in F}\widehat{x}_{i,j}d_j - \frac{p}{p+1}D(A)\right) \leq (p+1)\left(\sum_{i\in F}\widehat{y}_i u - \frac{p}{p+1}D(A)\right)\\
&= (p+1)(L-1)u + (p+1)R - p(L_A-1)u - pR_A\\
&\leq (p+1)(L-1)u + pu - pLu + 0 = ((p+1)L-(p+1)+p-pL)u\\
&= (L-1)u \leq \sum_{i\in F}\tilde{y}_i u.
\end{aligned}$$

The first inequality follows from the capacity constraints (3.2). The second inequality comes from the assumptions $L_A - 1 \geq L$ and $(p+1)R < pu$ (Lemma 3.3.7), as well as from the trivial fact $R_A \geq 0$. The last inequality comes from Property (a).
Again, this contradicts (3.5), so we can conclude $L = L_A$. ■

Thus, let us assume $R/u < \frac{p}{p+1}$ and $L_A = L$. By definition of L and R, we know that the LP opened (fractionally) a total of $(L-1)+R/u > L-1$ facilities in F. This means there are at least L facilities in F, and thus $|F| \geq L = L_A$. Thus, by definition, F is a cover of A. Furthermore, $F \in \mathcal{S}$. So, we know that the corresponding flow-cover inequality is valid for (MIP-CFL) and is thus included in the restrictions of (LP-CFL). That is, the fractional solution $(\widehat{x},\widehat{y})$ must satisfy it:

$$\begin{aligned}
D(A) - \sum_{j\in A}\sum_{i\in F}\widehat{x}_{i,j}d_j &\geq R_A\left(L_A - \sum_{i\in F}\widehat{y}_i\right) = R_A(L_A-(L-1)-R/u)\\
&= R_A(1-R/u) \geq \frac{1}{p+1}R_A.
\end{aligned}$$

Here, we used the flow-cover inequality, the definition of L and R, and the facts we proved above.

Thus, we have $(p+1)(D(A) - \sum_{j\in A}\sum_{i\in F}\widehat{x}_{i,j}d_j) \geq R_A$. Reformulating,

$$(p+1)\left(\sum_{j\in A}\sum_{i\in F}\widehat{x}_{i,j}d_j - \frac{p}{p+1}D(A)\right) \leq D(A) - R_A.$$

By (3.6), the left hand side is equal to $\sum_{j\in A}\sum_{i\in F} z_{i,j}$, and we know the right hand side is equal to $(L_A - 1)u$ by definition of R_A.
So, using this and Property (a) once more,

$$\sum_{j\in A}\sum_{i\in F} z_{i,j} \leq (L_A - 1)u = (L-1)u \leq \sum_{i\in F}\tilde{y}_i u.$$

Again, this contradicts (3.5), and so we are done. ■

3.4 Tightness of the Analysis.

First, we prove that the analysis of the algorithm is tight. Let us consider the following instance:

Let T be a star with $2p$ leaves. On every leaf, there is a facility with capacity one, in the center of the star there is a client with demand 1. The opening costs are all one, and the costs on the edges are all zero. Clearly, any optimal solution to the problem opens exactly one facility. It is also clear that we can express the solution $y = (\frac{1}{p+1}, \ldots, \frac{1}{p+1}, 0, \ldots, 0)$, $x = (\frac{1}{p+1}, \ldots, \frac{1}{p+1}, 0, \ldots, 0)$ (where the first $p+1$ entries are $\dfrac{1}{p+1}$ and the rest are 0 for both x and y) as a convex combination of $p+1$ optimal solutions. So, it is an optimal solution to the LP relaxation. Furthermore, since every optimal solution satisfies the flow-cover inequalities, so does this fractional solution.

The number of paths into which we can partition the facilities is at least p. Multiplying the fractional values by $p+1$, we obtain $\bar{y} = (1, \ldots, 1, 0, \ldots, 0)$, and therefore $\tilde{y} = (1, \ldots, 1, 0, \ldots, 0)$. The value of this solution is $p+1$-times the optimal value, so our analysis is tight.

Furthermore, let us examine the same instance, with the difference that the demand now is 2 instead of 1. In this case, any optimal solution opens exactly two facilities. Similarly to what we had before, the fractional solution $y = (\frac{1}{p}, \ldots, \frac{1}{p})$, $x = (\frac{1}{p}, \ldots, \frac{1}{p})$ is a convex combination of all optimal solutions for the MIP.

The whole idea of the algorithm is based on rounding such that the properties of Lemma 3.3.5 are satisfied, in particular Property (a). We will now show that if we satisfy Property (a), it is impossible to find an approximation algorithm with an approximation guarantee less than $O(p)$ following these ideas.

Suppose that there is an approximation algorithm with an approximation guarantee of k, where k is asymptotically less than $O(p)$. We look at the last instance described, and let $p \to \infty$. Since the optimal solution opens 2 facilities, the algorithm opens at most $2k$ facilities. Since $2k$ is asymptotically smaller than p, for p large enough we have $2k < p$. So, there are at least p facilities that were not opened by the algorithm. Since the fractional value of each of these facilities (before rounding) is $\frac{1}{p}$, their total weight is at least 1. So, by Property (a), there should be at least one open facility in this group in the final solution, since there is a subtree that contains only these facilities. This contradicts their choice.

So, if we want Lemma 3.3.5 to hold (independently of whether we partition into paths or use a different rounding technique), the best we can hope for is an approximation guarantee of $O(p)$.

In fact, the argument above also shows that if the multiplicative factor we use is smaller than or equal to p, we run into a problem. Thus, $p + 1$ is the best possible approximation guarantee for any algorithm that satisfies Lemma 3.3.5.

This is to be expected, since Lemma 3.3.5 is rather strong. In fact, it does not even hold for rounding algorithms for uncapacitated facility location. So, if we want to make use of the lemma, we will need to focus on a restricted class of structures.

3.5 Approximation Factor

In this section, we will briefly describe how we come up with the $p + 1$ approximation factor. Several details are intentionally omitted, since the procedure we follow is exactly the same that we used in the analyses of the algorithms in this and the previous chapter.

There are two places where that number comes into play when computing the objective value. The first and most obvious one is in the extended random-shift procedure. The second one is in the definition of flow requirements. Intuitively, if we use a larger factor in the extended random-shift procedure we open more facilities in smaller neighborhoods, and that means that the median facility in the definition of flow requirements can be closer to the client defining it.

Using the same notation as in the description of the algorithm, suppose we define flow requirements as

$$z_{i_g,j} := \begin{cases} \frac{1}{1-\epsilon}\widehat{x}_{i_g,j}d_j & \text{for } g = M+1,\dots,m, \\ (1 - \sum_{q=M+1}^{m} \frac{1}{1-\epsilon}\widehat{x}_{i_q,j})d_j & \text{for } g = M, \\ 0 & \text{for } g = 1,\dots,M-1, \end{cases}$$

where $c(i_1, j) \leq c(i_2, j) \leq \cdots \leq c(i_m, j)$ and M is the smallest index such that $\sum_{g=1}^{M} x_{i_g,j} \geq \epsilon$. This gives an approximation guarantee of at least $\frac{1}{1-\epsilon}$.

A careful reading of our analysis will reveal that the only part where we actually use this value is in the proof of Lemma 3.3.8. More particularly, where we see that $\sum_{j\in A}\sum_{i\in F} z_{i,j} \leq (L-1)u$. If we substitute $p+1$ for $\frac{1}{1-\epsilon}$ and $\frac{p}{p+1}$ for ϵ and follow the same steps, we get $\sum_{j\in A}\sum_{i\in F} z_{i,j} \leq \ell u - \frac{1}{1-\epsilon}(u-R)$.

If we want this to be upper-bounded by $(\ell-1)u$, we need $\frac{R}{u} < \epsilon$. We want to obtain this inequality from Lemma 3.3.7. It is not hard to see that for this to be true, we need to use a multiplicative factor of $\frac{p}{\epsilon}$ in the extended random-shift procedure.

This means that the opening costs give us an approximation of $\frac{p}{\epsilon}$ and the assignment costs give an approximation of $\frac{1}{1-\epsilon}$. Setting these two values equal yields $\epsilon = \frac{p}{p+1}$, which corresponds to a $(p+1)$-approximation.

3.6 Solving the LP

The starting point of our algorithm is an optimal solution to the LP relaxation that contains all flow-cover inequalities corresponding to subtrees. Here, we will see under what conditions we can do this effectively.

Our approach here is analog to the one used in the previous chapter. By the work of Levi et al.; we know that for a fixed F, flow-cover inequalities can be separated in polynomial time by solving a separation MIP (In the Appendix, we prove this only in context of the lot-sizing problem; but the proof for CFL is exactly the same). In the line case, there is a polynomial (in the number of facilities) number of intervals, which allows us to add all flow-cover inequalities corresponding to intervals of facilities to the LP relaxation and solve it efficiently.

However, in more general trees, $|\mathcal{S}|$ is not necessarily polynomial in the number of facilities, or even leaves (let us recall that $\mathcal{S}$ is the set of subtrees).

Consider for example a star of n leaves, with a facility on each leaf, and none in the center. Clearly, any subset of facilities is in $\mathcal{S}$, which means $|\mathcal{S}| = 2^n$.

However, we have the following result:[3]

Lemma 3.6.1 *Let T be a tree with l leaves and n vertices of degree 2. The number of connected subgraphs of T is at most polynomial on n and exponential in l.*

Proof: It is a well known result in graph theory (the proof of this is a very simple exercise) that a tree with l leaves has at most $l - 2$ vertices with degree three or more. Thus, it has at most $2l - 2$ vertices of degree different than 2. Let S denote this set, and let N denote the set of vertices of degree 2. So, $|S| \leq 2l - 2$ and $|N| = n$.

Now, given $S_1 \subseteq S$, we say S_1 *determines a subtree* if there exists a subtree T_1 with vertices V_1 such that $S \cap V_1 = S_1$. Given some S_1, we would like to know how many subtrees does S_1 determine. We also make a slight abuse of notation and define the *convex hull* of a vertex subset S' as the set of vertices in the smallest subtree determined by S'.

First, all the vertices in N contained in the convex hull of S_1 must clearly belong to any subtree determined by S_1. Let us call the subtree obtained by taking this convex hull the *core subtree* of S_1. Now, it is not hard to see that any edge that has a vertex in this core subtree and a vertex outside it, satisfies that the vertex in the core subtree actually belongs to S_1. We move along the path determined by this edge away from the subtree until right before we find a vertex in S. We call the path determined by all the visited vertices a *branch*. Clearly, all vertices in this branch belong to N. Also, if we want to extend the core subtree along this direction, we can take at most as many vertices as the branch has, because if we include a vertex in S, then S_1 will no longer determine the resulting subtree. Trivially, the branch has at most n vertices, so there are at most n ways of extending the core subtree along this direction. Also trivially, there exist at most l branches, so we have at most n^l ways of extending the core subtree into another subtree that is also determined by S_1.

Now, there are 2^{2l-2} subsets of S, and every subtree is determined by one of these subsets. Combining the two arguments, this gives us an upper bound of $2^{2l-2}n^l$ on the total number of subtrees, which is what we wanted. ∎

The upper bound given in the proof is very loose, but it is enough for our purposes. That is, if we impose an upper bound on the number of leaves (or facilities), our algorithm

[3] At this point, it is useful to notice that we can assume without loss of generality that all facilities are located on vertices of degree 1 or 2 (since otherwise we can move them along an edge by ϵ and this gives us only a slightly larger approximation guarantee).

gives us a constant factor approximation and we can solve the extended LP in polynomial time.

3.7 Separation

The weakness of the previous algorithm is that to have a guarantee of a polynomial time running time, we need the number of leaves to be bounded. However, even if this is not the case, there is a way of going around this issue by using the ellipsoid method. Indeed, it is known that given a polynomial time separation oracle, an LP with an exponential number of constraints can be solved in polynomial time using the ellipsoid method (See [7]).

Here, we will use the following separation oracle:

0. Let $(\widehat{x}, \widehat{y}) \in \mathbb{R}^{|\mathcal{F}|+|\mathcal{F}|\cdot|\mathcal{C}|}$.

1. Check validity of the constraints of the natural relaxation of (MIP-CFL) individually. If there is a violated inequality, output it and stop. Otherwise, go to 2.

2. Run the extended random-shift and extended median assignment procedures. If it gets stuck, output the corresponding violated flow-cover inequality. Otherwise, $(\widehat{x}, \widehat{y})$ is feasible.

We should note that the solution returned by running the ellipsoid method with this separation oracle does not necessarily satisfy all flow-cover inequalities corresponding to subtrees. That is, we use the ellipsoid method to solve an LP of which we do not know all its constraints in advance. Yet, since the extended median assignment procedure can be completed for the returned solution; the final iteration of the separation oracle actually gives us a rounded solution that is feasible for (MIP-CFL) and has cost at most $(p+1)OPT$.

So, we only need to prove that the separation oracle can be solved in polynomial time. Since the natural LP relaxation has a polynomial number of constraints, we do not need to worry about step 1.

Next, we show that we for a solution of the natural LP relaxation, we can identify a violated flow-cover inequality (if there exists one) in polynomial time by running our algorithm. Using the notation introduced in Section 3.2 above, let F^0 be a cluster corresponding to a frustrated flow requirement. The algorithm iteratively transfers capacity from open facilities to F^0 until the flow requirement is satisfied. By the way the algorithm is constructed, it is clear that in every iteration (except for the last one, when the flow requirement is finally satisfied) either the facility from which we are transferring capacity gets saturated (which means it uses all its capacity) or it does not. If it does not get saturated, it means that in the process it got disconnected from F^0.

In any case, either one facility that previously had free capacity gets saturated, or the connected component corresponding to F^0 gets smaller. Since the algorithm always transfers capacity towards F^0 and not away from it, we know the connected component never grows. So, since the size of the connected component is linear in the number of facilities and clients, and the number of facilities with free capacity is also linear, the separation oracle can be solved in polynomial time.

We can note that if we follow this approach, we do not need the separation result for flow-cover inequalities that we prove in the Appendix.

As we can see, this approach is very similar to the on-the-fly variant defined in the previous chapter. The main difference is that the on-the-fly algorithm does not necessarily rely on the ellipsoid method to solve the LP in each iteration, and so could be more efficient in practice, but has no big theoretical value.

3.8 Comments

- We can note that we are not simply decomposing the problem into p smaller problems and then merging the solutions. We use the decomposition only in the first part of the algorithm, and then for the analysis of the second part. Each of these paths does not necessarily induce a feasible problem. For example, it might be the case that all clients are on one path and that path contains only one facility, such that just that facility does not have enough capacity to satisfy the whole demand.

- As we may recall, the algorithm relies on partitioning the tree into paths that may intersect at client-vertices but not at facility-vertices. Clearly, the addition of this last constraint increases the minimum number of paths in a feasible partition. A good example is a star graph with a facility on each vertex and an even number of leaves, ℓ. If we were allowed to use the central vertex in several paths, we could partition the graph into $\ell/2$ paths. However, the restriction on facility-vertices does not let us partition into less than $\ell - 1$ paths, thus doubling the approximation guarantee. If we assume that there are no facilities at vertices of degree 3 or more, we do not have this problem. We can in fact safely assume that, because given a facility on a vertex of degree 3 or more, we can move it along any incident edge by some distance $\epsilon > 0$. For ϵ small enough, the effect on the optimal value is negligible.

- It is known that finite metrics can be probabilistically approximated by tree metrics with a distortion of at most $O(\log(n))$, where n is the number of elements in the metric space (See [20], pp. 211-216). In our case, this gives us a randomized $O(|\mathcal{F}|\log(|\mathcal{F}\cup\mathcal{C}|))$-approximation for the problem on general metric spaces. It is an interesting result in the sense that it tells us that if we want to find an instance that has a bad integrality gap after adding the flow-cover inequalities, we need a large number of facilities and/or a really large number of clients.

3.9 Generalization to Multi-Commodity Capacitated Facility Location with Monotone Costs on Trees

As a final comment, we show in this section that the results of the chapter can be extended to a more general framework.

Let us consider the following generalization of the problem we have been working on.

First, we will assume that there is a given set of N items, that each client j has a demand d_{ij} of item i, and that each facility can serve at most a total of u units of demand, regardless of the types of items. That is, we consider the multi-commodity variant, just like we did in Chapter 2 (the convention is to say *multi-commodity* facility location instead of *multi-item* facility location).

Second, we substitute the linear assignment costs by monotonous costs. By this, we mean the following. c is such that if facility i_2 lies on the unique path between facility i_1 and client j, then for every item k we have $c^k_{i_2j}\leq c^k_{i_1j}$.

We will call this problem the *multi-commodity capacitated facility location problem with monotonous costs on trees*, and we abbreviate it as MCCFL.

There has been some progress in the literature on different variants of multi-commodity uncapacitated facility location (see [16], [17] or [18]), but to the best of our knowledge, this specific problem has not been studied, not even for metric assignment costs or non-uniform capacities.

It is not hard to see that the algorithm can be very easily adapted to this more general problem, and that the analysis holds as well. Just as we did for the lot-sizing problem, where we previously defined a flow requirement corresponding to a client and a facility, now we define N flow requirements, one for every item.

The extended random-shift procedure can be carried out just as before, since it does not depend on the assignment costs or the types of items. Of course, when reassigning the demands, we consider chains whose clusters all correspond to the same item type. And clearly, by the monotonicity of the assignment costs, if we can satisfy all flow requirements, then we have the same approximation guarantee as before. This is the only part of the proof where we use the properties of the assignment costs. Proving that the flow requirements can be satisfied is done exactly as we did.

So, we have

Theorem 3.9.1 *The extended random-shift with extended median assignment procedure gives a randomized* $(p+1)$*-approximation to MCCFL.*

Furthermore, we can prove NP-hardness of this problem just like we did for the lot-sizing problem (while we do not know the complexity of the problem with a single item and tree metric costs). For the details, see Section A.2 in the Appendix.

Chapter 4

A Primal-Dual Algorithm for the Capacitated Facility Location Problem

In this chapter, we will start by working with a special case of the capacitated facility location problem, present an algorithm designed for it by Carnes and Shmoys; and then move on to the general problem and present our generalization.

We will take a similar approach to the problem as in the previous chapters. We will also strengthen the LP to reduce the integrality gap, but we will use a different type of inequalities. Also, instead of rounding a solution of the LP relaxation, we will present a primal-dual algorithm.

In this chapter, we define $\mathcal{F}$, $\mathcal{C}$, f_i, c_{ij}, d_j and u_i for $i \in \mathcal{F}$, $j \in \mathcal{C}$, as in the introduction. Note that, as opposed to what we did in Chapter 3, we do not assume uniformity of the capacities here.

4.1 The Single-Demand Case

This section is an overview of one of the results in [3], which we will generalize in the next section.

4.1.1 The Problem

We will first focus on the single-demand case of the CFL problem. That is, we will assume that there is only one client in $\mathcal{C}$, let d be its demand. As seen in the introduction, this variant of the problem also suffers from an unbounded integrality gap.

The problem is also NP-hard, because it generalizes the *minimum knapsack* problem (by setting the assignment costs to zero), which is well known to be NP-hard.

4.1.2 Formulation

Let us start by introducing the flow-cover inequalities we will use in this chapter. Although they share the name and are similar in nature to the ones used in the previous chapters, they are also significantly different. To avoid confusions, when we refer to *flow-cover* inequalities in this chapter, we will mean only those we are about to present.

We first give the MIP formulation of this simplified problem:

$$
\text{(MIP-SD)} \qquad
\begin{aligned}
\min \quad & \sum_{i\in\mathcal{F}} f_i y_i + \sum_{i\in\mathcal{F}} dc_i x_i & & \\
\text{s.t} \quad & \sum_{i\in\mathcal{F}} x_i = 1 & & (4.1)\\
& dx_i \le u_i y_i & \text{for every } i \in \mathcal{F} \qquad & (4.2)\\
& x_i \ge 0 & \text{for every } i \in \mathcal{F} \qquad & \\
& y_i \in \{0,1\} & \text{for every } i \in \mathcal{F}. \qquad &
\end{aligned}
$$

Given a set $A \subseteq \mathcal{F}$, let $d(A) = \max\{0, d - \sum_{i\in A} u_i\}$. That is, $d(A)$ is the remaining demand after we have completely used all the capacity of the facilities in A.

We also define $u_i(A) = \min\{u_i, d(A)\}$ for every $i \notin A$. This is called the *effective* (or *residual) capacity* of facility i. That is, by how much i can contribute to satisfy the demand given that the facilities in A are already serving the client with all their capacity.

Furthermore, let $\mathcal{P} = \{(F_1, F_2, A) | F_1, F_2, A \text{ partition } \mathcal{F}\}$.

Now we are ready to state the flow-cover inequality corresponding to $(F_1, F_2, A) \in \mathcal{P}$:

$$\sum_{i\in F_1} dx_i + \sum_{i\in F_2} u_i(A) y_i \ge d(A). \tag{4.3}$$

Let us understand why this inequality is true intuitively. We know A, F_1 and F_2 are a partition of $\mathcal{F}$. Suppose facilities in A are all open and fully used to satisfy the demand. Then, the remaining demand must be satisfied from facilities in F_1 and F_2. The first term is the amount of demand served from F_1, and the second term is the total effective capacity in F_2, which upper-bounds the demand served from F_2. We will prove this in detail below.

For us, the importance of this inequality lies in its use of effective capacities, because that will allow us to close the integrality gap. For example, the instance seen in the introduction that shows an unbounded integrality gap of the natural LP relaxation does not satisfy this flow-cover inequality. Intuitively, in that solution we are opening only a small fraction of the expensive facility because its capacity is large and that small fraction of the large capacity is enough to satisfy the remaining demand (after opening the other facility). However, the effective capacity of that facility is small, so the solution violates inequality (4.3).

Proposition 4.1.1 *Inequalities* (4.3) *are valid for all feasible points of* (MIP-SD)*.*

Proof: The validity of (MIP-SD) is not hard to see. Given some partition (F_1, F_2, A), we are interested in how much demand is required from F_1 and F_2. Let x be a feasible solution. By definition, the demand remaining after the facilities in A are used is at most $d(A)$. So,

$$d(A) \leq \sum_{i \notin A} dx_i = \sum_{i \in F_1} dx_i + \sum_{i \in F_2} dx_i.$$

At this point, we could use restriction (4.2) applied to the second sum to get a weaker version of the inequality. However, we can do better. Since both the capacity of a facility $i \in F_2$ and the remaining demand $d(A)$ are upper bounds on the size of the demand served from i to the client, by definition of $u_i(A)$ we can decrease the right hand side coefficient in (4.2) to obtain $dx_i \leq u_i(A)y_i$. Substituting above, the result follows. ■

These inequalities are in fact very strong; as they imply restrictions (4.1) and (4.2) of (MIP-SD) for feasible and optimal solutions of (MIP-SD), respectively. Indeed, by substituting $(F_1, F_2, A) = (\mathcal{F}, \emptyset, \emptyset)$ and dividing by d, we get (4.1). Similarly, setting $(F_1, F_2, A) = (\mathcal{F} \backslash \{i\}, \{i\}, \emptyset)$, and using equality (4.1), we get restriction (4.2). We are not going to use these facts in our work, but they are still interesting observations.

By Proposition 4.1.1, we can say that the following is a relaxation of (MIP-SD).

$$
\text{(LP-SD)}\qquad
\begin{array}{lll}
\min & \displaystyle\sum_{i\in\mathcal{F}} f_i y_i + \sum_{i\in\mathcal{F}} dc_i x_i & \\
\text{s.t} & \displaystyle\sum_{i\in F_1} dx_i + \sum_{i\in F_2} u_i(A) y_i \geq d(A) & \text{for every } (F_1, F_2, A) \in \mathcal{P} \qquad (4.4) \\
 & x_i, y_i \geq 0 & \text{for every } i \in \mathcal{F}
\end{array}
$$

Note that we are not including the restrictions used before. Since our goal is a primal-dual algorithm, we would like to keep the LP as simple as possible, and (LP-SD) is strong enough for a 2-approximation.

The dual problem looks as follows:

$$
\text{(D-SD)}\qquad
\begin{array}{lll}
\min & \displaystyle\sum_{(F_1,F_2,A)\in\mathcal{P}} d(A) v(F_1, F_2, A) & \\
\text{s.t} & \displaystyle\sum_{\substack{(F_1,F_2,A)\in\mathcal{P}:\\ i\in F_1}} v(F_1, F_2, A) \leq c_i & \text{for every } i \in \mathcal{F} \qquad (4.5) \\
 & \displaystyle\sum_{\substack{(F_1,F_2,A)\in\mathcal{P}:\\ i\in F_2}} u_i(A) v(F_1, F_2, A) \leq f_i & \text{for every } i \in \mathcal{F} \qquad (4.6) \\
 & v(F_1, F_2, A) \geq 0 & \text{for every } (F_1, F_2, A) \in \mathcal{P}
\end{array}
$$

It will be useful to notice that for a fixed $i \in \mathcal{F}$ and fixed $(F_1, F_2, A) \in \mathcal{P}$, either $v(F_1, F_2, A)$ appears in constraint (4.5) corresponding to i (if $i \in F_1$), or it appears in constraint (4.6) corresponding to i (if $i \in F_2$), or it does not appear in either (if $i \in A$).

4.1.3 The Algorithm

Here we describe the algorithm given by Carnes and Shmoys in [3].

We initialize all primal and dual variables to zero, and define the sets $F_1 = \mathcal{F}$, $F_2 = \emptyset$, $A = \emptyset$. Throughout the algorithm, we increase the dual variable corresponding to the current sets F_1, F_2, A until a dual constraint becomes tight. If it is a constraint of type (4.5), corresponding to facility i, then that means that until this point, facility i belonged to set F_1 (because in constraint (4.5) we sum only over dual variables with $i \in F_1$). In this case, we move i from F_1 to F_2. If the tight constraint is of type (4.6) corresponding to facility i, then it means that at this point i belongs to F_2; and we move it from F_2 to A. In this way, we make sure that if, for a given $i \in \mathcal{F}$, both constraints are tight at some point, then the first one to become tight was (4.5).

When a facility is moved to A, we open it and serve demand from it greedily. When $d(A)$ becomes 0, we stop.

We summarize it in Algorithm 2 below.

Algorithm 2

```
**Initialize** $x, y, v := 0$, $F_1 := \mathcal{F}$, $F_2, A := \emptyset$.
**while** $d(A) > 0$ **do**
  Increase $v(F_1, F_2, A)$ until a dual constraint goes tight and prevents it from growing more.
  **if** this constraint is of type (4.5) corresponding to $i \in \mathcal{F}$, **then**
    $F_1 := F_1 \backslash \{i\}$, $F_2 := F_2 \cup \{i\}$
  **else**
    $y_i := 1$, $x_i := u_i(A)/d$
    $F_2 := F_2 \backslash \{i\}, A := A \cup \{i\}$
  **end if**
**end while**
**return** $x, y, v$.
```

4.1.4 Analysis

Before starting the formal analysis, let us give an intuitive interpretation of the algorithm.

For every facility i, the right hand side of constraint (4.5) is the assignment cost corresponding to that facility, and the right hand side of constraint (4.6) is its opening cost. So, in the spirit of the classical primal-dual algorithm for uncapacitated facility location by Jain and Vazirani (see [10]), we can picture the v variables as "money" that the client pays to have its demand satisfied. For a facility to serve it, the facility needs to be *connected* to the client and it also needs to be *open*. In the algorithm, the client starts simultaneously paying the connection costs to all facilities. As soon as one of them becomes connected, the client starts paying towards its opening cost (this is represented by moving i from F_1 to F_2), while it keeps paying for the connection costs of facilities that are not yet connected. Once the opening cost has been paid for, we open the facility and use as much of its capacity as possible. The coefficient $u_i(A)$ in constraint (4.6) intuitively means that if a facility has a large effective capacity, then it is more attractive to the client than facilities with less effective capacity. So, the amount of money per unit of opening cost that the client needs to pay is smaller for facilities with larger effective capacity (We can think of the $u_i(A)$ coefficients as "discounts" corresponding to that facility. The more useful the facility, the bigger the discount.). This interpretation is not necessary for the analysis, but it helps understand the reasoning behind the algorithm. It will be especially useful for the analysis of the algorithm in the next section.

The fact that we open a facility only if its two corresponding constraints are met at equality will help us in the analysis, through the use of complementary slackness.

Theorem 4.1.2 *Algorithm 2 gives a 2-approximation for the single demand capacitated facility location problem.*

Proof: Let z_P and z_D represent respectively the values of the primal and dual solutions returned by the algorithm. We will prove that $z_P \leq 2z_D$, this clearly implies the result.

Let S be the set of facilities with $y_i = 1$, and let ℓ be the last facility opened. By definition of ℓ, and by the greedy nature of the algorithm, all facilities in $S \backslash \{\ell\}$ use their full capacity to satisfy the demand of the client. So, for any $(F_1, F_2, A) \in \mathcal{P}$, we have

$$\sum_{i \in F_2 \cap S \backslash \{\ell\}} u_i + \sum_{i \in F_1 \cap S \backslash \{\ell\}} dx_i \leq \sum_{i \in F_2 \cap S \backslash \{\ell\}} u_i + \sum_{i \in F_1 \cap S \backslash \{\ell\}} u_i \leq d(A).$$

Adding this and the trivial inequality $u_\ell(A) \leq d(A)$, we get

$$\sum_{i \in F_2 \cap S} u_i + \sum_{i \in F_1 \cap S} dx_i \leq 2d(A), \tag{4.7}$$

since no matter if ℓ belongs to F_1 or F_2, its corresponding term is upper-bounded by $u_\ell(A)$.

Now we are ready to prove the approximation guarantee.

$$\begin{aligned}
z_P &= \sum_{i \in S} f_i + \sum_{i \in S} c_i dx_i \\
&= \sum_{i \in S} \sum_{\substack{(F_1, F_2, A) \in \mathcal{P}: \\ i \in F_2}} u_i(A) v(F_1, F_2, A) + \sum_{i \in S} dx_i \sum_{\substack{(F_1, F_2, A) \in \mathcal{P}: \\ i \in F_1}} v(F_1, F_2, A) \\
&= \sum_{(F_1, F_2, A) \in \mathcal{P}} v(F_1, F_2, A) \left(\sum_{i \in F_2 \cap S} u_i(A) + \sum_{i \in F_1 \cap S} dx_i \right) \\
&\leq 2d(A).
\end{aligned}$$

Here, we used the fact that we open a facility only if its corresponding constraints are both tight, then we changed the order of summation, and finally we used (4.7).

■

4.2 The General Case

In this section, we will generalize the previous algorithm for the case of an arbitrary number of clients.

4.2.1 Formulation

Let $\mathcal{F}$, $\mathcal{C}$ and $\mathcal{P}$ be defined as before. The flow-cover inequalities we will use in this section depend not only on a partition of $\mathcal{F}$ but also on a subset of $\mathcal{C}$. To this end, we define, for $A \subseteq \mathcal{F}$ and $B \subseteq \mathcal{C}$,

$$d(A, B) = \max\{0, \sum_{j \in B} d_j - \sum_{i \in A} u_i\}.$$

That is, $d(A, B)$ represents the demand remaining in B after all facilities in A have been fully used to satisfy it.

Similarly, let $u_i(A, B) = \min\{u_i, d(A, B)\}$ for every $i \in \mathcal{F}, A \subseteq \mathcal{F}$ and $B \subseteq \mathcal{C}$.

The following proposition proves the validity of the generalized flow-cover inequalities that we will use:[1]

Proposition 4.2.1 *For every $(F_1, F_2, A) \in \mathcal{P}, B \subseteq \mathcal{C}$, the following is valid for (MIP-CFL):*

$$\sum_{i \in F_1} \sum_{j \in B} d_j x_{ij} + \sum_{i \in F_2} u_i(A, B) y_i \geq d(A, B). \tag{4.8}$$

The proof is completely analogous to that of (4.1.1), and will be omitted.

Thus, the LP relaxation we will work with is

[1] Actually, these inequalities were used by Carnes and Shmoys in a primal-dual algorithm for the lot-sizing problem, which is the main result of [3].

$$\begin{aligned}
&\min && \sum_{i\in\mathcal{F}} f_i y_i + \sum_{i\in\mathcal{F}}\sum_{j\in\mathcal{C}} c_{ij} d_j x_{ij} \\
\text{(LP-CFL2)}\quad &\text{s.t.} && \sum_{i\in F_1}\sum_{j\in B} d_j x_{ij} + \sum_{i\in F_2} u_i(A,B) y_i \geq d(A,B) \qquad \forall (F_1,F_2,A)\in\mathcal{P}, B\subseteq\mathcal{C} \\
& && x, y \geq 0.
\end{aligned}$$

Before writing the dual problem, we will make a small abuse of notation to simplify reading, and abbreviate $\sum_{\substack{(F_1,F_2,A)\in\mathcal{P},\\ B\subseteq\mathcal{C}}}$ as $\sum_{v}$. Similarly, $\sum_{\substack{v:i\in F_1,\\ j\in B}}$ stands for $\sum_{\substack{(F_1,F_2,A)\in\mathcal{P},B\subseteq\mathcal{C}:\\ i\in F_1, j\in B}}$, etc.

Using these conventions, the dual of (LP-CFL2) is

$$\begin{aligned}
&\max && \sum_{v} d(A,B) v(F_1,F_2,A,B) \\
&\text{s.t.} && \sum_{\substack{v:i\in F_1,\\ j\in B}} v(F_1,F_2,A,B) \leq c_{ij} && \text{for every } i\in\mathcal{F}, j\in\mathcal{C} \qquad (4.9) \\
\text{(D-CFL)}\quad & && \sum_{v:i\in F_2} u_i(A,B) v(F_1,F_2,A,B) \leq f_i && \text{for every } i\in\mathcal{F} \qquad (4.10) \\
& && v \geq 0.
\end{aligned}$$

4.2.2 The Algorithm

Our generalization to the algorithm of Carnes and Shmoys is the following:

We initialize all v variables to zero. Throughout the algorithm, we say that facility i and client j are *connected* if the corresponding constraint (4.9) is tight, and we say that facility i is *open* if the corresponding constraint (4.10) is tight.

At every point in the algorithm, the set of clients is partitioned into sets B^k. Initially, the sets B^k are all singletons $\{j\}$. As the algorithm progresses, we will iteratively merge these sets while keeping a partition of $\mathcal{C}$. At every moment, for every set B^k we consider a triple $(F_1^k, F_2^k, A^k) \in \mathcal{P}$; where F_1^k is the set of facilities that are not connected to any client in B^k, A^k is the set of facilities that are open and connected to at least one client in B_k, and F_2^k is the rest of the facilities. It is useful to remember that the set B^k and the current primal and dual solutions completely determine the sets F_1^k, F_2^k and A^k. As we will see in more detail later, for every B^k, A^k is a set of open facilities that serve only clients in B^k. So, very roughly speaking, we will start constructing solutions locally, and gradually, as these solutions intersect, we will merge them to obtain more global solutions. Of course, initially we have $F_1^k = \mathcal{F}, F_2^k = A^k = \emptyset$ for every k.

We increase all variables $v(F_1^k, F_2^k, A^k, B^k)$ simultaneously until a dual constraint goes tight. When that happens, we update F_1^k, F_2^k and A^k. If at some point a facility is connected to two clients, we merge their corresponding sets B^{k_1} and B^{k_2} to form B^k, and compute F_1^k, F_2^k, A^k as described above.

At all times, we serve demand greedily from facilities in A^k to clients in B^k for every k. Even if at some point we have for some k that $d(A^k, B^k) = 0$, we do not freeze the corresponding dual variable (If we do it and reactivate it when B^k is merged to some other set that still has unsatisfied demand, the result is the same, but not freezing it makes the analysis easier). That is, we continue increasing the dual variables corresponding to the currently active sets B^k until $d(A^k, B^k) = 0$ for every k, at which point we stop the algorithm and output all open facilities and assignments as determined by the algorithm. We summarize this in Algorithm 3. To clarify the procedure, we define a set K which contains the indices of the currently active sets B^k. Of course, initially we have $K = \mathcal{C}$, but as the algorithm progresses, we merge sets B^k and the size of K decreases. We also define $F_2(i) = \{k \in K | i \in F_2^k\}$ for every $i \in \mathcal{F}$.

Note that if we have $d(A^k, B^k) = 0$ and we keep increasing the dual variable corresponding to that B^k, new facilities may be connected to this set, but no new facilities will be opened (unless they have cost 0), because $u_i(A^k, B^k) \leq d(A^k, B^k) = 0$ for every $i \notin A^k$.

We can notice that if $|\mathcal{C}| = 1$, the algorithm described above is precisely the one in the first section of this chapter.

It is not hard to understand the intuition behind this algorithm. Given a set of clients B^k, we know that the facilities that get connected first to B^k are the closest ones; and of those that are connected, the ones that have the largest effective capacity (relative to that cluster) are opened first. So, for every cluster we keep a set A^k of open facilities that are "the best" for it, and we assign its demand to them. Once a facility is connected to two clusters, it makes sense to merge them, because both are paying simultaneously to open that facility. If we merge, we do not have to worry later about how to split the capacity of a facility if it was opened by several clusters. Of course, not merging could also lead to a good approximation algorithm, but we could not find a way to make that work.

Algorithm 3

```
Initialize $x, y, v := 0$, $F_1^k := \mathcal{F}, F_2^k := \emptyset, A^k := \emptyset, B^k := \{k\}$ for every $k \in \mathcal{C}$, $K := \mathcal{C}$.
while $\exists k' \in K$ such that $d(A^{k'}, B^{k'}) > 0$ do
  For every $k \in K$, increase $v(F_1^k, F_2^k, A^k, B^k)$ until a dual constraint goes tight.
  Update $F_1^k, F_2^k$ and $A^k$ for every $k \in K$.
  for $i \in \mathcal{F}$ do
    if $|F_2(i)| \geq 2$ then
      $B^{\bar{k}} := \bigcup_{k \in F_2(i)} B^k$ (for some index $\bar{k} \notin K$).
      Update $F_1^{\bar{k}}, F_2^{\bar{k}}$ and $A^{\bar{k}}$.
      $K := (K \backslash F_2(i)) \cup \{\bar{k}\}$.
    end if
  end for
  For every $k \in K$, $i \in A^k$, let $y_i := 1$. Serve demand greedily from the free capacity in
  $A^k$ to clients in $B^k$ and update $x$.
end while
return $x, y, v$.
```

4.2.3 Analysis

When the algorithm terminates, each of the sets B^k active at that moment determines a *cluster* of clients and facilities connected to these clients, such that the clusters are pairwise disjoint and the open facilities in every cluster provide for all the demand that the clients in the same cluster need, and do not serve clients in any other cluster. First, we will analyze these clusters separately. We will upper bound each cluster's cost using only the dual variables corresponding to it. Then, we will put the pieces together for a $2n$-approximation.

We will start by making some observations on the running of the algorithm. Since we raise the dual variables simultaneously and do not stop increasing until the algorithm terminates, it is clear that for every $j, k \in \mathcal{C}$, we have

$$\sum_{\substack{v: j \in B, \\ k \notin B}} v(F_1, F_2, A, B) = \sum_{\substack{v: k \in B, \\ j \notin B}} v(F_1, F_2, A, B). \tag{4.11}$$

We can interpret this number as the "time" it takes for the sets B containing j and k to merge.

Let $j, k \in \mathcal{C}$ and $i \in \mathcal{F}$. Furthermore, assume that after termination of the algorithm, i is open and connected to both j and k. Without loss of generality, assume $c_{ij} \leq c_{ik}$. Since the connection costs are paid at unit rate, the clusters corresponding to j and k merge precisely at time c_{ik}, and this time is precisely the number on both sides of (4.11). So,

Observation 4.2.2 *If after termination of the algorithm facility i is connected to clients j and k, then*

$$\{v(F_1, F_2, A, B) > 0 : i \in F_1; j, k \in B\} = \emptyset.$$

The following lemma is crucial for the analysis. Let us note that, since all facilities and clients are points in a metric space, the distance between clients is well defined.

Lemma 4.2.3 *Let j and k be two clients in the same cluster. If there is a facility connected to both, then,*

$$c_{jk} \leq 2 \sum_{\substack{v: j \in B, \\ k \notin B}} v(F_1, F_2, A, B) = 2 \sum_{\substack{v: k \in B, \\ j \notin B}} v(F_1, F_2, A, B).$$

Proof: Let i be the first facility that is connected to both j and k. Without loss of generality, assume $c_{ij} \leq c_{ik}$. As we remarked above, c_{ik} is equal to the number on both sides of (4.11). So,

$$c_{jk} \leq c_{ij} + c_{ik} \leq 2c_{ik} = 2 \sum_{\substack{v: k \in B, \\ j \notin B}} v(F_1, F_2, A, B) = 2 \sum_{\substack{v: j \in B, \\ k \notin B}} v(F_1, F_2, A, B).$$

■

This lets us prove a more general result.

Lemma 4.2.4 *Let j and k be two clients in the same cluster, not necessarily connected to the same facility. Let n be the number of clients in $\mathcal{C}$. Then,*

$$c_{jk} \leq (2n-2) \sum_{\substack{v: j \in B, \\ k \notin B}} v(F_1, F_2, A, B) = (2n-2) \sum_{\substack{v: k \in B, \\ j \notin B}} v(F_1, F_2, A, B)$$

Proof: Since j and k belong to the same set B, there is a chain of clients $j = j_0, j_1, \ldots, j_m = k$ such that any 2 consecutive clients have a facility connected to both. Thus, Lemma 4.2.3 applies to all these pairs. Applying it inductively, and using $m \leq n$, we get the desired result. ■

The intuition is that if j is being served by a facility i that is connected only to k, then this lemma allows us to bound c_{jk} by something in terms of dual variables corresponding only to j, so that k does not have to pay for the distance the demand travels from j to k. c_{ik} can be easily bounded using the dual, and this will allow us to prove the main theorem.

One of the main difficulties of the CFL problem is that if we open facilities and do not use all their capacity, we are "wasting" that capacity. In our case, however, the greedy nature of the algorithm guarantees that in every cluster, there will be at most one open facility that does not use all its capacity. To avoid unnecessary notation, let us assume for the moment that all open facilities and clients belong to a single cluster.

Let ℓ be the last facility opened. By our assumption, this is the only facility that may not be serving up to its full capacity. In the spirit of [3], we will first give a trivial upper bound on the cost incurred by opening it.

Lemma 4.2.5

$$f_\ell \leq \sum_{v:\ell\in F_2} d(A,B)v(F_1,F_2,A,B)$$

Proof:

$$f_\ell = \sum_{v:\ell\in F_2} u_\ell(A,B)v(F_1,F_2,A,B) \leq \sum_{v:\ell\in F_2} d(A,B)v(F_1,F_2,A,B).$$

■

Considering this facility separately from the others will let us do the following:

Observation 4.2.6 *For $v(F_1,F_2,A,B) > 0$ and $i \in F_2\backslash\{\ell\}$, we have $u_i(A,B) \leq \sum_{j\in\mathcal{C}} d_j x_{ij}$.*

Since i is using its full capacity, the validity of this observation is very easy to check.

Let S be the set of open facilities after termination of the algorithm. For every $k \in \mathcal{C}$, we define S_k as the set of open facilities that were connected to k before they were connected to any other client. Let us also assume that the sets S_k are pairwise disjoint (If they are not, we arbitrarily make them small enough that they become disjoint, while keeping $\bigcup_k S_k = S$). Thus, they partition S.

The next observation is clear from the definitions.

Observation 4.2.7 *Let $v(F_1,F_2,A,B) > 0$, $k \notin B$. Then, $S_k \subseteq F_1$. This implies $S_k = S_k\backslash A$ and $F_2 \cap S_k = \emptyset$.*

All we need now to prove our main theorem is the following:

Proposition 4.2.8

$$\sum_{i\in S\setminus\{\ell\}} f_i + \sum_{i\in S}\sum_{j\in\mathcal{C}} d_j x_{ij} c_{ij} \le (2n-1)\sum_{v} v(F_1,F_2,A,B)d(A,B).$$

Proof:

$$\sum_{i\in S\setminus\{\ell\}} f_i + \sum_{i\in S}\sum_{j\in\mathcal{C}} d_j x_{ij} c_{ij} = \sum_{i\in S}\Big[f_i + \sum_{j\in\mathcal{C}} d_j x_{ij} c_{ij}\Big] - f_\ell$$

$$= \sum_{k\in\mathcal{C}}\sum_{i\in S_k}\Big[f_i + \sum_{j\in\mathcal{C}} d_j x_{ij} c_{ij}\Big] - f_\ell \le \sum_{k\in\mathcal{C}}\sum_{i\in S_k}\Big[f_i + \sum_{j\in\mathcal{C}} d_j x_{ij} (c_{ik}+c_{jk})\Big] - f_\ell$$

$$\le \sum_{k\in\mathcal{C}}\sum_{i\in S_k}\Big[\sum_{v:i\in F_2\setminus\{\ell\}} u_i(A,B)v(F_1,F_2,A,B)$$

$$+\sum_{j\in\mathcal{C}} d_j x_{ij}\Big(\sum_{\substack{v:i\in F_1,\\ k\in B}} v(F_1,F_2,A,B) + (2n-2)\sum_{\substack{v:j\in B,\\ k\notin B}} v(F_1,F_2,A,B)\Big)\Big]$$

$$\le \sum_{k\in\mathcal{C}}\sum_{i\in S_k}\Big[\sum_{v:i\in F_2} v(F_1,F_2,A,B)\sum_{j\in\mathcal{C}} d_j x_{ij}$$

$$+\sum_{j\in\mathcal{C}} d_j x_{ij}\Big(\sum_{\substack{v:i\in F_1,\\ k\in B}} v(F_1,F_2,A,B) + (2n-2)\sum_{\substack{v:j\in B,\\ k\notin B}} v(F_1,F_2,A,B)\Big)\Big]$$

$$\le \sum_{k\in\mathcal{C}}\sum_{i\in S_k}\sum_{j\in\mathcal{C}} d_j x_{ij}\Big(\sum_{v:i\in F_2} v(F_1,F_2,A,B) + \sum_{\substack{v:i\in F_1,\\ k\in B}} v(F_1,F_2,A,B)$$

$$+(2n-2)\sum_{\substack{v:j\in B,\\ k\notin B}} v(F_1,F_2,A,B)\Big)$$

For the second inequality, we used the facts that all considered facilities are open, and that i is connected to k; as well as Lemma 4.2.4. For the third inequality, we use Observation 4.2.6, and the trivial inequality $0 \le \sum_{j\in\mathcal{C}} d_j x_{\ell j}$. Then, we reverse the order of summation.

At this point, we would like the coefficient of $d_j x_{ij}$ to be in terms of only dual variables with $j \in B$. To this end, we will first notice that by Observation 4.2.7, $i \in F_2 \cap S_k$ implies

$k \in B$. Thus, the following is true for $i \in S_k$:

$$\begin{aligned}
\sum_{v:i\in F_2} v(F_1,F_2,A,B) + \sum_{\substack{v:i\in F_1\\ k\in B}} v(F_1,F_2,A,B) &= \sum_{\substack{v:i\in F_2\\ k\in B}} v(F_1,F_2,A,B) + \sum_{\substack{v:i\in F_1\\ k\in B}} v(F_1,F_2,A,B)\\
&= \sum_{\substack{v:i\notin A\\ k\in B}} v(F_1,F_2,A,B)\\
&= \sum_{\substack{v:i\notin A\\ k\in B\\ j\notin B}} v(F_1,F_2,A,B) + \sum_{\substack{v:i\notin A\\ j,k\in B}} v(F_1,F_2,A,B)\\
&\le \sum_{\substack{v:k\notin B\\ j\in B}} v(F_1,F_2,A,B) + \sum_{\substack{v:i\notin A\\ j,k\in B}} v(F_1,F_2,A,B),
\end{aligned}$$

where for the last inequality we used (4.11).

So, the original expression is at most

$$\sum_{k\in\mathcal{C}}\sum_{i\in S_k}\sum_{j\in\mathcal{C}} d_j x_{ij}\Big(\sum_{\substack{v:i\notin A,\\ j,k\in B}} v(F_1,F_2,A,B) + (2n-1)\sum_{\substack{v:j\in B,\\ k\notin B}} v(F_1,F_2,A,B)\Big).$$

Interchanging the order of summation:

$$\begin{aligned}
&\sum_{v} v(F_1,F_2,A,B)\sum_{j\in B}\Big(\sum_{k\in B}\sum_{i\in S_k\setminus A} d_j x_{ij} + (2n-1)\sum_{k\notin B}\sum_{i\in S_k} d_j x_{ij}\Big)\\
=\ &\sum_{v} v(F_1,F_2,A,B)\sum_{j\in B}\Big(\sum_{k\in B}\sum_{i\in S_k\setminus A} d_j x_{ij} + (2n-1)\sum_{k\notin B}\sum_{i\in S_k\setminus A} d_j x_{ij}\Big)\\
=\ &\sum_{v} v(F_1,F_2,A,B)\sum_{j\in B}\Big(\sum_{i\in S\setminus A} d_j x_{ij} + (2n-2)\sum_{k\notin B}\sum_{i\in S_k\setminus A} d_j x_{ij}\Big)\\
\le\ &\sum_{v} v(F_1,F_2,A,B)\sum_{j\in B}\Big(\sum_{i\in S\setminus A} d_j x_{ij} + (2n-2)\sum_{i\in S\setminus A} d_j x_{ij}\Big)\\
\le\ &\sum_{v} v(F_1,F_2,A,B)\sum_{j\in B}(2n-1)\sum_{i\in S\setminus A} d_j x_{ij}\\
\le\ &(2n-1)\sum_{v} v(F_1,F_2,A,B)d(A,B).
\end{aligned}$$

For the first inequality, we used Observation 4.2.7. ■

Together with the bound from Lemma 4.2.5, we get that the total primal cost of this cluster (let us call it cluster $\mathcal{C}^m$) is at most $2n \sum_{v \in \mathcal{V}^m} v(F_1, F_2, A, B)$, where $\mathcal{V}^m$ is the set of positive dual variables with $F_2 \cup A \cup B \subseteq \mathcal{C}^m$.

Clearly, every positive dual variable appears in the bound of exactly one cluster. We are finally ready to prove our main theorem:

Theorem 4.2.9 *The primal-dual algorithm described gives a $2n$-approximation for the capacitated facility location problem.*

Proof: Let z_P be the objective primal value of the solution obtained, and z_D the corresponding dual value. Using Lemma 4.2.5 and Proposition 4.2.8, we get

$$\begin{aligned} z_P &= \sum_m \sum_{i \in S \cap \mathcal{C}^m} \left[f_i + \sum_{j \in \mathcal{C}^m} d_j x_{ij} \right] \leq \sum_m 2n \sum_{v \in \mathcal{V}^m} v(F_1, F_2, A, B) d(A, B) \\ &= 2n \sum_v v(F_1, F_2, A, B) d(A, B) = 2n z_D. \end{aligned}$$

■

4.2.4 Tightness of the Analysis

Finally, we prove that our analysis of the algorithm is tight. Consider an instance with n facilities and n clients, such that $d_1 = u_n = D$, $d_2 = \cdots = d_n = u_1 = \cdots = u_{n-1} = 1$. All opening costs are zero, and the underlying tree is a path of 2n vertices. One of the leaves has j_1, and the other one has i_n. All other clients and facilities are located on the remaining vertices in alternating order. The edge incident to j_1 has weight $1 + \epsilon$ for some $\epsilon > 0$, all other edges have weight 1. We can see this graphically in Figure 4.1.

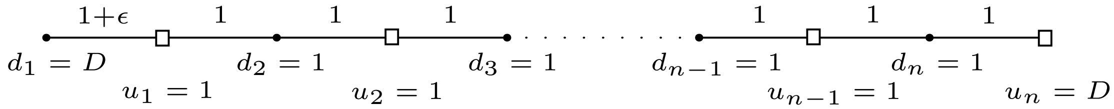

Figure 4.1: This instance proves tightness of the analysis. As before, squares represent facilities and circles represent clients.

Clearly, after time 1, facility i_m gets connected to client j_{m+1} for every $m = 1, \ldots, n-1$; and facility i_m gets connected to client j_m for every $m = 2, \ldots, n$. So, facilities $i_1, \ldots, i_n$

and clients $j_2, \ldots, j_n$ immediately form a big cluster. Since opening costs are zero, all these facilities are opened. Since we did not specify any tie-breaking rules for assigning demand within a cluster, let us assume that demand is assigned in such a way that facility i_m always serves only client j_m, for $m = 1, \ldots, n-1$. After this point, the demand of all clients in the cluster is satisfied and all facilities in the cluster, except for i_n, are using their full capacity. $\epsilon/2$ units of time later, client j_1 gets connected to the big cluster. Immediately, its demand is satisfied by i_n and the algorithm terminates.

It is easy to check that the primal value of this solution is (as $\epsilon \to 0$) $z_P = D(2n-1) + n - 1$. However, the value of the final dual solution is $z_D = D + n - 1$. If we set $D = n$, this gives us a gap of $\dfrac{z_P}{z_D} = \dfrac{2n^2-1}{2n-1} = \Theta(n)$, which proves that the analysis is asymptotically tight. Intuitively, this happens because in the last iteration of the algorithm, client j_1 is only connected to facility i_1 (which is roughly at distance 1), but it gets served by facility i_n, which is roughly at distance $2n-1$.

Several attempts were made to modify the algorithm, but we could not break the $\Theta(n)$ barrier.

Curiously, the approximation itself is not bad, since the optimal solution of the instance can be proved to be $z_{OPT} = D(2n-1) - n + 1$. So, the algorithm is only giving us a bad dual solution, but a somewhat good primal solution. However, to prove a better approximation guarantee for this algorithm, we would need a proof that does not compare the primal value only to the dual value.

Based on these observations and on personal experience, we can say that this is a topic with a lot of potential for future research.

Chapter 5

Conclusions

In this thesis, we presented two original approximation algorithms for the capacitated facility location problem.

The first one is based on the algorithm described in [12], and is based on rounding the solution to an LP relaxation strengthened by flow-cover inequalities. It gives a $2p$-approximation for the capacitated facility location problem with uniform capacities and tree-metric assignment costs, where p is the number of paths into which we decompose the tree during the running of the algorithm. Since it is trivial to find a decomposition into $\ell - 1$ paths, where ℓ is the number of leaves in the tree, this gives us an approximation guarantee of $O(\ell)$. We can also assume without loss of generality that every leaf contains a facility, so the approximation is $O(|\mathcal{F}|)$. In particular, if an upper bound on the number of facilities (or leaves) is given, this is a constant factor approximation algorithm. In this case, we also know that we can run it in polynomial time without using the ellipsoid method. If the number of leaves is not bounded, we can still run the algorithm in polynomial time, by using the ellipsoid method to solve the LP.

The second algorithm we present is a generalization of the primal-dual scheme of Carnes and Shmoys introduced in [3], and is based on an LP with only flow-cover inequalities (of a different type as the ones used before) in the restrictions. Their algorithm gives a 2-approximation for the single-client capacitated facility location problem (with arbitrary capacities), ours gives a $2n$-approximation for the general CFL problem; where n is the number of clients. Again, if the number of clients is bounded, this gives a constant factor approximation.

While it is not common to see approximation algorithms in the literature with an approximation guarantee that is linear on the size of the problem, the importance of these results is that they show that flow-cover inequalities do close the integrality gap of the natural relaxation, at least to some extent. Indeed, the standard example that shows that the natural relaxation has an arbitrarily large integrality gap uses only one client and two facilities, as we saw in the introduction. Our results tell us that to find instances that show an arbitrarily large integrality gap of the relaxation used in Chapter 3, we would need to use an arbitrarily large number of facilities on leaves of the tree; while to do the same thing for the relaxation of Chapter 4, we would have to use an arbitrarily large number of clients.

We conjecture that flow-cover inequalities (of either type) are strong enough to give an LP relaxation with a bounded integrality gap. In particular, we have not been able to find instances that prove an integrality gap greater than 2. However, as seen in Chapters 3 and 4, the techniques we used are not enough to prove a constant approximation guarantee, and this calls for new ideas.

APPENDICES

Appendix A

Complexity

A.1 Complexity of the Multi-Item Capacitated Lot-Sizing Problem

NP-hardness of the multi-item lot-sizing problem with uniform capacities can be proved by doing a reduction from the well known strongly NP-hard problem 3-PARTITION.[1] In the 3-PARTITION problem, we are given a set P of $3m$ jobs of integer sizes $p_1, \dots, p_{3m}$ such that $B/4 < p_i < B/2$ for every $i = 1, \dots, 3m$ and for some positive integer B. The problem consists in partitioning the set of jobs into m groups, $S_1, \dots, S_m$ containing 3 jobs each, such that $\sum_{i \in S_j} p_i = B$ for every $j = 1, \dots, m$.

Theorem A.1.1 *The capacitated multi-item lot-sizing problem with uniform capacities is strongly NP-hard.*

Proof: Given an instance of 3-PARTITION, let us construct the following instance of capacitated multi-item lot-sizing.

Every job i corresponds to an item i in the lot-sizing problem. Let $\mathcal{S} = \{P_s \subseteq P : |P_s| = 3 \text{ and } \sum_{i \in P_s} p_i = B\}$. We will consider a time horizon of $T = |\mathcal{S}| + 1$ periods. The last period has a demand of 1 for every item (that is, $d_{i(T+1)} = 1$ for every i), all other demands are zero. Furthermore, we assume that every order has a capacity $u = 3$. Next, we need to define the holding costs. By the way we defined T, we can assume that every time period s corresponds to a set $P_s \in \mathcal{S}$, except for the last one. So, for every

[1] A proof of this result can be found in the classical book by Garey and Johnson, [6]

i, s, t we define the holding costs $c^i_{st} = \begin{cases} 2(t-s) & \text{if } i \in P_s \\ 2(t-s)+1 & \text{otherwise} \end{cases}$. These holding costs are clearly monotonous and non-negative. Intuitively, we can interpret the additive factor 1 as a "penalty" that an item has to pay if it is not being ordered at a time corresponding to a set P_s containing it (or, alternatively, we can think that if it is ordered at time s with $i \in P_s$, the item gets a "discount"). We define the opening costs as $f_s = 6s$ for $s \in \{1, \ldots, T-1\}$, and $f_T = M$, for some very large number M. By doing this, we prevent order T from being opened in an optimal solution.

Any solution to the 3-PARTITION problem induces a natural solution to this lot-sizing problem instance. That is, for every set P_s in the solution (let us say this solution is $P_{s_1}, P_{s_2}, \ldots, P_{s_m}$), we open the corresponding order, serve the 3 corresponding items and hold them until time T. Clearly, this satisfies all demands and does not violate the capacity constraints. The value of this solution is

$$\begin{aligned} \sum_{j=1}^{m} \left(f_{s_j} + \sum_{i \in P_{s_j}} c^i_{s_j T} \right) &= \sum_{j=1}^{m} (6s_j + 3 \cdot 2(T - s_j)) \\ &= \sum_{j=1}^{m} (6s_j + 3 \cdot 2(T - s_j)) = \sum_{j=1}^{m} 6T = 6mT. \end{aligned}$$

These last lines should clarify the reason for choosing these seemingly arbitrary ordering and holding costs; we choose them as necessary to make the quantity above not depend on the partition of jobs (as long as it is feasible).

Next, we will prove that $6mT$ is a lower bound on the value of all feasible solutions for the lot-sizing problem, and that the only solutions that attain this value correspond to feasible partitions.

Let us consider a feasible solution to the lot-sizing problem. First of all, it is not hard to see that if this solution opens $k > m$ orders, we can always close the first $k - m$ orders and reassign to the open slots in the remaining m orders the demand they were serving; all of this without increasing the cost. This is so because of the way we defined the holding costs (even if an item is getting the "discount", this discount is of size 1, and by delaying the time at which it is ordered, we save at least 2), and because there are exactly $3m$ items and each order has a capacity of 3.

So, let us assume without loss of generality that the solution we are considering opens exactly m orders.

Following a procedure analogous to the one above, we get that the value of such a solution is $6mT + n$, where n is the number of "misplaced" items. That is, items that we order at a time that does not correspond to a set P_s containing this item. By the definition of holding costs above, each of these items pays a penalty of 1.

From this discussion, it is clear that the objective value is $6mT$ if there are no misplaced items, and $6mT + n > 6mT$ if there are $n > 0$ misplaced items. And clearly, every solution with no misplaced items corresponds to a valid partition; so we are done. Note that solutions having misplaced items can still correspond to a valid partition, but this does not matter for our purposes.

■

A.2 Complexity of the Multi-Commodity Capacitated Facility Location Problem with Monotonous Costs on Trees

To finish this section, we prove NP-hardness for MCCFL, as defined in Section 3.9. In fact, the problem is strongly NP-hard even when the underlying structure is a path instead of a tree.

Theorem A.2.1 *MCCFL is strongly NP-hard.*

Proof: Again, we will reduce the 3-PARTITION problem to MCCFL, and the construction is almost identical to the previous one. There are only some details we need to take care of.

Define $\mathcal{S}$ and T exactly as before. Consider a path of $T + 1$ vertices, with a client on one of the extremes and facilities in all remaining vertices. Suppose all edges have a length of one. The client has a demand of one unit of every item, and all facilities have capacity 3. The opening costs are also the same as before. For every item k, facility i (and for the unique client j), we define the assignment cost c_{ij}^k as twice the length of the i, j-path if $k \in P_i$, and that quantity plus one otherwise. The rest of the proof follows just like that of Theorem A.1.1. ■

Note that the proof for lot-sizing could so easily be adapted to MCCFL because the only non-trivial demand point in the constructed instance is the last time period.

Actually, we could have done a more direct reduction. It is not hard to see that MCCFL encodes capacitated multi-item lot-sizing, since our definition of monotonous costs allows us to keep the "left-to-right" assignment costs as in the original lot-sizing instance, and make all "right-to-left" costs be prohibitively high.

Appendix B

Separation of Flow-Cover Inequalities

Here, we prove that certain subsets of flow-cover inequalities can be separated in polynomial time. This proof (with some small differences) appears in [12], and is based on a proof of an analogue statement for a slightly different type of inequalities by Aardal (See [1]).

We will denote the set of facilities as $\boldsymbol{F}$ to emphasize that it is fixed.

Proof of Theorem 2.2.3:
Recalling the definition of r_A and $D(A)$, the flow-cover inequality (2.3) can be rewritten as

$$\sum_{(i,t)\in A} \frac{d_{it}}{u}\left(1-\sum_{s\in \boldsymbol{F}} x^i_{st}\right) \geq r_A\left(\ell_A - \sum_{s\in\boldsymbol{F}} y_s\right). \tag{B.1}$$

The separation problem for a solution $(\widehat{x},\widehat{y})$ of the LP is

$$\begin{aligned}
\min \quad & \sum_{i=1}^{N}\sum_{t=1}^{T}\frac{d_{it}}{u}\left(1-\sum_{s\in\boldsymbol{F}}\widehat{x}^i_{st}\right) z^i_t - r_A\left(\ell_A - \sum_{s\in\boldsymbol{F}}\widehat{y}_s\right) \\
\text{s.t.} \quad & \sum_{i=1}^{N}\sum_{t=1}^{T}\frac{d_{it}}{u} z^i_t = (\ell_A - 1) + r_A, \\
& z^i_t \in \{0,1\} \qquad \text{for } i=1,\dots,N; t=1,\dots,T, \\
& r_A \in (0,1], \\
& \ell_A \in \mathbb{Z}_+.
\end{aligned} \tag{B.2}$$

That is, an inequality of type (B.1) will be violated if and only if the optimal value of the separation problem is negative. This is not hard to see: If there is a solution (z, r_A, ℓ_A) to the problem above that yields a negative value, let $A = \{(i,t) | z_t^i = 1\}$. Constraint (B.2) ensures that the numbers r_A and ℓ_A are consistent with this choice of A, and the fact that the objective value is negative says that (B.1) is violated. Similarly, if for some A we know that (B.1) is violated, we can define z, r_A and ℓ_A accordingly and that will give us a negative value.

The objective function in the separation MIP is non-linear. Our next goal is to find a simpler equivalent formulation.

To simplify the notation, we will abbreviate $\sum\limits_{s \in \boldsymbol{F}} \widehat{x}_{st}^i$ as X_t^i and $\sum_{s \in \boldsymbol{F}} \widehat{y}_s$ as Y.

At this point, we can make several observations.

Observation B.1.2 *If* (B.1) *is violated, then* $r_A < 1$.

Proof: Suppose the separation MIP returns a solution (z, r_A, ℓ_A) with $r_A = 1$. Then (after defining A as the support of z, as explained before), equation (B.2) becomes $\sum\limits_{(i,t) \in A} \dfrac{d_{it}}{u} = \ell_A$. Capacity constraint (2.2) implies

$$\sum_{(i,t) \in A} \frac{d_{it}}{u}(-X_t^i) \geq \sum_{i=1}^{N} \sum_{t=1}^{T} \frac{d_{it}}{u}(-X_t^i) \geq -Y.$$

Adding both inequalities, we obtain

$$\sum_{i=1}^{N} \sum_{t=1}^{T} \frac{d_{it}}{u}(1 - X_t^i) \geq \ell_A - Y,$$

which is precisely inequality (B.1) (recall $r_A = 1$).

So, the flow-cover inequality given by any such solution will be satisfied. ■

Thus, we can safely substitute the second-to-last constraint in the separation MIP (that is, $r_A \in (0,1]$) by $r_A \in (0,1)$.

Using this and constraint (B.2), we obtain

$$\ell_A - 1 = \lfloor \ell_A - 1 + r_A \rfloor < \sum_{i=1}^{N} \sum_{t=1}^{T} \frac{d_{it}}{u} z_t^i < \lceil \ell_A - 1 + r_A \rceil = \ell_A.$$

Note that these two inequalities are equivalent to (B.2). Indeed, r_A can be trivially obtained from any solution satisfying them.

Furthermore, by Lemma 2.2.2 we have $\lfloor Y \rfloor = \ell_A - 1$ and $\lceil Y \rceil = \ell_A$.

All of this allows us to substitute r_A and ℓ_A away from the constraints. We will next do the same thing for the objective function.

This is easily accomplished by using $\ell_A = \lceil Y \rceil$ and $r_A = \sum_{i=1}^{N} \sum_{t=1}^{T} \frac{d_{it}}{u} z_t^i - (\ell_A - 1) = \sum_{i=1}^{N} \sum_{t=1}^{T} \frac{d_{it}}{u} z_t^i - \lfloor Y \rfloor$. Here, we used (B.2) and $\lceil Y \rceil - \lfloor Y \rfloor = 1$.

Using all of this, the new separation problem is the following IP:

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{N} \sum_{t=1}^{T} \frac{d_{it}}{u} z_t^i \left(-X_t^i - \lfloor Y \rfloor + Y \right) + \left(\lceil Y \rceil - Y \right) \lfloor Y \rfloor \\
\text{s.t.} \quad & \sum_{i=1}^{N} \sum_{t=1}^{T} \frac{d_{it}}{u} z_t^i < \lceil Y \rceil , \\
& \sum_{i=1}^{N} \sum_{t=1}^{T} \frac{d_{it}}{u} z_t^i > \lfloor Y \rfloor , \\
& z_t^i \in \{0, 1\} \qquad\qquad \text{for } i = 1, \dots, N; t = 1, \dots, T.
\end{aligned}
$$

Now, we can note that the second additive term in the objective function is a non-negative constant, so we need to care mostly about the first term.

Surprisingly, this IP can be solved by the following really simple greedy algorithm:

(i) Let $\tilde{A} = \{(i, t) : -X_t^i - \lfloor Y \rfloor + Y < 0; i = 1, \dots, N; t = 1, \dots, T\}$.

(ii) Let $z_t^i = 1$ if $(i, t) \in \tilde{A}$, and 0 otherwise.

(iii) If z is feasible in the MIP above and its objective value is negative, then $\boldsymbol{F}$ and $\tilde{A}$ define a violated inequality of the kind (B.1). Otherwise, no such violated inequality exists.

In other words, if we choose only those demand points whose inclusion in $\tilde{A}$ makes the objective value smaller; then the resulting solution is feasible.

It is clear that this algorithm runs in time $O(NT)$. We only need to prove the feasibility of the solution found by it.

Clearly, the objective value corresponding to $\tilde{A}$ (that is, the value given by z, the characteristic vector of $\tilde{A}$) is a lower bound on the value of every feasible solution to the separation IP, no matter if z is feasible or not. If $\tilde{A}$ is feasible and it has a negative value, then there is clearly a violated inequality. So, to prove the correctness of the algorithm (and thus, to complete the proof of the theorem), all we need to do is prove that if one of the restrictions of the IP is not satisfied, then the corresponding objective value is non-negative.

Suppose $\displaystyle\sum_{(i,t)\in\tilde{A}} \frac{d_{it}}{u} \geq \lceil Y \rceil$. That is, the first constraint of the IP is violated.

The corresponding objective value is

$$\begin{aligned}
&\sum_{(i,t)\in\tilde{A}} \frac{d_{it}}{u}(-X_t^i - \lfloor Y \rfloor + Y) + (\lceil Y \rceil - Y)\lfloor Y \rfloor \\
&\geq -\sum_{(i,t)\in\tilde{A}} \frac{d_{it}}{u} X_t^i + \lceil Y \rceil(-\lfloor Y \rfloor + Y) + (\lceil Y \rceil - Y)\lfloor Y \rfloor \\
&\geq -\sum_{(i,t)\in\tilde{A}} \frac{d_{it}}{u} X_t^i + Y(\lceil Y \rceil - \lfloor Y \rfloor) \\
&= -\sum_{(i,t)\in\tilde{A}} \frac{d_{it}}{u} X_t^i + Y \\
&\geq 0.
\end{aligned}$$

For the last inequality, we used capacity constraints (2.2).

Similarly, if $\sum_{(i,t)\in\tilde{A}} \frac{d_{it}}{u} \le \lfloor Y \rfloor$, the objective value is

$$\begin{aligned}
&\sum_{(i,t)\in\tilde{A}} \frac{d_{it}}{u}(-X_t^i - \lfloor Y \rfloor + Y) + (\lceil Y \rceil - Y)\lfloor Y \rfloor \\
&\quad \ge \sum_{(i,t)\in\tilde{A}} \frac{d_{it}}{u}(-X_t^i - \lfloor Y \rfloor + Y) + (\lceil Y \rceil - Y) \sum_{(i,t)\in\tilde{A}} \frac{d_{it}}{u} \\
&\quad = \sum_{(i,t)\in\tilde{A}} \frac{d_{it}}{u}(-X_t^i - \lfloor Y \rfloor + \lfloor Y \rfloor) \\
&\quad = \sum_{(i,t)\in\tilde{A}} \frac{d_{it}}{u}(1 - X_t^i) \\
&\quad \ge 0.
\end{aligned}$$

The last inequality follows by constraint (2.1). ■

References

[1] Karen Aardal. Capacitated facility location: Separation algorithms and computational experience. *Mathematical Programming*, 81:149–175, 1998. 10.1007/BF01581103.

[2] Ankit Aggarwal, L. Anand, Manisha Bansal, Naveen Garg, Neelima Gupta, Shubham Gupta, and Surabhi Jain. A 3-approximation for facility location with uniform capacities. In Friedrich Eisenbrand and F. Shepherd, editors, *Integer Programming and Combinatorial Optimization*, volume 6080 of *Lecture Notes in Computer Science*, pages 149–162. Springer Berlin / Heidelberg, 2010.

[3] Tim Carnes and David B. Shmoys. primal-dual schema for capacitated covering problems. In *IPCO*, pages 288–302, 2008.

[4] Fabián Chudak and David Williamson. Improved approximation algorithms for capacitated facility location problems. In *Integer Programming and Combinatorial Optimization*, volume 1610 of *Lecture Notes in Computer Science*, pages 99–113. Springer Berlin / Heidelberg, 1999.

[5] Michael Florian, Jan K. Lenstra, and Alexander H. G. Rinnooy Kan. Deterministic production planning: Algorithms and complexity. *Management Science*, 26(7):pp. 669–679, 1980.

[6] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, NY, USA, 1979.

[7] Martin Grötschel, Lászlo Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics.* Springer, 1988.

[8] Sudipto Guha and Samir Khuller. Greedy strikes back: improved facility location algorithms. In *Proceedings of the ninth annual ACM-SIAM symposium on Discrete*

algorithms, SODA '98, pages 649–657, Philadelphia, PA, USA, 1998. Society for Industrial and Applied Mathematics.

[9] Stan P. M. Van Hoesel and Albert P. M. Wagelmans. Fully polynomial approximation schemes for single-item capacitated economic lot-sizing problems. *Math. Oper. Res.*, 26:339–357, May 2001.

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

[11] Madhukar R. Korupolu, C. Greg Plaxton, and Rajmohan Rajaraman. Analysis of a local search heuristic for facility location problems. In *Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*, SODA '98, pages 1–10, Philadelphia, PA, USA, 1998. Society for Industrial and Applied Mathematics.

[12] Retsef Levi, Andrea Lodi, and Maxim Sviridenko. Approximation algorithms for the capacitated multi-item lot-sizing problem via flow-cover inequalities. *Math. Oper. Res.*, 33(2):461–474, 2008.

[13] Retsef Levi, David Shmoys, and Chaitanya Swamy. Lp-based approximation algorithms for capacitated facility location. In Daniel Bienstock and George Nemhauser, editors, *Integer Programming and Combinatorial Optimization*, volume 3064 of *Lecture Notes in Computer Science*, pages 21–27. Springer Berlin / Heidelberg, 2004.

[14] Mohammad Mahdian and Martin Pál. Universal facility location. In Giuseppe Di Battista and Uri Zwick, editors, *Algorithms - ESA 2003*, volume 2832 of *Lecture Notes in Computer Science*, pages 409–421. Springer Berlin / Heidelberg, 2003.

[15] Martin Pál, Éva Tardos, and Tom Wexler. Facility location with nonuniform hard capacities. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 329–338, oct. 2001.

[16] Ramamoorthi Ravi and Amitabh Sinha. Approximation algorithms for multicommodity facility location problems. *SIAM J. Discrete Math.*, 24(2):538–551, 2010.

[17] David B. Shmoys, Chaitanya Swamy, and Retsef Levi. Facility location with service installation costs. In *SODA*, pages 1088–1097, 2004.

[18] Zoya Svitkina and Éva Tardos. Facility location with hierarchical facility costs. *ACM Transactions on Algorithms*, 6(2), 2010.

[19] Vijay V. Vazirani. *Approximation algorithms*. Springer, 2001.

[20] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.

[21] Jiawei Zhang, Bo Chen, and Yinyu Ye. A multiexchange local search algorithm for the capacitated facility location problem. *Mathematics of Operations Research*, 30(2):pp. 389–403, 2005.