Approximation Algorithms for Facility Location Problems

Adriana Bumb

2002

Ph.D. thesis
University of Twente

Twente University **Press**

Also available in print:
http://www.tup.utwente.nl/tupress/catalogue/book/index.jsp?isbn=9036517877

Approximation Algorithms For Facility Location Problems

Twente University **Press**

Publisher:

Twente University Press, P.O. Box 217, 7500 AE Enschede, The Netherlands,

www.tup.utwente.nl

Print: Océ Facility Services, Enschede

ISBN 9036517877

APPROXIMATION ALGORITHMS FOR FACILITY LOCATION PROBLEMS

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
prof.dr. F.A. van Vught,
volgens besluit van het College voor Promoties
in het openbaar te verdedigen
op vrijdag 4 oktober 2002 te 13.15 uur

door

Adriana Felicia Bumb
geboren op 9 juni 1974
te Cluj-Napoca, Roemenië

Dit proefschrift is goedgekeurd door de promotoren

prof.dr. U. Faigle

en

prof.dr. G. Woeginger

Acknowledgements

Five years ago, when I came to The Netherlands, I could hardly imagine that I will ever have a Ph.D. thesis. I want to thank all the people who helped me to achieve this goal.

I thank prof. Ulrich Faigle for encouraging me to do research and most of all, for offering me a Ph.D. position at the University of Twente. I am also thankful to prof. Gerhard Woeginger, for being helpful and kind.

A special word of thanks goes to dr. Walter Kern, my daily supervisor, for the nice discussions we had on several topics of this thesis, for his willingness to listen any time to my ideas and for the freedom he gave me to work on the problems I liked. His comments on an early version of this thesis helped me to considerably improve the presentation.

I want to express my gratitude to prof. M. Labbé, prof. J. van Leeuwen, prof. H. Zijm and prof. C. Hoede for participating in my graduation committee.

I thank all the members of the DOS department for their very pleasant company during the years. I was very lucky to share the same office with Paul Bonsma, whose calm proved to be a good antidote to my agitation during the last period. I particularly enjoyed his sharp sense of humor and the capacity he has to give a funny turn to many situations. It is very difficult to thank properly Kees Hoede for everything he has done for me since I have begun to work at the University of Twente. With his enthusiasm and good advises, he has helped me keeping my optimism on a high level all these years. I also want to thank Jan-Kees van Ommeren for the nice conversations we had and for his help with Latex. I am grateful to all three of you, for never loosing the patience of correcting my Dutch.

For me, The Netherlands is not only the place where I have studied for the last five years, but also the place where I made many good friends. Orest, Israel, Zenith, Alex, Iustina, Bibi, Florin, Sacha, Corina, Petrica and all the others, thanks for the nice time we had together. Especially, I want to thank Marisela for her wonderful friendship.

Last, but not least, I want to thank my parents, my sister, Peti and Andrei for their constant support, without which my stay abroad would have been much more difficult.

Writing an acknowledgement section is a very difficult task, since it is impossible to mention all the people you would like to thank. Hoping for understanding, I want to thank really all those who made the years I worked on this thesis so enjoyable.

Contents

Chapter 1

Introduction

1.1 Discrete facility location problems

Suppose that a media company plans to place newspaper stands in a city. The company has already identified potential stand sites in a number of different neighborhoods and knows the cost of placing and maintaining a stand at each potential site. Further assume that the demand for newspapers in each neighborhood of the city is known. If the company wants to open any number of stands, where should they be located in order to minimize the sum of the total placing and maintaining cost and the average traveling distance of the customers?

The preceding question is an example of a *facility location problem* . Location problems have occupied a central place in Operations Research since the early 1960's. They model design situations such as deciding placements of factories, warehouses, fire stations or hospitals (see the work of Stollsteimer [St63], Kuehn and Hamburger [KH63], Manne [Ma64]) and clustering analysis (see Mulvey and Crowder [MC79]). As modern day applications, we mention the placement of data objects in capacity caches in a network in order to optimize latency of access (see, e.g., the work of Baev and Rajaraman [BR01], Meyerson *et al.* [MMP01]).

Basically, a location problem is characterized by four elements:

• A set of *locations* where facilities may be built/opened. For every location some information about the *cost of building* or *opening* a facility at that location is given.

• A set of *demand points (clients)* that have to be assigned for service to some facilities. For every client one receives some information regarding its demand and about the costs/profits incurred if he would be served by a certain facility.

• A list of *requirements* to be met by the open facilities and by any assigment of demand points to facilities.

• A *function* that associates to each set of facilities the cost/profit incurred if one would open all the facilities in the set and would assign the demand points to them such that the requirements are satisfied.

The goal of the problem is then to find the set of facilities to be opened in order to optimize the given function.

There are a variety of types of facility location problems corresponding to the features of the four elements mentioned before. Some basic classes of facility location problems are listed below.

If the sets of demand points and facility locations are finite one speaks about a *discrete facility location problem*, otherwise about a *continuous facility location* problem. If all the data are exact, models are referred to as *deterministic*. If some parameter values are given by probability distributions, the model is considered to be *stochastic*.

We can furthermore classify a model as *capacitated* as opposed to *uncapacitated* where the former term refers to upper bounds on the number of clients a facility can serve.

Models are called *dynamic* (as opposed to *static*) if the time element is explicitely represented. Such models may be used in decision problems where not only the location of facilities, but also "the moment in time" to establish them is important.

An extensive survey of location problems, their applications and methods used to handle them can be found in the book edited by Mirchandani and Francis [MF90].

The problems to which we restrict our attention in this thesis can be characterised as discrete, deterministic and static and with one exception, they are also uncapacitated. Although very simple in formulation, these problems are very difficult to solve, except for some special cases.

More precisely, we will discuss algorithmic aspects of different variants of the easiest model in facility location theory, the *uncapacitated facility location problem* ($UFLP$).

In the *uncapacitated facility location problem*, a set of locations F and a set of demand points D are given. The cost of opening a facility at location $i \in F$ is f_i. Every demand point $j \in D$ has a *demand* d_j and the cost of transporting one unit of demand from facility i to demand point j, the so called *service cost*, is c_{ij}. Every demand point has to be served by one facility. There are no "capacity"-restrictions on the demand a facility may serve. The goal is to decide which facilities to open and to assign demand points to facilities such that the *total cost*, i.e, the opening cost and the service cost is minimized.

Clearly, the example at the beginning of this section can be modeled as an $UFLP$. There the locations are the sites where newspaperstands may be placed, the facilities are the newspaper stands and the demand points are the neighborhoods. For every location i, the cost f_i is the sum of the cost of opening a newspaperstand at location i and of maintaining it. The average travelling time for the clients in the neighborhood j to location i can be modeled as the service cost c_{ij}.

We seek efficient algorithms for specific location problems. The question that arises is when do we consider an algorithm efficient? Is there a relation between the difficulty of a problem and the efficiency of the algorithms we are looking for? This type of questions can be answered if one analyzes the *computational complexity* of the encountered problem.

1.2 Complexity and approximability

All the problems treated in this thesis have in common that they require to find the "best" solution from a finite set. In other words, they are all *optimization problems.* Formally, an *optimization problem* Ψ is characterized by four components:

• A set of input *instances* $\mathcal{I}_\Psi$. We will assume that all numbers specified in an input are integers (rationals can be transformed to integers).

• The set of all *feasible solutions* $S_\Psi(x)$ for an instance $x \in \mathcal{I}_\Psi$.

• A function f_Ψ, defined for pairs (x, y) such that $x \in \mathcal{I}_\Psi$ and $y \in S_\Psi(x)$. For every such pair (x, y), f_Ψ assigns a positive integer which is the value of the feasible solution y. f_Ψ is called the *objective function.*

• The *goal* of the problem, namely whether it is a *maximization* or a *minimization* problem.

Example *The $UFLP$*

Instance: The number of facility locations $n_f = |F|$, the number of demand points $n_d = |D|$, the costs of opening facilities f_i, $1 \leq i \leq n_f$, the demands d_j, $1 \leq j \leq n_d$ and the transportation costs c_{ij}, $1 \leq i \leq n_f$, $1 \leq j \leq n_d$.

Feasible solutions: A collection of subsets $S \subseteq F$ and an assignment $\sigma_S : D \to S$ of demand points to facilities in S.

Objective function: For every (S, σ_S), $f(S, \sigma_S) = \sum_{j \in D} d_j c_{\sigma_S(j)j} + \sum_{i \in S} f_i$. f represents the total cost incurred by opening facilities in S and serving every demand point j from $\sigma_S(j)$.

Goal: Minimize

In general, a minimization problem Ψ can be formulated as follows (a maximization problem can be formulated in a similar way):

Given an instance $x \in \mathcal{I}_\Psi$, find a solution $s_{opt} \in S(x)$ such that for every $s \in S(x)$, $f_\Psi(s_{opt}) \leq f_\Psi(s)$.

Such an s_{opt} is called an *optimal solution* for x. We denote the value of f_Ψ in s_{opt} by $f_\Psi^*(x)$.

With every optimization problem we can associate a *decision problem*, i.e., a problem the answer of which is either 'yes' or 'no'. This can be done by giving a bound on the optimal solution. Thus, with a minimization problem Ψ we can associate the following decision problem

Given an instance $x \in \mathcal{I}_\Psi$ and a rational number f_Ψ^*, is there a solution $s \in S_\Psi(x)$ such that $f_\Psi(s) \leq f_\Psi^*$?

More formally, a decision problem Π is a problem for which the set of all instances $\mathcal{I}_\Pi$ is partitioned into a set $\mathcal{Y}_\Pi$ of *positive instances* and a set $\mathcal{N}_\Pi$ of *negative instances* and the problem asks, for any instance $x \in \mathcal{I}_\Pi$, to verify whether $x \in \mathcal{Y}_\Pi$.

The *size* of an instance x, denoted by *size*(x), is the number of bits necessary to encode x. If encoded in binary, the size of an integer a is $\log a + 1$ (one bit is encoding the sign). We suppose that all the algorithms run on the same machine, the Turing machine, but we could also assume other polynomially related models of computation such as the random access machine.

The most widely accepted performance measure for an algorithm is the time it uses for producing the final answer. In this thesis we measure the *computation time* of an algorithm by its number of performed elementary operations (additions, multiplications, comparisons,etc.). The *running time* $t(\xi)$ of an algorithm is defined as the maximum computation time of the algorithm on a problem instance of size ξ. The running time is often expressed in terms of $O(g(\xi))$, where $g : N \to R_+$ is a function (for two functions $f : N \to R_+$ and $g : N \to R_+$ we write $f(n) = O(g(n))$ if there exists positive numbers $c, n_0 \in N$ such that $f(n) \leq cg(n)$ for $n \geq n_0$).

A *polynomial time algorithm* is an algorithm that for any input x has a running time of $O\left(size(x)^k\right)$, for some $k \in N$.

Example *TheUFLP with no opening costs*
Consider a special case of the $UFLP$ in which the cost of opening a facility at each location i is zero. Clearly, the optimal solution opens all the facilities and assigns each demand point to the closest one. The closest facility to each demand point j can be found in $O(n_f)$. Hence, the solution can be found in time $O(n_d n_f)$. If encoded in binary, the length of an instance is $O(n_f n_d \log \max\{c_{ij}, d_j, n_f, n_d\})$ and therefore we can conclude that the algorithm is polynomial.

The problems that can be solved in polynomial time are in general considered *easy* problems.

In the next chapters we will concentrate on *hard* problems, i.e., problems for which no polynomial time algorithm is known and is unlikely to exist. The notion of hardness was formalized in the early seventies. The foundations have been laid in the work of Cook [Co71]. His work is based on decision problems. Since with every optimization problem we can associate a decision problem, this is not restrictive at all.

A decision problem Π is *solved* by an algorithm $\mathcal{A}$ if the algorithm halts for every instance $x \in \mathcal{I}_\Pi$ and returns 'yes' if and only if $x \in \mathcal{Y}_\Pi$.

The class of decision problems solvable by a polynomial time algorithm is denoted by $\mathcal{P}$.

Cook's attention focuses on the class $\mathcal{NP}$ of decision problems. These are problems Π for which there exists a polynomial p and an algorithm $\mathcal{A}$ (the *certificate-checking algorithm)* such that the following is true: $x \in \mathcal{Y}_\Pi$ if and only if there exists a *certificate* $c(x)$ with $size(c(x)) \leq p(size(x))$ with the property that $\mathcal{A}$, supplied with x and $c(x)$ as input, reaches the answer 'yes' in time $p(size(x))$.

Example *The decision version of the UFLP is in $\mathcal{NP}$.*
The decision version of the UFLP is: Given an instance $x \in \mathcal{I}$ and a rational number f^*,

is there a set of facilities $S \subseteq F$ and an assignment σ_S of demand points to facilities in S of total cost at most f^*? A certificate is a list with open facilities and an assignment of the demand points to them. Clearly, the size of the certificate is bounded by the size of the instance. The algorithm $\mathcal{A}$ simply calculates the total cost, i.e., the cost of opening facilities and the service cost.

Obviously $\mathcal{P} \subseteq \mathcal{NP}$ holds. It is widely assumed that $\mathcal{P}$=$\mathcal{NP}$ is very unlikely. Perhaps the most compelling reason why scientists believe that $\mathcal{P} \neq \mathcal{NP}$ is the existence of the class of $\mathcal{NP}-$*complete problems.* These problems have the following property: If one can prove that a $\mathcal{NP}$-complete problem is in $\mathcal{P}$, then $\mathcal{P} = \mathcal{NP}$. The technique used here is that of polynomially transforming a problem into another.

A decision problem Π_1 is *polynomially transformable* to a decision problem Π_2 if there exists an algorithm that, for every instance x_1 of Π_1, produces in polynomial time exactly one instance x_2 of Π_2 such that the following holds: $x_1 \in \mathcal{Y}_{\Pi_1}$ if and only if $x_2 \in \mathcal{Y}_{\Pi_2}$.(another term used in literature for polynomially transformable is *polynomially Karp reducible*).

This means that a polynomial time algorithm $\mathcal{A}$ for Π_2 can also be used to solve an instance of Π_1 efficiently: first polynomially transform Π_1 into Π_2 and then use $\mathcal{A}$.

We can now define an $\mathcal{NP}-$complete problem.

A decision problem Π is $\mathcal{NP}-$*complete* if Π is in $\mathcal{NP}$ and all other decision problems in $\mathcal{NP}$ can be polynomially transformed to Π.

Note that polynomial transformability is a transitive relation. If Π_1 is polynomially transformable to Π_2 and Π_2 is polynomially transformable to Π_3, then Π_1 is polynomially transformable to Π_3. Hence, $\mathcal{P}$=$\mathcal{NP}$ must hold if one can prove for any $\mathcal{NP}-$complete problem that it is in $\mathcal{P}$. On the other hand, if we want to prove that a decision problem Π is $\mathcal{NP}-$complete, we only have to show that Π is in $\mathcal{NP}$ and that some decision problem already known to be $\mathcal{NP}$-complete can be polynomially transformed to Π.

The first problem proven to be $\mathcal{NP}-$complete is the satisfiability problem.(Cook [Co71]). Other well known $\mathcal{NP}-$complete problems are vertex cov, hamiltonian circuit, the maximum clique problem.

Example *The decision version of the UFLP is $\mathcal{NP}-$complete*

We have already proved that the $UFLP$ is in $\mathcal{NP}$. It only remains to prove that a known

$\mathcal{NP}-$complete problem can be polynomially transformed to the $UFLP$. For this, we will choose the vertex cover problem: Given a graph G and an integer k, does there exist a subset of k nodes of G that cover all the edges of G? (Node v is said to cover edge e if v is an endnode of e).
Now we polynomially transform the vertex cover problem to the $UFLP$. Consider a graph $G = (V, E)$ and an integer k. Construct an instance of the decision version of the $UFLP$ with the set of potential facilities $F = V$, set of demand points $D = E$ and upper bound $f^* = k + |E|$. Let $c_{ij} = 1$ if v_i is an endnode of e_j and let $c_{ij} = 2$ otherwise. Let $f_i = 1$ for all $v_i \in V$. The transformation is polynomial in the size of the graph. Clearly, the answer is 'yes' for the vertex cover problem if and only if it is 'yes' for the corresponding instance of the $UFLP$.

A more general technique that is used to prove that a problem can be solved in polynomial time, in case another problem can, is *polynomial reduction* (known also as *polynomial Turing reduction*): Let Π_1, Π_2 be two problems, not necessarily decision problems. A *polynomial reduction* from Π_1 to Π_2 is an algorithm $\mathcal{A}_1$ for Π_1 that uses an algorithm $\mathcal{A}_2$ for Π_2 as a subroutine and that would be a polynomial time algorithm for Π_1 if $\mathcal{A}_2$ were a polynomial time algorithm for Π_2. Hence, a polynomial transformation is a polynomial reduction in which the subroutine $\mathcal{A}_2$ is used only once.

Now we return back to optimization problems. One can define for optimization problems similar classes to $\mathcal{P}$ and $\mathcal{NP}$. These classes are $\mathcal{PO}$ and $\mathcal{NPO}$.

As for the decision problems, we are interested in classifying the "hard" optimization problems.

A problem Π is $\mathcal{NP}-$*hard* if there exists an $\mathcal{NP}-$complete decision problem that can be polynomially reduced to Π.

Note that this definition includes all optimization problems for which the associated problem is $\mathcal{NP}-$complete. In particular, the $UFLP$ is $\mathcal{NP}-$hard. Moreover, we have that if $\mathcal{P} \neq \mathcal{NP}$ then $\mathcal{PO} \neq \mathcal{NPO}$.

If we know that a problem is $\mathcal{NP}-$hard we already know that it cannot be solved exactly by a polynomial time algorithm unless $\mathcal{P} = \mathcal{NP}$. Hence, we have to be satisfied with an algorithm that does not return the optimal solution, but an approximate one. There are different ways to measure the quality of the solution obtained. We focus on *worst case analysis*, that is, the guarantees that we obtain must hold for any possible input. This is in contrast with *probabilistic analysis,* where a probabilistic distribution over the input is assumed and the quality measure depends on the average case behaviour of the algorithm.

An algorithm $\mathcal{A}$ is called a *$\rho-$approximation algorithm* for a minimization (maximization) problem Π, if for every instance x of Π it delivers a solution $s \in S(x)$ of value at most (at least) $\rho f(s_{opt})$, where $f(s_{opt})$ denotes the value of an optimal solution for x.

Clearly, $\rho \geq 1$ for a minimization problem and $\rho \leq 1$ for a maximization one. The value ρ can be viewed upon as a quality measure of an algorithm and is referred to as the *performance (approximation) guarantee* or the *worst case ratio* of the algorithm. The closer ρ comes to 1, the better the algorithm. The $\rho-$approximation algorithms with $\rho \in R$ are called *constant approximation algorithms.* Note that this measure is *relative* to the optimal value; the quality of the solution can be measured also with respect to the *absolute error.*

In this thesis, we are only interested in constant approximation algorithms that run in polynomial time. The class of $\mathcal{NPO}$ problems for which a constant approximation algorithm that runs in polynomial time exists is called the *class of $APX-$problems.* Among many others, the following problems belong to the class APX: maximum satisfiability, maximum knapsack, maximum cut, minimum binpacking.

We will see in Chapter 2 that the metric $UFLP$ (the $UFLP$ in which the service costs are induced by a metric) is in APX. In fact in Chapter 2 we will describe several polynomial time constant approximation algorithms for the metric $UFLP$.

From the definition of APX follows that $APX \subseteq \mathcal{NPO}$. Sahni and Gonzalez [SG76] proved that if the traveling salesman problem is in APX, then $\mathcal{P} = \mathcal{NP}$. Hence, $APX \subset \mathcal{NPO}$, unless $\mathcal{P} = \mathcal{NP}$.

Sometimes an approximable problem may even allow a *polynomial time approximation scheme,* i.e., an algorithm that for any given $\varepsilon > 0$ and any instance $x \in \mathcal{I}$ outputs in time polynomial in $size(x)$ a solution with value at most $(1+\varepsilon) f(s_{opt})$ (at least $(1-\varepsilon) f(s_{opt})$ in case of a maximization problem). Note that the running time could be exponential (or worse) in $\frac{1}{\varepsilon}$. The class of problems with such approximation schemes is called $PTAS$.

From the definition follows that $PTAS \subseteq APX$. The question that arises now is whether $PTAS$ is *strictly* contained in APX. It was shown that $PTAS \subset APX$, unless $\mathcal{P} = \mathcal{NP}$ (see Ausiello *et al.* [AC *et al.* 99]).

As in the case of decision problems, we are interested how optimization problems relate one to another in terms of approximability. For this, several approximation preserving reductions were introduced over the years. One of the most used is the *AP-reducibility.*

Let Ψ_1 and Ψ_2 be two optimization problems in $\mathcal{NPO}$. Ψ_1 is said to be *AP-reducible* to Ψ_2, if there exist two functions g and h and a constant $\alpha \geq 1$ such that:

• For any instance $x \in \mathcal{I}_{\Psi_1}$ and for any rational number $r > 1, g(x,r) \in \mathcal{I}_{\Psi_2}$.

• For any instance $x \in \mathcal{I}_{\Psi_1}$ and for any rational number $r > 1$, if $S_{\Psi_1}(x) \neq \emptyset$ then $S_{\Psi_2}(g(x,r)) \neq \emptyset$.

• For any instance $x \in \mathcal{I}_{\Psi_1}$, for any rational number $r > 1$ and for any $y \in S_{\Psi_2}(g(x,r))$, $h(x,y,r) \in S_{\Psi_1}(x)$.

• g and h are computable by two algorithms $\mathcal{A}_g$ and $\mathcal{A}_h$, respectively, whose running time is polynomial for any fixed rational number r

• For any instance $x \in \mathcal{I}_{\Psi_1}$, for any rational number $r > 1$ and for any $y \in S_{\Psi_2}(g(x,r))$,

$$R_{\Psi_2}(g(x,r),y) \leq r \text{ implies } R_{\Psi_1}(x,h(x,y,r)) \leq 1 + \alpha(r-1),$$

where $R_{\Psi}(x,y) = \max\left\{\frac{f_\Psi(x,y)}{f^*_\Psi(x)}, \frac{f^*_\Psi(x)}{f_\Psi(x,y)}\right\}$.

Note that the $AP-$reducibility is transitive.

A problem Ψ is said to be $\mathcal{NPO}-$*complete*, respectively $APX-$*complete* if it belongs to $\mathcal{NPO}$, respectively APX and any problem $\Psi' \in \mathcal{NPO}$, respectively APX can be $AP-$reduced to Ψ. If we show that a problem Ψ is $\mathcal{NPO}-$complete, then we have that $\Psi \notin APX$, unless $\mathcal{P} = \mathcal{NP}$. Similarly, if we show that a problem Ψ is $APX-$complete, then we have that $\Psi \notin PTAS$, unless $\mathcal{P} = \mathcal{NP}$. As an example of an $\mathcal{NPO}-$complete problem we mention maximum weighted satisfiability and as an $APX-$complete problem, maximum 3-satisfiability.

Some important location problems are $APX-$complete as well. Guha and Khuller [GK99] showed that the metric $UFLP$ belongs to this class. Therefore the metric $UFLP$ is not in $PTAS$, unless $\mathcal{P} = \mathcal{NP}$.

For an in-depth presentation of complexity theory we recommend the books of Garey and Johnson [GJ84], Ausiello *et al.* [AC *et al.* 99] and the survey on hardness of approximations written by Arora and Lund [AL96].

1.3 Integer and linear programming

Quite often the set of feasible solutions of a combinatorial optimization problem can be described by linear inequalities and possibly integrality constraints on some of the variables, i.e., as an *integer linear program.*

In general, a (finite) integer linear program is a problem of type:

$$
\begin{aligned}
(ILP) \qquad & \text{minimize } c^T x \\
& \text{subject to } Ax \geq b \\
& \qquad x \geq 0,\ x \text{ integer,}
\end{aligned}
$$

where $c \in R^m, b \in R^n, A \in R^{mn}$. We will assume that A, b, c are integral.

The problem of solving an integer linear program is $\mathcal{NP}-$complete (for a detailed discussion on this topic see e.g. Schrijver [Sc87]). Hence, the formulation of a combinatorial problem as an (ILP) does not simplify the problem.

However, if the integrality condition imposed on the variables is omitted, the new problem, called a *linear program,* becomes polynomial. There are several polynomial algorithms for linear programming, e.g. the ellipsoid algorithm, algorithms based on interior point method etc.(see Bertsimas and Tsitsiklis [BT97] for a survey on recent developments on this topic). The best known asymptotic running time in the literature is $O(n^3 L)$, where $L = size(\max\{|\det(A')|\,|A' \text{ is a square submatrix of } A = (a_{ij})_{i=\overline{1,m},j=\overline{1,n}})\}$ $+size(\max_i(|b_i|)\ +size(\max_j(|c_j|)))$, achieved among others by Ye's algorithm ([Ye91]).

Linear programming is a very important tool in designing approximation algorithms. Before presenting how one may use linear programming in approximations, we will review some basic facts.

Consider a general linear program,

$$
\begin{aligned}
(LP) \quad & \text{minimize } \sum_{j=1}^{n} c_j x_j \\
& \text{subject to} \sum_{j=1}^{n} a_{ij} x_j \geq b_i,\ i = 1, ..., m \qquad (1.1) \\
& \qquad x_j \geq 0, \qquad , j = 1, ..., n. \qquad (1.2)
\end{aligned}
$$

A vector $x \in R^n$ is a *feasible solution* of (LP) if it satisfies constraints (1.1) and (1.2). The value $c^T x$ is called the *objective value* of x. If a feasible solution x^* satisfies $c^T x^* \leq c^T x$, for every feasible solution x, we say that x^* is an *optimal solution* of (LP) and its value cx^* is the *optimal value.* If, in addition, all the components of x^* are integers, x^* is an optimal solution of (ILP). Clearly, if x^* and $\widehat{x}$ are optimal solutions of (LP) respectively (ILP), then $c^T x^* \leq c^T \widehat{x}$.

We are now interested in finding a good lower bound for $c^T x^*$. One way of doing this is by finding suitable nonnegative multipliers for the constraints so that when we take their sum, the coefficient of each x_j in the sum is dominated by the coefficient in the objective function. More precisely, we are looking for $y_i \geq 0$, $1 \leq i \leq m$, such that for every j, $\sum_{i=1}^m a_{ij} y_i \leq c_j$. Once we have found the y_i's, we conclude that any feasible solution of (LP) satisfies

$$\begin{aligned} \sum_{j=1}^n c_j x_j &\geq \sum_{j=1}^n \left(\sum_{i=1}^m a_{ij} y_i \right) x_j \\ &= \sum_{i=1}^m \left(\sum_{j=1}^n a_{ij} x_j \right) y_i \geq \sum_{i=1}^m b_i y_i. \end{aligned} \tag{1.3}$$

Hence, $y^T b$ is a lower bound on $c^T x^*$. Clearly, we would like this lower bound to be tight. Therefore, the searching for the best multipliers y can be reduced to the problem of solving the following linear program, called the *dual* of (LP) ((LP) itself is called the *primal* program) :

$$\begin{aligned} (D) \qquad \text{maximize } & \sum_{i=1}^m b_i y_i \\ \text{subject to } & \sum_{i=1}^m a_{ij} y_i \leq c_j, \ j = 1, ..., n \\ & y_i \geq 0, \qquad\quad ,i = 1, ..., m. \end{aligned}$$

A feasible solution to (LP) is called a *primal feasible solution* and a feasible solution to (D) is called a *dual feasible solution.*

The following two theorems can be directly derived from (1.3).

Theorem 1.1 ***(Weak duality)*** *If x is a primal feasible solution and y is a dual feasible solution, then $y^T b \leq c^T x$.*

Theorem 1.2 ***(Complementary slackness conditions)*** *Let x be a feasible primal solution and y a feasible dual solution. Then, x and y are both optimal if and only if all of the following conditions are satisfied:*

Primal complementary slackness conditions

For each $1 \leq j \leq n$: either $x_j = 0$ or $\sum_{i=1}^{m} a_{ij} y_i = c_j$.

Dual complementary slackness conditions

For each $1 \leq i \leq m$: either $y_i = 0$ or $\sum_{j=1}^{n} a_{ij} x_j = b_i$.

The complementary slackness conditions play an important role in the design of efficient algorithms, as we will see in the next section.

We conclude this section with the central result on linear programming duality

Theorem 1.3 ***(Strong duality)*** *The primal program has finite optimum if and only if its dual has finite optimum. Moreover, if x^* and y^* are optimal solutions for the primal and dual program, respectively, then $c^T x^* = y^{*T} b$.*

For an introduction to the field of integer and linear programming see the books of Chvátal [Ch83] and Bertsimas and Tsitslikis [BT97].

1.4 Approximation algorithms based on linear programming

In this section we describe two ways of using linear programming in designing approximation algorithms: linear programming rounding and the primal dual-technique. For examples and a more in depth discussion on these techniques we refer to the books of Vazirani [Va01], and Ausiello *et al.* [AC *et al.* 99]. As before, we will refer to minimizations problems.

All the approximation algorithms based on linear programming contain the following general steps:

1. Formulate the problem as an integer linear program (ILP). Denote by OPT its optimal value.

2. Relax (ILP) to a linear program (LP). Denote by $\overline{OPT}$ its optimal value.

3. Use the linear program (LP) in order to obtain in polynomial time a feasible solution to (ILP) of value bounded by $\rho\overline{OPT}$.

Since (LP) is a relaxation of (ILP), we have $\overline{OPT} \leq OPT$. Hence, by applying the steps $(1)-(3)$ we obtain in polynomial time a feasible solution to (ILP) of value at most ρOPT.

The most difficult step is, of course, the third one. Next we will sketch two methods for realizing it.

1.4.1 Linear programming rounding

This method requires first to solve (LP). In most of the cases, the solution will not be integral. Using properties of the problem, in polynomial time, we round the solution of the (LP) to a feasible solution of (ILP), with value no more than $\rho\overline{OPT}$. As we have seen, this will lead to a feasible solution of (ILP) of value no larger than ρOPT.

In general the rounding step is not very easy and depends on the structure of the problem.

A special rounding technique, called *randomized rounding*, can be useful in the cases when some of the variables in the (ILP) are restricted to be in $\{0,1\}$. This technique was introduced by Raghavan and Thompson [RT87] and uses the following general idea.

Assume that in (ILP) $x_j \in \{0,1\}$ for some $j \in J$ and that in (LP) the integrality condition of these variables is relaxed to $0 \leq x_j \leq 1$, for every $j \in J$. Denote by x^* the optimal solution of (LP). Being in $[0,1]$, the variables x_j^*, for $j \in J$, can be interpreted as probabilities. Next, we round each variable x_j to 1 with probability $g\left(x_j^*\right)$, where $g:[0,1] \to [0,1]$ is chosen such that it can be computed in polynomial time, the rounded solution is feasible for (ILP) and

$$\sum_{j=1}^{n} c_j Prob(x_j = 1) \leq \rho\overline{OPT}, \tag{1.4}$$

Suppose for the moment that such a function g is found.

Clearly, since the rounding rule is probabilistic, we can receive different outputs by applying this algorithm. In this case, we are interested in the *expected cost* of the algorithm, which is the expected value of $\sum_{j=1}^{n} c_j x_j$,i.e.,

$$E(\sum_{j=1}^{n} c_j x_j) = \sum_{j=1}^{n} c_j Prob(x_j = 1).$$

Then, from (1.4) and from $\overline{OPT} \leq OPT$ we conclude that the expected cost of the rounded solution is within ρ times the optimal value of (ILP).

Depending on the way the function g is chosen, we distinguish *dependent randomized rounding* (the events of rounding the variables are dependent) and *independent randomized rounding* (the events of rounding the variables are independent).

We will see examples of dependent randomized rounding in Chapters 2 and 4 and examples of independent randomized rounding in Chapter 5.

The main drawback of the randomized approach is that even if we prove that an algorithm returns solutions of expected good quality, we may get poor approximate solutions with small probability. However, it is sometimes possible to overcome this drawback by *derandomizing* the algorithm, that is, by transforming the given randomized algorithm (with expected cost less than ρOPT, for a minimization problem) into a deterministic one, which always returns, in polynomial time, a solution with a value smaller than ρOPT. A standard method of derandomization is the *method of conditional expectations* due to Erdös and Selfridge [ES73] (for a detailed discussion of this method see also the book of Alon, Spencer and Erdös [ASE92]). All the randomized algorithms presented in this thesis can be derandomized using this method.

Basically, the method can be described as follows (we present the main idea for a minimization problem). Suppose that we have m $(0,1)$ random variables $u_1, u_2, ...u_m$ and we know that $E(f) = \overline{h}$ where $f = h(u_1, ..., u_m)$ is a nonnegative real valued function h on m variables. Our task is to determine a set of values $\overline{u_1}, \overline{u_2}, .., \overline{u_m}$ for the random variables $u_1, ..., u_m$ such that $h(\overline{u_1}, \overline{u_2}, .., \overline{u_m}) \leq \overline{h}$. We begin with u_1. We wish to decide whether to set u_1 to 0 or to 1. The expected value $E(f)$ is a convex combination of the conditional expectations $E(f|u_1 = 0)$ and $E(f|u_1 = 1)$, more precisely

$$E(f) = E(f|u_1 = 0)\,Prob(u_1 = 0) + E(f|u_1 = 1)\,Prob(u_1 = 1).$$

We would certainly have made the right decision if the conditional expectation decreases. Thus we set $\overline{u_1}$ to 1 if $E(f|u_1 = 1) \leq E(f|u_1 = 0)$ and to 0 otherwise. Note that

$E(f|u_1 = \overline{u_1}) \le E(f) = \overline{h}$. The process continues inductively. Suppose that we have already decided on the values $\overline{u_1}, .., \overline{u_t}$ in $\{0,1\}$ so that the conditional expected value $E(f|u_1 = \overline{u_1}, ..., u_t = \overline{u_t})$ is at most $\overline{h}$. Then,

$$E(f|u_1 = \overline{u_1}, ..., u_t = \overline{u_t}) = E(f|u_1 = \overline{u_1}, ..., u_t = \overline{u_t}, u_{t+1} = 0)\, Prob\,(u_{t+1} = 0) + \\ + E(f|u_1 = \overline{u_1}, ..., u_t = \overline{u_t}, u_{t+1} = 1)\, Prob\,(u_{t+1} = 1)\,.$$

We choose $u_{t+1} = 1$ if the expected value decreases, that is if
$E(f|u_1 = \overline{u_1}, ..., u_t = \overline{u_t}, u_{t+1} = 1) \le E(f|u_1 = \overline{u_1}, ..., u_t = \overline{u_t}, u_{t+1} = 0)$ and 0 otherwise. Clearly, at termination, we obtain a set of $(0,1)$ values $\overline{u_1}, \overline{u_2}, .., \overline{u_m}$ such that
$h(\overline{u_1}, \overline{u_2}, .., \overline{u_m}) \le \overline{h}$, as desired. Notice that we only need to compute $2m$ conditional expected values. However, for this process to work we have to be able to compute all the intermediate conditional expectations in polynomial time. Unfortunately, there are many interesting instances where this is not the case. A trick that can be useful in such cases is the introduction of *pessimistic estimators*, introduced by Raghavan [Ra98]. Suppose that, for each $0 \le t \le m$ we have a function $s_t(\overline{u_1}, ..., \overline{u_t})$ which can be efficiently computed and verifies the following inequalities:

$$\begin{aligned} s_0 &\ge \min\{s_1(0), s_1(1)\} \\ s_t(\overline{u_1}, ..., \overline{u_t}) &\ge \min\{s_{t+1}(\overline{u_1}, ..., \overline{u_t}, 0), s_{t+1}(\overline{u_1}, ..., \overline{u_t}, 1)\} \\ s_t(\overline{u_1}, ..., \overline{u_t}) &\ge E(f|u_1 = \overline{u_1}, ..., u_t = \overline{u_t})\,. \end{aligned}$$

If, at the beginning, $s_0 \le \overline{h}$ and we choose the values of u_t such as to minimize s_t in each step, we get at the end a point $(\overline{u_1}, ..., \overline{u_m})$ for which $s_m(\overline{u_1}, ..., \overline{u_m}) \le \overline{h}$. The functions s_t, $0 \le t \le m$, with the above properties are called *pessimistic estimators*.

We will see an example where this method is applied in Chapter 5.

1.4.2 Primal-dual algorithms

In order to apply the rounding technique described in the previous section one has to solve a linear program, which is a computationally expensive process. A different approach, still based on linear programming, allows us to obtain solutions more efficiently. This approach is called the *primal-dual method.* The method may be applied for designing both exact and approximate algorithms. In this section we will describe only the use of the method for the second type of algorithms. For more information about the primal-dual technique see the book of Vazirani [Va01] and the surveys of Vazirani [Va02] and Goemans and Williamson [GW97].

Consider the primal program (LP), the dual (D) and the complementary slackness conditions associated, in the form presented in the previous section. The method first relaxes the complementary slackness conditions as follows. Let $\alpha \geq 1$ and $\beta \geq 1$.The relaxed complementary slackness conditions are

Relaxed primal complementary slackness conditions

For each $j, 1 \leq j \leq n$: either $x_j = 0$ or $\frac{c_j}{\alpha} \leq \sum_{i=1}^{m} a_{ij} y_i \leq c_j$.

Dual complementary slackness conditions

For each $i, 1 \leq i \leq m$: either $y_i = 0$ or $b_i \leq \sum_{j=1}^{n} a_{ij} x_j \leq \beta b_i$.

It is straightforward to see that

Lemma 1.1 *If x and y are primal and dual feasible solutions satisfying the relaxed primal and dual complementary slackness conditions then*

$$\sum_{j=1}^{n} c_j x_j \leq \alpha\beta \sum_{i=1}^{m} b_i y_i \leq \alpha\beta \overline{OPT} \leq \alpha\beta OPT.$$

Hence, if we can find such x and y, primal respectively dual feasible, in polynomial time, we have an $\alpha\beta-$approximation algorithm. Usually x and y are constructed as follows. One starts with a primal infeasible solution and a dual feasible solution. We then iteratively improve the feasibility of the primal solution and the optimality of the dual one, ensuring that in the end a primal feasible solution is obtained and the relaxed complementary slackness conditions are satisfied. The primal solution is always kept integral, thus ensuring that the final solution is integral.

We will see examples of algorithms using the primal-dual technique in Chapters 2, 3 and 4.

1.5 Outline of the thesis

In this thesis we study approximation algorithms for different facility location problems.

Chapter 2 concentrates on the metric uncapacitated facility location problem ($UFLP$). We review the main results on the metric $UFLP$ (the transportation costs are induced by a metric), discussing in detail the known approximation algorithms for this problem.

In Chapter 3 we discuss an extension of the metric uncapacitated facility location problem, namely the fault tolerant facility location problem. In this version of the metric $UFLP$, each demand point requires to be assigned to a prescribed number (≥ 1) of open facilities. First we describe a simple LP based approximation algorithm , which achieves a performance guarantee of 4. For the case where the connectivity requirements are equal, we show that a simple extension of Jain and Vazirani's algorithm, together with a greedy improvement procedure developed by Guha, Meyerson and Munagala in [GMM01], gives a solution with value within 1.85 times the optimum.

Chapter 4 is dedicated to metric multilevel facility location problems ($MFLP$). In this problem there are k types of facilities to be opened: one type of depots and $(k-1)$ types of transit stations. Each demand must be shipped from a depot through transit stations of type $k-1, ..., 1$ to the demand points. We call level l of facilities all the facilities of type l. The transportation costs between demand points and facilities on the first level, as well as transportation costs between facilities on consecutive levels are known and they satisfy the triangle inequality. The goal is to decide which facilities to open and to assign demand points to paths along open facilities such that the total cost (the cost of opening facilities plus the assignment cost) is minimized. First we present the main ideas of a $3-$approximation algorithm based on $LP-$rounding proposed by Aardal, Chudak and Shmoys [ACS99] for the metric uncapacitated $MFLP$. Then we propose a $6-$approximation algorithm based on the primal-dual technique. Using Lagrangian multipliers, we extend the algorithm to a capacitated case (each facility i can serve at most $u_i \in N$ demand points) in which opening several copies of a facility to a location is allowed. For this capacitated version of the problem, the performance guarantee is 12. Finally, we show that a simple modification of the algorithm yields a $6-$approximation algorithm for the multilevel facility location problem with outliers. In this version of the multilevel facility location problem, only a specified fraction of the customers are to be served, for the others, the outliers, we may choose to pay a penalty and not to connect them to facilities.

In Chapter 5 we focus on facility location problems in the maximization version. The maximization version of the $UFLP$ is very similar to the minimization one. The only difference is that for each assignment of demand points to facilities, not the cost but the profit is given. The goal is to maximize the total profit, i.e. the profit obtained from the assignment minus the opening facility cost. After reviewing the known results for the case where there is only one level of facilities, we propose a $0.47-$approximation algorithm for the case where the facilities are situated on two levels. The main technique we use is that of randomized rounding.

Chapter 2

Approximation algorithms for the metric uncapacitated facility location problem

In Section 2.1 we review the existing results on the computational complexity of the *uncapacitated facility location problem (UFLP),* and we introduce an integer programming formulation of the problem together with an LP relaxation. For the latter we discuss some properties. In Section 2.2 we present the main ideas of the constant approximation algorithms developed for the metric $UFLP$. First we will focus on the algorithms that use linear programming rounding and then on the primal-dual algorithm of Jain and Vazirani [JV01]. The algorithms based on dual fitting will be briefly discussed at the end of the section. In Section 2.3 we discuss a greedy method of improving a solution of the metric $UFLP$, developed by Guha and Khuller [GK99] and by Charikar and Guha [CG99]. Finally, we will give some examples of algorithms using this method to obtain better performance guarantees.

2.1 Preliminaries

Recall the *uncapacitated facility location problem* ($UFLP$), introduced in Chapter 1.

Let $G = (V, E)$ be a bipartite graph with bipartition (F, D), where D is the set of *demand points* and F is the set of possible *facility locations.* Every demand point $j \in D$ has a demand d_j and every facility $i \in F$ can provide an unlimited amount of a certain commodity. We are given *an opening cost vector* $f \in R_+^{|F|}$ (i.e., opening a facility at $i \in F$ incurs a cost $f_i \geq 0$) and *a service cost vector* $c \in R_+^{|E|}$ (i.e., if demand point j is assigned to the open facility i a service cost of c_{ij} is incurred for satisfying every unit of demand of j). The goal is to determine a subset of the set of potential facility locations to open facilities and an assignment of demand points to these facilities so as to minimize the overall total cost, that is, the fixed costs of opening facilities plus the total service costs.

In this chapter we will focus on the *metric* version of this problem, when the service costs are induced by a metric on V.

Remark 2.1 *We will assume that each $j \in D$ has a demand of one unit. All the results presented here can be easily extended to arbitrary demands. Furthermore, we will assume that $f_i > 0$ for every $i \in F$. This is not a restrictive assumption, since if there is a facility i with no opening cost, we can open it at the end and assign to it all the demand points for which i is the closest open facility.*

We have seen in Chapter 1 that the $UFLP$ is $\mathcal{NP}$-hard (Cornuejols, Nemhauser and Wolsey [CNW90]). In fact there we reduced the vertex cover problem to the metric uncapacitated facility location problem, so that we can conclude that the metric $UFLP$ is NP-hard. Moreover, Guha and Khuller [GK99] have shown that it is $APX-$complete and that the best approximation factor we can get for this problem is 1.467 unless $\mathcal{NP} \subseteq TIME\left[n^{O(\log\log n)}\right]$ ($TIME\left[g\left(n\right)\right]$ is the class of problems that can be solved in deterministic time $g\left(n\right)$ on instances of size n). This result was improved by Sviridenko [Sv99], who proved that the performance guarantee of an approximation algorithm for the metric $UFLP$ cannot be smaller than 1.467 unless $\mathcal{P} = \mathcal{NP}$.

However, there are some special cases of the $UFLP$ that allow polynomial approximation schemes or can even be solved optimally.

For the uncapacitated facility location problem in the 2-dimension Euclidean space, Arora, Raghavan and Rao [ARR98] give a randomized polynomial approximation scheme. Kolliopoulos and Rao [KR99] generalized the result to Euclidean spaces of constant dimension.

Kolen [Ko83] has shown that the $UFLP$ is solvable in $O\left(r^3\right)$ when it is defined on a tree with r nodes and under the following assumptions. The demand points and the facility locations are nodes in a tree and the service cost c_{ij} is the length of the path between i and j (w.r.t. given nonnegative edge weights). Bárány, Edmonds and Wolsey

[BEW86] generalized the $UFLP$ on trees to a tree partioning problem, which they solved in $O\left(|V|^2\right)$ on a graph $G = (V, E)$. Recently, Beresnev [Be01] has proved that in the case where the $UFLP$ has the service costs c given as a matrix with totally balanced characteristic matrix, the problem can be solved in polynomial time as well. For more examples of polynomially solvable cases of the $UFLP$ see Cornuejols, Nemhauser and Wolsey [CNW90] and Ageev and Beresnev [AB90].

As we have seen in the previous chapter, an integer programming formulation of a problem and a linear programming relaxation of it can be of great help in designing approximation algorithms.

2.1.1 An integer programming formulation

To derive an integer programming formulation of the $UFLP$, we introduce the $(0,1)$ variables y_i $(i \in F)$ to indicate whether facility i is open and the $(0,1)$ variables x_{ij} $(j \in D, i \in F)$ to indicate whether demand point j is served by facility i. If j is served by i we will also say that j is assigned to i.

We let $c(x) = \sum_{i\in F,\, j\in D} c_{ij}x_{ij}$ and $f(y) = \sum_{i\in F} f_i y_i$. In other words, $c(x)$ represents the service cost (assignment cost) and $f(y)$ the cost of opening facilities.

The $UFLP$ is now equivalent with

$$
\begin{aligned}
(\mathbf{P}_{\text{int}}) \qquad & \text{minimize } c(x)+f(y) & & \\
& \text{subject to} \sum_{i\in F} x_{ij} \ge 1, & & \text{for each } j \in D \qquad (2.1)\\
& x_{ij} \le y_i, & & \text{for each } i \in F \text{ and } j \in D \qquad (2.2)\\
& x_{ij}, y_i \in \{0,1\} & & \text{for each } i \in F \text{ and } j \in D.
\end{aligned}
$$

Constraints (2.1) ensure that each demand point $j \in D$ is assigned to a facility, whereas constraints (2.2) ensure that demand points are served only by open facilities.

We will denote by C_{OPT} the optimum value for $(\mathbf{P}_{\text{int}})$.

Consider the following LP-relaxation of ($\mathbf{P}_{\text{int}}$):

$$\begin{aligned} (\mathbf{P}) \qquad & \text{minimize } c(x) + f(y) \\ & \text{subject to } (2.1)\,,\ (2.2) \\ & \qquad\qquad x_{ij} \geq 0 \text{ for each } i \in F \text{ and } j \in D. \end{aligned}$$

Note that $x_{ij} \leq 1$ and $y_i \leq 1$ hold automatically for any optimal solution (x, y) of ($\mathbf{P}$). Furthermore, inequality (2.2) implies that $y_i \geq 0$ for any solution of ($\mathbf{P}$).

We will denote by C_{LP} the optimum value of ($\mathbf{P}$). Clearly, $C_{LP} \leq C_{OPT}$. Moreover, Guha and Khuller [GK99] give a family of examples for which C_{OPT}/C_{LP} is about 1.463. This implies that any rounding strategy applied to a solution of ($\mathbf{P}$) will be forced to increase the cost of the integral solution by a factor of about 1.463.

Let (x, y) be a solution of ($\mathbf{P}$). For every demand point j and facility i for which $x_{ij} > 0$ we will say that j is *fractionally served* by i in (x, y).

The dual program of ($\mathbf{P}$) can be formulated as follows

$$\begin{aligned} (\mathbf{D}) \qquad & \text{maximize } \sum_{j \in D} v_j \\ & \text{subject to } v_j - t_{ij} \leq c_{ij}, \text{ for each } i \in F \text{ and } j \in D && (2.3) \\ & \qquad \sum_{j \in D} t_{ij} \leq f_i, \text{ for each } i \in F && (2.4) \\ & \qquad v_j, t_{ij} \geq 0, \text{ for each } i \in F \text{ and } j \in D \end{aligned}$$

Based on the complementary slackness condition, a very nice interpretation to the dual can be given (as proposed by Jain and Vazirani in [JV01]).

Let (x^*, y^*) and (v^*, t^*) be an optimal primal, respectively an optimal dual solution.

The primal and dual complementary slackness conditions are:

$(S1)$ $\forall\ i \in F,\ j \in D$: $x^*_{ij} > 0$ implies that $v^*_j = c_{ij} + t^*_{ij}$

$(S2)$ $\forall\ i \in F$: $y^*_i > 0$ implies that $\sum\limits_{j \in D} t^*_{ij} = f_i$

$(S3)$ $\forall\ j \in D$: $v^*_j > 0$ implies that $\sum\limits_{i \in F} x^*_{ij} = 1$

$(S4)$ $\forall\ i \in F,\ j \in D$: $t^*_{ij} > 0$ implies that $y^*_i = x^*_{ij}$.

Let i be a facility for which $y_i^* > 0$. Then $\sum\limits_{j \in D} t_{ij}^* = f_i$ by $(S2)$. Thus, we can think of t_{ij}^* as the amount that j is willing to contribute for opening facility i. For a facility i to be opened, the demand points have to pay the entire cost f_i. This interpretation is also sustained by the following fact. Let $j \in D$ be a demand point that is not fractionally served by i in (x^*, y^*), i.e., $x_{ij}^* = 0$. Since $y_i^* \neq x_{ij}^*$, $(S4)$ implies that $t_{ij}^* = 0$, so a demand point j is not willing to pay for a facility to which it is not assigned to.

Now let $j \in D$ be a demand point that is fractionally served by i in (x^*, y^*). Condition $(S1)$ implies that $v_j^* = c_{ij} + t_{ij}^*$. Hence, if we consider c_{ij} as the price that j has to pay for using edge (i, j), v_j^* can be interpreted as the total price demand point j is willing to pay.

2.2 Approximation algorithms

In the next sections we will use the notations $|D| = n_d$, $|F| = n_f$ and $n = n_f + n_d$.

For the problem with arbitrary service costs, Hochbaum [Ho82] proposed a greedy algorithm with a $O(\log n_d)$ performance guarantee. Later on, Lin and Vitter [LV92] obtained the same guarantee by using a new technique, called the filtering technique. This method consists in constructing a "filtered problem" by fixing some variables to zero in the integer programming formulation. One has to decide which variables to set to zero such that every integral solution of the filtered problem has the objective value within a factor of $(1 + \varepsilon)$ of the optimal value of the original linear program. Then one finds a feasible integer solution to the filtered program, either by rounding a fractional solution, i.e., a solution to the linear programming relaxation, or by using some combinatorial algorithm.

A large number of approximation algorithms, using a variety of techniques, have been proposed during the recent years for the metric $UFLP$. In Table I we give a short review of these results. We will discuss in more detail some of the techniques used in these algorithms, in order to provide a foundation for the next chapters.

Performance guarantee	Reference	Technique/running time
3.16	Shmoys, Tardos and Aardal [STA97]	filtering+LP rounding
2.47	Guha and Khuller [GK99]	LP rounding+greedy augmentation
1.736	Chudak and Shmoys [C98a], [C98b], [CS98]	LP rounding
$5+\varepsilon$	Korupolu *et al.* [KPR 98]	local search/ $O\left(n^6 \log\left(n/\varepsilon\right)\right)$
3	Jain and Vazirani [JV01]	primal-dual method/ $O\left(n^2 \log n\right)$
3	Arya *et al.* [AG *et al.* 01]	local search
3	Mettu and Plaxton [MP00]	combinatorial/ $O\left(n^2\right)$
1.85	Charikar and Guha [CG99]	primal-dual method+greedy augmentation/ $O\left(n^3\right)$
1.72	Charikar and Guha [CG99]	LP rounding+primal-dual method+greedy augmentation
1.86	Mahdian *et al* [MMSV01]	dual fitting/ $O\left(n^2 \log n\right)$
1.61	Jain, Mahdian and Saberi [JMS01]	dual fitting/ $O\left(n^3\right)$
1.52	Mahdian, Ye and Zhang [MYZ02]	dual fitting+greedy . augmentation/ $O\left(n^3\right)$

TABLE I

Approximation algorithms for the $UFLP$

2.2.1 Algorithms for the metric UFLP based on linear programming rounding

The algorithms based on linear programming rounding first find an optimal solution to $(\mathbf{P})$ and then round it to an integer solution (x, y) that satisfies

$$c(x) + f(y) \leq \rho C_{LP} \leq \rho C_{OPT},$$

where ρ is the performance guarantee of the algorithm.

We will present the main ideas of these algorithms with the help of a simplified version of Shmoys, Tardos and Aardal's algorithm, due to Chudak ([C98a]).

First we introduce some notations and definitions.

Let (x^*, y^*) and (v^*, t^*) be optimal solutions to $(\mathbf{P})$, respectively $(\mathbf{D})$.

For every $j \in D$, we will call the *neighborhood* of j the set of facilities that fractionally serve j, i.e.,

$$N_j = \left\{ i \in F |\ x^*_{ij} > 0 \right\}.$$

Some properties of a neighborhood are listed in the following lemma.

Lemma 2.1 (i) *For every* $j \in D$ *and* $i \in N_j$, $c_{ij} \leq v_j^*$.

(ii) *For every* $j \in D$, $\sum_{i \in N_j} x_{ij}^* = 1$ *and* $\sum_{i \in N_j} y_i^* \geq 1$.

(iii) *Let* f_{i_0} *be the facility with the minimum opening cost in the neighborhood* N_j *of a demand point* j. *Then*

$$f_{i_0} \leq \sum_{i \in N_j} f_i y_i^*.$$

(iv) *Let* j, j' *be two demand points with* $N_j \cap N_{j'} \neq \emptyset$ *and* $v_j^* \leq v_{j'}^*$. *Then for every* $i \in N_j$, $c_{ij'} \leq 3v_{j'}^*$.

Proof. (i) Follows from the complementary slackness condition $(S1)$

(ii) Follows from (2.1) and (2.2).

(iii) Follows from (ii) and from $f_{i_0} \leq f_i$ for every $i \in N_j$.

(iv) Let $i' \in N_j \cap N_{j'}$ (see Figure 2.1).

By the triangle inequality,

$$c_{ij'} \leq c_{i'j'} + c_{i'j} + c_{ij}.$$

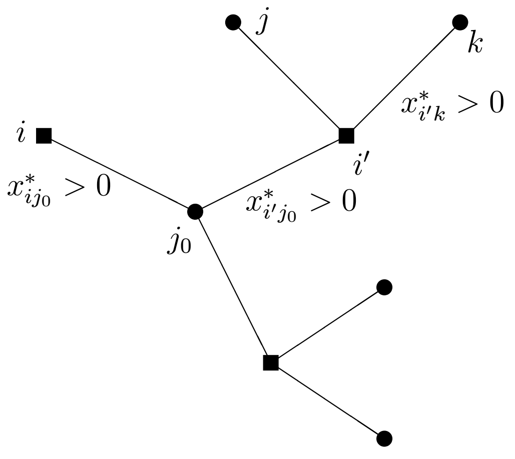
Finally, using (i) and $v_j^* \leq v_{j'}^*$ we obtain that $c_{ij'} \leq 3v_{j'}^*$. ■

The rounding algorithm can be described as follows.

It starts by clustering (partitioning) the demand points into *clusters* (blocks). Each cluster has a unique *cluster center* $j \in D$. The set of centers will be denoted by $\mathcal{C} \subseteq D$. The clustering along with the corresponding centers is constructed as follows: We pass through the demand points in increasing order of v_j^* :

$$v_1^* \leq ...v_n^*.$$

Node j becomes a (new)cluster center if $N_j \cap N_{j_0} = \emptyset$ for all cluster centers j_0, $j_0 \leq j$, constructed so far. Otherwise, if $N_j \cap N_{j_0} \neq \emptyset$ for some center j_0, we add j to the cluster centered at j_0. Having constructed the clustering, we open the cheapest facility $i \in N_{j_0}$ for each center $j_0 \in \mathcal{C}$ and assign each demand point in this cluster to $i \in F$.



● are demand points and ■ are facilities

Figure 2.1: Cluster centered at j_0

```
$\mathcal{S} := D$, $\mathcal{C} := \emptyset$
WHILE $\mathcal{S} \neq \emptyset$ DO
        Choose $j \in \mathcal{S}$ with the smallest $v_j$ value ($j \in \mathcal{S}$)
        IF $N_j \cap N_{j_0} \neq \emptyset$ for some $j_0 \in \mathcal{C}$ THEN add $j$ to the cluster
        centered at $j_0$
        ELSE $\mathcal{C} := \mathcal{C} \cup \{j\}$ and add to $C_j$ the set $N_j$
        $\mathcal{S} := \mathcal{S} \backslash \{j\}$
FOR each $j \in \mathcal{C}$
        Open the cheapest facility $i \in N_j$ and assign all demand points
        in this cluster to $i$
```

Remark 2.2 *Note that the clustering procedure assigns every demand point j to a cluster centered in some point j_0 with $v^*_{j_0} \leq v^*_j$. The facilities that are in a cluster centered at j_0 are in fact the facilities in N_{j_0}. Hence, $\sum\limits_{i \in C_j} x^*_{ij} = 1$. Furthermore, the neighborhoods of any two centers are disjoint.*

Pictorially, the cluster centered at j might look as in Figure 2.1:

The integer solution obtained as a result of this algorithm is

$$y_i = \begin{cases} 1, \text{ if } i \text{ is opened by the algorithm} \\ 0, \text{ otherwise} \end{cases}$$

and

$$x_{ij} = \begin{cases} 1, \text{ if } j \text{ is assigned to } i \text{ by the algorithm} \\ 0, \text{ otherwise} \end{cases} .$$

Since the neighborhoods of two cluster centers in $\mathcal{C}$ are disjoint, by Remark 2.2 the cost of opening facilities incurred by (x, y) can be bounded by

$$f(y) \le \sum_{j_0 \in \mathcal{C}} f_{i_0(j_0)} \le \sum_{j_0 \in \mathcal{C}} \sum_{i \in N_{j_0}} f_i y_i^* \le C_{LP},$$

where $i_0(j_0)$ is the facility with the minimum cost in the cluster centered at j_0, C_{j_0}.

By Lemma 2.1 (iv) and by the way of choosing cluster centers, for every demand point j there is an open facility within distance $3v_j^*$. Therefore, the assigning cost incurred by (x, y) can be bounded by

$$c(x) \le 3 \sum_{j \in D} v_j^* = 3C_{LP}.$$

We have proved

Theorem 2.1 *The above solution (x, y) satisfies $c(x) + f(y) \le 4C_{LP} \le 4C_{OPT}$.*

The algorithm developed by Shmoys, Tardos and Aardal differs from the one presented here only in the fact that it does not use the optimal dual solution, but a solution obtained by using the filtering technique of Lin and Vitter [LV92]. This solution allows to improve the performance guarantee to 3.16. For details, see [STA97].

The main changes Chudak brought to the algorithm above in order to reduce the performance guarantee from 4 to 1.73 are in the way of choosing the cluster centers and opening facilities.

When selecting a new cluster center, choose the demand point j with minimum $v_j^* + \sum_{i \in F} c_{ij} x_{ij}^*$ (instead of choosing the demand point j with the minimum v_j^*).

Note that since $x_{ij}^* \ge 0$ and $\sum_{i \in N_j} x_{ij}^* = 1$, in each cluster the x_{ij}^*'s define a probability distribution. In every cluster centered at some $j \in \mathcal{C}$ open a facility i at random, with probability x_{ij}^*. Open independently each facility i that is not contained in the neighborhood of any center with probability y_i^*. This is the crucial point of the algorithm, since it insures that with large probability (larger than $1 - \frac{1}{e} \simeq 0.63$) every demand point will have an open facility in its neighborhood. Finally, assign each demand point to the open facility in its cluster.

For a detailed analysis of this algorithm see [C98a].

2.2.2 The primal-dual algorithm of Jain and Vazirani and related algorithms

Jain and Vazirani's algorithm

A very simple combinatorial algorithm for the $UFLP$ is the one developed by Jain and Vazirani [JV01]. We will present the algorithm in a form slightly different from that in [JV01], in order to emphasize the structures that allow generalizations to other facility location problems.

Jain and Vazirani follow in their algorithm the main lines of the primal-dual technique. They construct, in polynomial time, an integer feasible solution (x, y) of $(\mathbf{P})$ and a dual feasible solution (v, t) of $(\mathbf{D})$ that satisfy the dual complementary slackness conditions and satisfy a relaxation of the primal ones. The relaxed complementary slackness conditions are:

$(S1')$ $\forall\ i \in F$ and $j \in D$ with $x_{ij} > 0$, $\frac{1}{3}c_{ij} \leq v_j - t_{ij} \leq c_{ij}$

Once (x, y) and (v, t) are obtained, the cost of the solution (x, y) can be bounded by

$$\begin{aligned} c(x) + 3f(y) &= \sum_{i\in F, j\in D} c_{ij}x_{ij} + 3\sum_{i\in F} f_i y_i \leq \sum_{i\in F, j\in D} c_{ij}x_{ij} + 3\sum_{i\in F, j\in D} t_{ij}x_{ij} \\ &= \sum_{j\in D}\sum_{i\in F}(c_{ij} + 3t_{ij})\,x_{ij} \leq 3\sum_{j\in D} v_j, \end{aligned}$$

where the first inequality is based on $(S2)$ and on $(S4)$ and the second inequality on $(S1')$ and on $(S3)$. Furthermore, since

$$c(x) + 3f(y) \leq 3\sum_{j\in D} v_j \leq 3C_{LP} \leq 3C_{OPT},$$

Jain and Vazirani's algorithm is a $3-$approximation algorithm.

Before presenting the construction of (x, y) and (v, t) with the above properties, we introduce some notations with respect to an arbitrary feasible solution (v, t) of $(\mathbf{D})$. This is motivated by the interpretation of the dual variables (cf. Section 2.2).

A demand point j is *willing to contribute* towards opening a facility i if $t_{ij} > 0$. Of course, t_{ij} represents the *contribution* he is willing to pay for opening facility i.

A facility $i \in F$ is *fully paid* when

$$\sum_{j \in D} t_{ij} = f_i.$$

A demand point $j \in D$ *has reached* a facility i when $v_j \geq c_{ij}$. If, in addition, i is fully paid, we say that j gets *connected.* We will call i a *connecting facility* for j.

We introduce a notion of time, so that each event can be associated with the time when it happens.

The algorithm starts at time 0, with $v \equiv t \equiv 0$. Increase all v_j uniformly (”with unit speed”). When some $j \in D$ reaches a not fully paid facility $i \in F$, start increasing t_{ij} with unit speed, until f_i is fully paid. When j gets connected, stop increasing v_j and t_{ij}, for every $i \in F$.

More precisely, the algorithm proceeds as follows.

```
UNTIL all $j \in D$ are connected DO
    Increase $v_j$ for all $j \in D$ not yet connected
    Increase $t_{ij}$ for all $i \in F$, $j \in D$ satisfying
      • $j$ has reached $i$
      • $j$ is not yet connected
      • $i$ is not yet fully paid
```

The algorithm has the property that, at time T, the dual variables v_j that are still raised are equal to T. Clearly, (v, t) is a dual feasible solution.

The next step is to construct a feasible integer solution (x, y) to ($\mathbf{P}_{\text{int}}$). A natural integer solution is the one obtained by opening all the fully paid facilities and assigning each demand point to the closest open facility. Unfortunately, this solution may not satisfy the dual complementary condition $(S4)$, since for a $j \in D$ there will be only one facility i with $x_{ij} > 0$, but there can be more facilities $i' \in F$ with $y_{i'} > 0$ and $t_{i'j} > 0$. In order to have condition $(S4)$ satisfied, one has to open facilities such that each demand point is willing to contribute for opening at most one facility.

One way to do this is as follows.

For every fully paid facility i, let T_i be the time when i became fully paid.

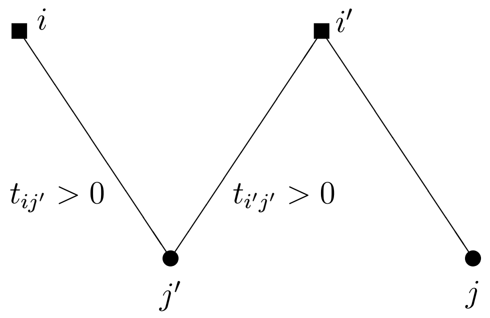


Figure 2.2:

To every fully paid facility i we associate *a set of clients,* which is the set of demand points willing to contribute for opening i, i.e,

$$D_i = \{j \in D|\ t_{ij} > 0\}.$$

Lemma 2.2 *(i) For every demand point j that reached a facility i we have $c_{ij} \leq v_j$.*

(ii) If $j \in D_i$, then $v_j \leq T_i$.

(iii) If i is a connecting facility for some j then $T_i \leq v_j$.

(iv) If i and i' are two fully paid facilities with $D_i \cap D_{i'} \neq \emptyset$ and i' is a connecting facility for some demand point j, then $\frac{1}{3}c_{ij} \leq v_j - t_{ij} \leq c_{ij}$.

Proof. (i) and (ii) follow immediately from the fact that, if j has reached i, $v_j = c_{ij}+t_{ij}$ and j gets connected at most at time T_i.

(iii) If $j \in D_i$ and i is a connecting facility for j then $v_j = T_i$. Otherwise, it means that j reached i after T_i, so $T_i \leq v_j$.

(iv) If j has reached i then $v_j - t_{ij} = c_{ij}$. Hence, in this case both inequalities in the proposition are verified.

If j did not reach i, $t_{ij} = 0$ and $v_j \leq c_{ij}$ and the second inequality follows immediately.

Let j' be a point in $D_i \cap D_{i'}$ (see Figure 2.2).

From the triangle inequality we conclude that

$$c_{ij} \leq c_{i'j} + c_{i'j'} + c_{ij'}.$$

Since $j' \in D_i \cap D_{i'}$, j' reached i and i'. (i) implies that $c_{i'j'} \leq v_{j'}$ and $c_{ij'} \leq v_{j'}$. Moreover, by (ii), $v_{j'} \leq \min\{T_{i'}, T_i\}$. Since i' is a connecting facility for j, $c_{i'j} \leq v_j$ and $T_{i'} \leq v_j$. Thus, $v_{j'} \leq v_j$. Now we can bound c_{ij} by

$$c_{ij} \leq 2v_{j'} + v_j \leq 3v_j.$$

■

For every fully paid facility i, denote by F_i the set of fully paid facilities with the property that the set of clients associated to them intersect D_i, i.e,

$$F_i = \{i' \in F |\ i' \text{ is fully paid and } D_i \cap D_{i'} \neq \emptyset\}.$$

The way of opening facilities can now easily be described.

Let S be the set of fully paid facilities. In every iteration, open a facility i in S and construct a cluster C_i, centered at i, from all the demand points j with the property that there is an $i' \in F_i$ that is a connecting facility for j. Set $S = S \backslash F_i$ and repeat the procedure until $S = \emptyset$.

```
$S = \{i \in F |\ i \text{ is fully paid}\}$
UNTIL $S = \emptyset$
  Open an $i \in S$ and construct the cluster $C_i$, where
        $C_i = \{j \in D |\ \exists\ i' \in F_i$ s.t. $i'$ is a connecting facility for $j\}$.
  Set $S = S \backslash F_i$
```

Remark 2.3 *Note that the set of clients associated to two open facilities are disjoint and that each demand point is assigned to a cluster. Each demand point contributes to the opening cost of at most one facility. There is an open facility for each cluster.*

Assign the demand points to facilities as follows. Assign every demand point j to the open facility i to which he is contributing, if such a facility exists. Otherwise, assign j to the open facility in the cluster to which j belongs.

The primal solution obtained in this way is

$$y_i = \begin{cases} 1, \text{ if } i \text{ is open} \\ 0, \text{ otherwise} \end{cases}$$

and

$$x_{ij} = \begin{cases} 1, & \text{if } j \text{ was assigned to } i \text{ by the algorithm} \\ 0, & \text{otherwise} \end{cases}.$$

Remark 2.4 *Note that each demand point j is assigned to an open facility i that became fully paid at some time $T_i \leq v_j$. Hence, if we stop increasing v_j at time T_i, from the dual obtained in this way we can construct the same primal solution (x, y). This remark will be very useful in extending the algorithm at the case where each demand point j requires to be connected not to one, but to r_j open facilities. The latter extension of the $UFLP$ will be studied in Chapter 3.*

It is straightforward to verify, with the help of Lemma 2.2, that (x, y) and (v, t) are feasible to (**P**), respectively (**D**) and that they are satisfying $(S1')$, $(S2)$, $(S3)$ and $(S4)$.

Hence, we have proved that

Theorem 2.2 *Jain and Vazirani's algorithm obtains an integer feasible solution (x, y) to (P_{int}) that satisfies*

$$c(x) + 3f(y) \leq 3C_{OPT}.$$

In [JV01] it is proved that the analysis of this algorithm is tight and that the running time is $O(n^2 \log n)$, where $n = n_d + n_f$. Therefore, it is a $3-$ approximation algorithm.

Charikar and Guha [CG99] showed that Jain and Vazirani's algorithm can construct a dual whose value is $3 - \varepsilon$ away from the optimum for arbitrary small ε. This means that it is not possible to prove an approximation ratio better than $3 - \varepsilon$ using the dual constructed as lower bound.

Related algorithms

The combinatorial algorithm design by Mettu and Plaxton [MP00] is very similar to the one above. For every facility i, they find a quantity T_i such that

$$\sum_{j \in D} \max\{T_i - c_{ij}, 0\} = f_i. \tag{2.5}$$

Then, by using a clustering similar to the one in Jain and Vazirani's algorithm, they prove a $3-$performance guarantee.

Although the algorithms for the $UFLP$ using dual fitting ([MMSV01], [JMS01], [MYZ02]) are based on a different idea, they result in a similar construction as the Jain and Vazirani algorithm.

The central idea is the following. If for a vector v and for every facility i and set of k demand points holds that

$$\sum_{j=1}^{k} v_j \leq \beta \left(f_i + \sum_{j=1}^{k} c_{ij} \right). \tag{2.6}$$

then the pair $\beta^{-1}\left(v, \overline{t}\right)$ with $\overline{t_{ij}} = \max\{v_j - \beta c_{ij}, 0\}$ is a dual feasible solution. By the weak duality theorem, $\beta^{-1} \sum\limits_{j \in D} v_j$ is a lower bound for the optimal solution and therefore the approximation ratio of the algorithm is β.

The first algorithm for the $UFLP$ using this technique was developed by Mahdian, Markakis, Saberi and Vazirani [MMSV01]. They construct the pair (v, t) exactly as in Jain and Vazirani's algorithm, with the only difference that a facility i is considered fully paid when $\sum\limits_{j \text{ unconnected}} t_{ij} = f_i$. Once a facility becomes fully paid, it keeps this property until the end of the algorithm. All the fully paid facilities are opened and each demand point is assigned to the closest open facility. This modification in Jain and Vazirani's algorithm has the advantage that a facility f_i is fully paid only by the contribution of demand points that are connected to it. However, it can happen that, for some facility i, $\sum\limits_{j \in D} t_{ij} > f_i$, so that the pair (v, t), with $t_{ij} = \max\{v_j - c_{ij}, 0\}$, might be infeasible. Fortunately, it can be shown that relation (2.6) is satisfied for $\beta = 1.86$, so that we can prove that the algorithm has a 1.86 performance guarantee by constructing $\overline{t}$ with $\beta^{-1}\left(v, \overline{t}\right)$ feasible.

This algorithm was improved by Jain, Mahdian and Saberi in [JMS01]. The change they made to the algorithm in [JV01] and [MMSV01] is inspired again by the interpretation of the dual program.

Suppose a demand point j gets connected to a facility i.

In the algorithm designed by Jain and Vazirani, j could contribute towards fully paid facilities that finally did not serve j, while in the algorithm designed by Mahdian *et al.* the contribution of j towards opening a facility i', $i' \neq i$, was not considered anymore.

In [JMS01] a more realistic way of contributing is considered: a demand point j has to contribute towards opening i' with the gain that j will have if i' is opened and j will be served by i' instead of i. More precisely, a facility i is fully paid when $\sum\limits_{j \in D} t_{ij} = f_i$. We construct (v, t) exactly as in [JV01] with the difference that after a demand point gets connected, say to a facility i, we set $t_{i'j} = \max\{c_{ij} - c_{i'j}, 0\}$ for all the facilities $i' \neq i$.

As a consequence, the solution found by this algorithm has the property that there is no closed facility that one can open to decrease the cost (without closing any other facilities). Because of this improvement, the performance guarantee drops to 1.61.

The algorithm for the $UFLP$ with the best known performance guarantee, developed by Mahdian, Ye and Zhang [MYZ02], combines the algorithm of Jain, Mahdian and Saberi with the greedy improvement technique we will describe in the next section.

2.2.3 The greedy improvement heuristic and scaling

In this section we will describe a very useful tool for improving the performance guarantee of approximation algorithms for facility location problems, namely the greedy improvement heuristic.

We will present very briefly this technique and show how, combined with scaling, it can improve the performance guarantee of a particular type of algorithms for the $UFLP$. All details omitted here can be found in [GK99], [Gu00] and [CG99].

Let (x, y) be a solution to ($\mathbf{P}_{\text{int}}$). Suppose that we have obtained (x, y) as a result of a polynomial algorithm ALG.

We will use the term *adding* a facility i to a solution for opening i and assigning to i all the demand points for which the assignment cost decreases by this operation.

Define the *Gain(i)* of a facility i to be the decrease in the total cost (decrease in service cost minus the facility cost of i) of the solution if facilitiy i would be added to a solution. The quantity $\frac{Gain(i)}{f_i}$ is the *gain ratio* of facilty i.

Lemma 2.3 *([CG99]) Let (x, y) be the current solution. For every integral solution $(\widehat{x}, \widehat{y})$, there exists a facility with gain ratio larger than $\frac{c(x) - c(\widehat{x}) - f(\widehat{y})}{f(\widehat{y})}$.*

Proof. Suppose we add a facility $i_0 \in F$ opened by $(\widehat{x}, \widehat{y})$ (i.e., with $\widehat{y_{i_0}} = 1$) to the set of the facilities opened by (x, y). The gain is formed from the gain obtained by assigning to i_0 demand points j for which i is closer than the facility to which they are assigned by (x, y), minus the cost of opening facility i_0. Hence,

$$Gain\,(i_0) \geq -f_{i_0} + \sum_{j \in D} \left(\sum_{i \in F} c_{ij} x_{ij} - c_{i_0 j} \right) \widehat{x_{i_0 j}}, \tag{2.7}$$

since the second term on the right hand side represents the assignment gain that is obtained by connecting all $j \in D$ to i_0 if they are connected to i_0 by $\widehat{x}$ (even though this may increase the current assignment cost for some $j \in D$). Note that the gain might be negative. Furthermore, not that (2.7) also holds in case i_0 is opened by (x, y) , since $Gain\,(i_0) = 0$ and $\sum\limits_{i \in F} c_{ij} x_{ij} \leq c_{i_0 j}$.

Summing the gain obtained by adding to (x, y) all facilities opened by $(\widehat{x}, \widehat{y})$, and taking into account that since $(\widehat{x}, \widehat{y})$ is integral, $\widehat{x}_{ij} \widehat{y}_i = \widehat{x}_{ij}$, we obtain

$$\sum_{i_0 \in F} Gain\,(i_0)\, \widehat{y_{i_0}} \geq -\sum_{i_0 \in F} f_{i_0} \widehat{y_{i_0}} + \sum_{i_0 \in F} \sum_{j \in D} \left(\sum_{i \in F} c_{ij} x_{ij} - c_{i_0 j} \right) \widehat{x_{i_0 j}}.$$

Since $\sum\limits_{i_0 \in F} \widehat{x_{i_0 j}} = 1$, the total gain is equal to

$$\sum_{i \in F} Gain\,(i)\, \widehat{y}_i \geq c\,(x) - c\,(\widehat{x}) - f(\widehat{y}).$$

Suppose now that for every facility opened by $(\widehat{x}, \widehat{y})$, $\frac{Gain(i)}{f(i)} < \frac{c(x) - c(\widehat{x}) - f(\widehat{y})}{f(\widehat{y})}$. This would imply that

$$\sum_{i \in F} Gain\,(i)\, \widehat{y}_i < \frac{c\,(x) - c\,(\widehat{x}) - f(\widehat{y})}{f(\widehat{y})} \sum_{i \in F} f_i \widehat{y}_i = c\,(x) - c\,(\widehat{x}) - f(\widehat{y}).$$

Hence, there is a facility i opened by $(\widehat{x}, \widehat{y})$ with gain ratio larger than $\frac{c(x) - c(\widehat{x}) - f(\widehat{y})}{f(\widehat{y})}$. ■

Remark 2.5 *Lemma 2.3 holds also for the case when the solution $(\widehat{x}, \widehat{y})$ is fractional (see [CG99]).*

The greedy improvement heuristic proceeds iteratively. In every iteration we pick a facility i with the maximum gain ratio. If the ratio is positive, we add i to the solution, otherwise we stop.

Lemma 2.4 *([CG99]) If (x, y) is an initial solution and if there exists a (possibly fractional) solution $(\widehat{x}, \widehat{y})$, then the solution obtained at the end of the heuristic will incur a total cost of at most*

$$f\,(y) + f(\widehat{y}) + c\,(\widehat{x}) + f(\widehat{y}) \max \left\{ 0, \ln \frac{c\,(x) - c\,(\widehat{x})}{f(\widehat{y})} \right\}.$$

Proof. We will give only a sketch of the proof. Denote by C_i respectively F_i the assignment cost, respectively the facility cost, associated with the solution obtained in the $i-th$ iteration of the greedy improvement procedure. Initially, $C_0 = c(x)$ and $F_0 = f(y)$. If $C_0 \leq c(\widehat{x}) + f(\widehat{y})$, the lemma is true. Suppose $C_0 > c(\widehat{x}) + f(\widehat{y})$.

Since in every iteration l we add the facility with the best gain ratio,

$$\frac{C_l - C_{l+1} - (F_{l+1} - F_l)}{F_{l+1} - F_l} \geq \frac{C_l - c(\widehat{x}) - f(\widehat{y})}{f(\widehat{y})}$$

and therefore the cost of a facility i added in the l-th iteration can be bounded by $f_i \leq \frac{C_l - C_{l+1}}{C_l - c(\widehat{x})} f(\widehat{y})$. Furthermore, $\frac{C_l - C_{l+1}}{C_l - c(\widehat{x})} = 1 - \frac{C_{l+1} - c(\widehat{x})}{C_l - c(\widehat{x})} \leq \ln \frac{C_l - c(\widehat{x})}{C_{l+1} - c(\widehat{x})}$, where the last inequality is derived from the fact that for all $x, 0 < x \leq 1$, $1 - x \leq \ln(1/x)$. Hence,

$$f_i \leq \ln \frac{C_l - c(\widehat{x})}{C_{l+1} - c(\widehat{x})}. \tag{2.8}$$

Denote by m the iteration with the property that for every $i, i < m$, $C_i > c(\widehat{x}) + f(\widehat{y})$ and $C_m \leq c(\widehat{x}) + f(\widehat{y})$. Since in every iteration the cost decreases, it is sufficient to bound the total cost at the m-th iteration. Combining (2.8) with the bound on C_m one receives for the cost of the final solution the bound announced in the lemma. ■

Charikar and Guha's algorithm for the $UFLP$ carries out the three steps described below:

• Scale the facility costs by a factor of δ
• Run for the scaled instance the algorithm ALG
• Scale back the solution and apply greedy improvement

Lemma 2.5 *If (x, y) is a feasible solution of ($\boldsymbol{P}_{int}$) such that $c(x) + \rho f(y) \leq \rho C_{OPT}$ then, by applying Charikar and Guha's algorithm, the $UFLP$ can be approximated within a factor of* $\min_\delta \max \left\{2 - \frac{1}{\rho\delta}, 1 + \frac{\rho - 1}{\rho\delta}, 1 + \ln(\rho\delta)\right\}$.

Proof. Denote by ($\mathbf{P}_{\text{sc}}$) the integer program corresponding to the problem with scaled opening facility costs.

Let (x^*, y^*) be an optimal solution for ($\mathbf{P}_{\text{int}}$) and (x', y') the feasible solution for ($\mathbf{P}_{\text{sc}}$) after the second step is completed.

Since (x^*, y^*) is feasible for ($\mathbf{P}_{\text{sc}}$), the optimal solution of ($\mathbf{P}_{\text{sc}}$) will have a total cost of at least $c(x^*) + \delta f(y^*)$.

Being obtained by applying ALG, (x', y') verifies

$$c(x') + \rho\delta f(y') \leq \rho\left(c(x^*) + \delta f(y^*)\right). \tag{2.9}$$

Clearly, $(x'y')$ is also feasible for ($\mathbf{P}_{\text{int}}$) and incurs a total cost of $c(x') + f(y')$.

If $c(x') \leq c(x^*) + f(y^*)$, then

$$\begin{aligned} c(x') + f(y') &\leq \frac{c(x') + \rho\delta f(y')}{\rho\delta} + \left(1 - \frac{1}{\rho\delta}\right) c(x') \\ &\leq \frac{\rho c(x^*) + \delta f y^*}{\delta} + \left(1 - \frac{1}{\rho\delta}\right)\left(c(x^*) + f(y^*)\right) \\ &\leq \left(1 + \frac{\rho - 1}{\rho\delta}\right) c(x^*) + \left(2 - \frac{1}{\rho\delta}\right) f(y^*). \end{aligned} \tag{2.10}$$

Suppose now that $c(x') > c(x^*) + f(y^*)$. Since

$$c(x') \leq \rho c(x^*) + \rho\delta f(y^*) - \rho\delta f(y'),$$

from Lemma 2.4, the cost of the solution after the greedy improvement can be bounded by

$$f(y') + f(y^*) + c(x^*) + f(y^*) \ln \frac{(\rho - 1) c(x^*) + \rho\delta f(y^*) - \rho\delta f(y')}{f(y^*)}.$$

This expression, as function of $f(y')$, is maximized for $f(y') = \frac{\rho-1}{\rho\delta} c(x^*)$. Choosing $f(y') = \frac{\rho-1}{\rho\delta} c(x^*)$, we obtain the following upper bound for the total cost

$$\left(1 + \frac{\rho - 1}{\rho\delta}\right) c(x^*) + \left(1 + \ln(\rho\delta)\right) f(y^*). \tag{2.11}$$

Thus, from 2.10 and 2.11 we conclude that the solution obtained by ALG is within $\min_\delta \max\left\{2 - \frac{1}{\rho\delta}, 1 + \frac{\rho-1}{\rho\delta}, 1 + \ln(\rho\delta)\right\}$ the optimum. ■

Choosing as ALG the algorithm designed by Jain and Vazirani and applying Lemma 2.5 with $\rho = 3$, Charikar and Guha obtained that the $UFLP$ can be approximated within a factor of 1.85. Note that the scaling technique is used in order to take advantage of the asymmetry between the performance of the algorithm with respect to facility cost and service cost.

By using the algorithm developed by Jain, Mahdian and Saberi as ALG, Mahdian, Ye and Zhang [MYZ02] obtain the best performance guarantee known for the $UFLP$, namely 1.52. In their case, scaling gives a different advantage: intuitively, facilities opened by the algorithm in [JMS01] with scaled-up facility costs are those that are very economical, because they are weighted more than the service cost in the objective function.

Chapter 3

Approximation algorithms for the metric fault tolerant facility location problem

In the so-called fault tolerant version of the $UFLP$, denoted by $TFLP$, each demand point $j \in D$ requires to be connected to a given number $r_j, r_j \geq 1$ facilities. As in the $UFLP$, one has to decide where to open facilities and how to assign demand points to the required number of facilities, such that the total cost, i.e., the transportaion cost and the opening facility cost is minimized.

In Section 3.1 we introduce the problem and formulate it as an integer program. Then we discuss a linear programming relaxation and its dual. Section 3.2 is devoted to approximation algorithms for the $TFLP$. After reviewing the known results on this aspect, we present in Section 3.2.2 a very simple $4-$approximation algorithm based on linear programming rounding. In Section 3.2.3 we present, for the special case where all connectivity requirements r_j are equal, a primal dual algorithm achivieng the performance guarantee 3. Next, we show how, with the help of scaling and the greedy improvement technique developed by Guha, Meyerson and Munagala in [GMM01], the approximation guarantee of the algorithm presented in this section can be improved to 1.85.

3.1 Preliminaries

As we have seen, the $UFLP$ models the tradeoff of developing resources (facilities) and the utility (reduction in assignment cost) accruing from such. In several applications, as for example caching on a network, fault tolerance is also desirable. The placement of caches should be resistent to failures of nodes and links. The $UFLP$ does not provide any guarantee about the second closest open facility to any demand point. In this chapter we consider the fault tolerant facility location problem, denoted by $TFLP$, in which every demand point j has to be assigned to a number of r_j facilities. More precisely, the problem can be formulated as follows.

Consider a complete bipartite graph $G = (V, E)$ with $V = F \cup D$ and $E = F \times D$. The set F is the set of possible *facility locations* and D is the set of *demand points*. We are given a *service (transportation) cost vector* $c \in R_+^{|E|}$ (i.e., the cost of transporting a unit of demand from i to j is c_{ij}) and an *opening cost vector* $f \in R_+^{|E|}$. (i.e., opening a facility at $i \in F$ incurs a cost $f_i \geq 0$). We assume that c is induced by a metric on V. Every demand point $j \in D$ must be connected to r_j open facilities. r_j is called the *connectivity requirement* of j. The goal is to decide which facilities to open and how to connect the demand points to the required number of facilities such that the total cost (the opening costs plus the connection costs) is minimized.

Remark 3.1 *For simplicity, we will assume that each $j \in D$ demands exactly one unit from each of the r_j facilities to which it gets connected. All the results presented in this chapter can be easily generalized to arbitrary demands. Furthermore, we will assume that $f_i > 0$ for every $i \in F$. This is not a restrictive assumption, since if there is a facility i with no opening cost, we can open it at the end and assign to it all the demand points for which a reassignment to i would mean a decrease in the service cost.*

Although a very natural extension of the $UFLP$, the $TFLP$ was not much studied in the literature. As far as we know, it was introduced by Jain and Vazirani [JV00] and was studied only from the point of view of approximation algorithms. We give a review of the results obtained so far in Section 3.2.

3.1.1 An integer programming formulation

One of the ways of formulating the $TFLP$ as an integer program is the following. Let y_i ($i \in F$) be indicator variables denoting whether facility $i \in F$ is open and x_{ij} ($i \in F, j \in D$) indicator variables denoting whether demand point j is connected to facility i. We let

$$c(x) := \sum_{i \in F, j \in D} c_{ij} x_{ij}$$

and

$$f(y) := \sum_{i\in F} f_i y_i.$$

The $TFLP$ is then equivalent with

$$(\mathbf{TP}_{\text{int}}) \qquad \begin{aligned} &\text{minimize } c(x) + f(y) \\ &\text{subject to} \sum_{i\in F} x_{ij} \geq r_j, \quad \text{for each } j \in D && (3.1) \\ &\qquad x_{ij} \leq y_i, \quad \text{for each } i \in F \text{ and } j \in D && (3.2) \\ &\qquad x_{ij}, y_i \in \{0,1\} \ \text{ for each } i \in F \text{ and } j \in D \end{aligned}$$

Constraints (3.1) ensure that each demand point $j \in D$ is connected to r_j facilities and constraints (3.2) ensure that each of these facilities must be open.

We consider the following LP-relaxation of $(\mathbf{TP}_{\text{int}})$:

$$(\mathbf{TP}) \qquad \begin{aligned} &\text{minimize } c(x) + f(y) \\ &\text{subject to } (3.1), (3.2) \\ &x_{ij} \leq 1, \ \text{for each } i \in F \text{ and } j \in D && (3.3) \\ &x_{ij}, y_i \geq 0, \ \text{for each } i \in F \text{ and } j \in D \end{aligned}$$

Note that constraints (3.3) are necessary to ensure that for each $j \in D$, the r_j facilities to which j is assigned are distinct.

As in the case of the $UFLP$, the dual program and the complementary slackness conditions will play an important role in developing approximation algorithm for the $TFLP$.

Introducing dual variables v_j, t_{ij} and z_i, corresponding to constraints $(3.1), (3.2)$ and (3.3) in $(\mathbf{TP})$, the dual becomes

$$(\mathbf{TD}) \qquad \begin{aligned} &\text{maximize} \sum_{j\in D} r_j v_j - \sum_{i\in F}\sum_{j\in D} z_{ij} && (3.4) \\ &\text{subject to } v_j - t_{ij} - z_{ij} \leq c_{ij}, \ \text{for each } i \in F \text{ and } j \in D && (3.5) \\ &\sum_{j\in D} t_{ij} \leq f_i, \ \text{for each } i \in F && (3.6) \\ &v_j, z_{ij}, t_{ij} \geq 0, \ \text{for each } i \in F \text{ and } j \in D \end{aligned}$$

Let (x^*, y^*) and (v^*, t^*, z^*) be optimal solutions to $(\mathbf{TP})$, respectively $(\mathbf{TD})$. The primal and dual complementary slackness conditions are:

$(TS1)$ For every $i \in F$ and $j \in D$, $x_{ij}^* > 0$ implies that $v_j^* - t_{ij}^* - z_{ij}^* = c_{ij}$.

$(TS2)$ For every $i \in F$, $y_i^* > 0$ implies that $\sum_{j \in D} t_{ij}^* = f_i$.

$(TS3)$ For every $j \in D$, $v_j^* > 0$ implies that $\sum_{i \in F} x_{ij}^* = r_j$

$(TS4)$ For every $i \in F$ and $j \in D$, $t_{ij}^* > 0$ implies that $x_{ij}^* = y_i^*$.

$(TS5)$ For every $i \in F$ and $j \in D$, $z_{ij}^* > 0$ implies that $x_{ij}^* = 1$.

Conditions $(TS1)$ and $(TS2)$ have an intuitive interpretation as well. $(TS1)$ says that if $x_{ij}^* > 0$ for a demand point $j \in D$ and a facility $i \in F$, then serving j from i incurrs a total cost of v_j^*. From condition $(TS2)$ we conclude that if $y_i^* > 0$ for a facility i, then the total amount demand points are willing to contribute for opening i is equal to f_i. Condition $(TS5)$ is a very important, since together with (3.2) it implies that all the facilities i for which there is a demand point j such that $z_{ij}^* > 0$ are opened ($y_i^* = 1$) by an optimal solution to (**TP**). Hence, if we design an approximation algorithm based on rounding a solution of (**TP**), the variables x_{ij}^* and y_i^*, corresponding to an $i \in F$ and $j \in D$ for which $z_{ij}^* > 0$, are already integers.

3.2 Approximation algorithms

There are not many approximation algorithms for the $TFLP$. The first approximation algorithm for this problem was developed by Jain and Vazirani [JV00] and has a performance guarantee of $O(\log R)$, where $R = \max_{j \in D} r_j$ Their algorithm proceeds in R phases. Each phase decreases by one the maximum *residual requirement*, that is, the maximum number of further facilities needed by a demand point. In other words, at the beginning of the p-th phase the maximum residual requirement is p and at the end of the phase is $p-1$. The decrease in the maximum residual requirement is realised by running in every phase a primal-dual algorithm very similar to the one developed for the $UFLP$. The approximation guarantee is obtained by proving that the cost of the solution obtained in the $p-1-$th phase minus the cost of the solution obtained in the $p-$th phase is at most $\frac{3}{p} C_{TOPT}$, where C_{TOPT} is the optimal solution of $(\mathbf{TP}_{int})$.

The first constant approximation algorithm for the $TFLP$ was developed by Guha, Meyerson and Munagala [GMM01], who proposed a 4 approximation algorithm based on filtering and linear programming rounding. If the filtering is done randomly, the performance guarantee can be improved to 3.16. Combining filtering and greedy improvement

, they improve the performance guarantee to 2.47. Basically, the algorithm proposed by Guha *et al.* is constructed using the same ideas as the 3.16 approximation algorithm of Shmoys, Tardos and Aardal [STA97] for the $UFLP$. However, the more general connectivity requirements lead to a a much more complicated analysis as compared to the $UFLP$ case.

Later on, Mahdian *et al.* [MMSV01] proposed a $2-$ approximation algorithm based on dual fitting for the special case of $TFLP$ in which all connectivity requirements are equal.

First we will present a $4-$approximation algorithm based on linear programming rounding. The rounding procedure is much simpler than the one in [GMM01]. Instead of comparing the solution obtained by rounding with a primal filtered solution, we will compare it to the optimal dual solution. However, unlike a filtered primal solution, the optimal dual solution does not allow an improvement of the performance guarantee.

Then we will design a combinatorial approximation algorithm for the special case of $TFLP$ where all the conectivity requirements are equal, i.e., $r_j = r$ for all $j \in D$. The algorithm uses the primal-dual technique and is a generalization of the algorithm developed by Jain and Vazirani for the $UFLP$. Finally, we will show how, by using the scaling and greedy improvement technique proposed by Guha *et al.,* the approximation guarantee of the algorithm presented can be reduced to1.85.

3.2.1 An approximation algorithm based on linear programming rounding.

Let (x^*, y^*) and (v^*, t^*, z^*) be optimal primal respectively dual solutions.

Clearly, we may assume that $\sum\limits_{i\in F} x^*_{ij} = r_j$ for all $j \in D$. We round (x^*, y^*) to a binary solution (x, y) as follows.

Let $O = \{i \in F|\ y^*_i = 1\}$.

We start with $(x, y) := (x^*, y^*)$ and perform the following preprocessing:

```
WHILE $0 < x_{i_0 j} < y_{i_0}$ for some $i_0 \in O, j \in D$ DO
    Set $x_{i_0 j} = 1$
    Decrease some $x_{ij}$ with $x_{ij} < 1$ so as to keep $\sum_{i\in F} x_{ij} = r_j$
```
.

At the end of this preprocessing phase, we have constructed a partial assignment, assigning each $j \in D$ to a set $O_j \subseteq O$ of facilities $(|O_j| \leq r_j)$.

Lemma 3.1 *The partial assignment constructed this way has a total cost of*

$$\sum_{j \in D} \sum_{i \in O_j} c_{ij} + \sum_{i \in O} f_i \leq \sum_{j \in D} |O_j| \, v_j^* - \sum_{i \in F, j \in D} z_{ij}^*.$$

Proof. Note that, by construction, $i \in O_j$ implies $x_{ij}^* > 0$. So, $(TS1)$ yields $c_{ij} = v_j^* - t_{ij}^* - z_{ij}^*$. Furthermore, if $i \in O$ and $t_{ij}^* > 0$, then $x_{ij}^* = 1$ (cf. $(TS4)$) and $i \in O_j$. Finally, we conclude from $(TS5)$ that $z_{ij}^* > 0$ implies $i \in O_j$. Hence we get

$$\sum_{j \in D} \sum_{i \in O_j} c_{ij} = \sum_{j \in D} |O_j| \, v_j^* - \sum_{j \in D} \sum_{i \in O} t_{ij}^* - \sum_{j \in D} \sum_{i \in F} z_{ij}^*,$$

and the claim follows from $(TS2)$.

■

We now concentrate on the remaining properly fractional part of the assignment, defined by (x_{ij}) and (y_i) for $i \in F \backslash O$ and $j \in D$. We claim that we can open certain facilities in $F \backslash O$ at a total opening cost of at most

$$\sum_{i \in F \backslash O} f_i y_i \leq \sum_{i \in F \backslash O} f_i y_i^*$$

and assign each $j \in D$ to $r_j - |O_j|$ open facilities in $F \backslash O$ with assignment cost bounded by $3 \, (r_j - |O_j|) \, v_j^*$. Together with Lemma 3.1 this will then give a solution within 4 times the optimum.

The rounding procedure achieving this is basically the same as the standard procedure in the case $r_j = 1, j \in D$. Intuitively, this is due to the fact that we now deal with assignments x_{ij} with $i \in F \backslash O$. For these we have $z_{ij}^* = 0$, i.e., the constraints $x_{ij} \leq 1$ are not "active".

Redefine $F \leftarrow F \backslash O$ and $r_j \leftarrow r_j - |O_j|$. In what follows,

$$N_j = \{ i \in F | \; x_{ij} > 0 \}$$

always denotes the neigborhood of $j \in D$ with respect to the current x and F. We modify (x, y) as follows. Assume

$$v_1^* \leq ... \leq v_{|D|}^*.$$

In each iteration we assign the currently smallest $j_0 \in D$ with $r_{j_0} > 0$ to the cheapest facility $i_0 \in N_{j_0}$. More precisely, we perform the following steps.

```
Set $y_{i_0} = 1$
Decrease $y$ on $N_{j_0} \setminus \{i_0\}$ so as to keep $\sum_{i \in N_{j_0}} y_i$ invariant
FOR $j \geq j_0$ with $N_j \cap N_{j_0} \neq \emptyset$ DO
        Set $x_{i_0 j} := 1$
        Decrease $x_{ij}$ on $N_j \setminus \{i_0\}$ such that $x_{ij} \leq y_i$ on $N_j$
          $\sum_{i \in F} x_{ij}$ remains unchanged
        Update $r_j \leftarrow r_j - 1$
Update $F \leftarrow F \setminus \{i_0\}$
```

Observe (inductively) that each iteration maintains a feasible fractional assignment $x_{ij} \leq y_i$ with $\sum_{i \in F} x_{ij} = r_j, j \in D$. To analyze the total cost of the final assignment, first note that we certainly do not increase $\sum_{i \in F} f_i y_i$, since i_0 is the cheapest facility in N_{j_0}. So the total opening cost of the constructed solution is bounded by $\sum_{i \in F} f_i y_i^*$.

Secondly, observe that at the beginning of each iteration, $x_{ij} \leq x_{ij}^*$ for $i \in F, j \in D$. In particular, $i_0 \in N_{j_0}$ implies $0 < x_{i_0 j_0} \leq x_{i_0 j_0}^*$. So by $(TS1)$, $c_{i_0 j_0} \leq v_{j_0}^*$.

The same argument yields for $j > j_0$ with, say, $i \in N_j \cap N_{j_0}$ that $c_{ij_0} \leq v_{j_0}^*$ and $c_{ij} \leq v_j^*$. Hence the triangle inequality gives

$$c_{i_0 j} \leq c_{i_0 j_0} + c_{ij_0} + c_{ij} \leq 2v_{j_0}^* + v_j^* \leq 3v_j^*.$$

So each assignment $x_{ij} = 1$ has cost at most $3v_j^*, j \in D$.

We have proved

Theorem 3.1 *Given an optimal primal solution (x^*, y^*) we can find in polynomial time an integer solution (x, y) to ($\boldsymbol{TP}_{int}$) of cost*

$$c(x) + f(y) \leq 3 \sum_{j \in D} v_j^* - 3 \sum_{i \in F, j \in D} z_{ij}^* + \sum_{i \in F} f_i y_i^* \leq 4C_{TLP} \leq 4C_{TOPT}.$$

Remark 3.2 *Obviously, a more efficient variant of the rounding procedure would open the r_{j_0} cheapest facilities in N_{j_0} in each step. The "slow version" above is presented only for notational convenience.*

3.2.2 A primal-dual algorithm for the case $r_j = r$

In this section we assume that $r_j = r$, $r > 1$, holds. If $r = 1$, the problem reduces to the $UFLP$ and we can apply any algorithm described in Chapter 2.

Applying the algorithm described in previous sesction, we obtain a solution with a value within 4−times the optimum. However, for the special case where $r_j = r$, $r > 1$, the algorithm we will present bellow has a better performance guarantee.

The algorithm we propose follows the main lines of a primal-dual algoritm. It constructs, in polynomial time, a feasible dual solution (v, t, z) and a feasible integral primal solution (x, y) that satisfy the complementary slackness conditions $(TS2) - (TS5)$ and the following relaxation of $(TS1)$:

$(TS1')$ $\forall\ i \in F$ and $j \in D$ with $x_{ij} > 0$, $\frac{1}{3}c_{ij} \leq v_j - t_{ij} - z_{ij} \leq c_{ij}$

The cost of such a primal solution (x, y) can be then bounded by

$$\begin{aligned} c(x) + 3f(y) &= \sum_{i \in F, j \in D} c_{ij}x_{ij} + 3\sum_{i \in F} f_i y_i = \sum_{i \in F, j \in D} c_{ij}x_{ij} + 3\sum_{i \in F, j \in D} t_{ij}x_{ij} \\ &= \sum_{j \in D}\sum_{i \in F}(c_{ij} + 3t_{ij})\, x_{ij} \leq 3\sum_{j \in D}\sum_{i \in F}(v_j - z_{ij})\, x_{ij} = 3\sum_{j \in D}(rv_j - \sum_{j \in D}\sum_{i \in F} z_{ij}), \end{aligned}$$

where the second equality is based on $(TS2)$ and $(TS4)$, the first inequality is based on $(TS1')$ and the last equality on $(TS3)$ and $(TS5)$.

The solution (x, y) then satisfies

$$c(x) + 3f(y) \leq 3C_{TLP} \leq 3C_{TOPT}.$$

Hence, an algorithm that constructs in polynomial time a primal feasible solution (x, y) and a dual feasible solution (v, t, z) satisfying $(TS1')$ and $(TS2)-(TS5)$ is a 3−approximation algorithm for the $TFLP$.

Next, we will present some properties of a primal integral feasible solution (x, y) and a dual feasible solution (v, t, z) satisfying $(TS1')$ and $(TS2) - (TS5)$. These properties are easier to be checked during the algorithm than $(TS3)$-$(TS5)$.

Lemma 3.2 *Let (x, y) and (v, t, z) be primal, respectively dual feasible solution. Suppose that (x, y) is integral and that (x, y) and (v, t, z) satisfy $(TS3)$, $(TS4)$ and $(TS5)$. Then every demand point j with $v_j > 0$ verifies the following relations*

$(P1)$ $|\{i \in F|\ z_{ij} > 0\}| \leq r_j$.

$(P2)$ $|\{i \in F|\ y_i = 1$ *and* $t_{ij} + z_{ij} > 0\}| \leq r_j$.

$(P3)$ $|\{i \in F|\ t_{ij} > 0$ *and* $\sum_{j \in D} z_{ij} > 0\}| \leq r_j$.

Proof. From $(TS3)$ follows that $\sum_{i \in F} x_{ij} = r_j$, for every demand point j with $v_j > 0$. Moreover, $(TS5)$ implies that for every facility $i \in F$ with $z_{ij} > 0, x_{ij} = 1$. Hence, there can be at most r_j facilities with $z_{ij} > 0$.

From $(TS4)$ follows that $x_{ij} = y_i = 1$, for every demand point j and facility i with $t_{ij} > 0$ and $y_i = 1$. Together with $(TS3)$ and $(TS5)$, this implies $(P2)$.

In order to prove $(P3)$, note that $\sum_{j \in D} z_{ij} > 0$ for a facility i only if $y_i = 1$. Then $(P3)$ is a direct consequence of $(P2)$.

■

Before describing the algorithm in detail, we review and introduce some notations w.r.t. a dual feasible solution (v, t, z).

A demand point $j \in D$ *reaches* a facility $i \in F$ if

$$c_{ij} \leq v_j.$$

Denote by F_j the set of facilities reached by j.

If $t_{ij} > 0$ we say that j is *willing to contribute* with t_{ij} at opening facility i. A facility i is *fully paid* when

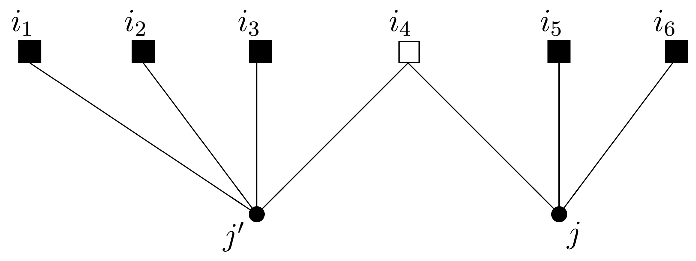
$$\sum_{j \in D} t_{ij} = f_i.$$

We will call a demand point j *saturated* when there are at least r facilities satisfying the following conditions :

- i is open
- $i \in F_{j'}$ for some $j' \in D$ with $v_{j'} \leq v_j$ and $F_{j'} \cap F_j \neq \emptyset$.

A demand point j that reaches r open facilities i is called *directly saturated.* A closed facility i is *blocked* by j if j is directly saturated and $t_{ij} > 0$.

Example 3.1 *In Figure 3.4 we give an example of a saturated demand point. In this example,* $r = 3$, *facilities* $\{i_1, i_2, i_3, i_5, i_6\}$ *are open,* $F_{j'} = \{i_1, i_2, i_3, i_4\}$, $F_j = \{i_4, i_5, i_6\}$. *Since* r *facilities in* $F_{j'}$ *are open, demand point* j' *is directly saturated. If* $v_{j'} \leq v_j$, *then* j *is (indirectly) saturated. Any* 3 *facilities among the open ones satisfy the conditions in the definition of a saturated demand point. Facility* i_4 *is blocked by* j' *if* $t_{i_4 j'} > 0$.



■ are open facilities and □ is not open

Figure 3.1: Example of saturated demant point

As an indirect consequence of the definition, we obtain

Lemma 3.3 *If i is a facility blocked by a directly saturated demand point j', then any demand point j that reaches i and has $v_{j'} \leq v_j$ is saturated.*

The definition of a saturated demand point is justified by the following lemma.

Lemma 3.4 *Let j, j' be two demand points such that $F_j \cap F_{j'} \neq \emptyset$ and $v_{j'} \leq v_j$. For every $i \in F_{j'}$ we then have $c_{ij} \leq 3v_j$.*

Proof. Let $i' \in F_{j'} \cap F_j$. The definition of the sets F_j, respectively $F_{j'}$ implies that $c_{i'j} \leq v_j$, $c_{i'j'} \leq v_{j'}$ and $c_{ij'} \leq v_{j'}$. The triangle inequality thus yields

$$c_{ij} \leq c_{i'j} + c_{i'j'} + c_{ij'} \leq 2v_{j'} + v_j \leq 3v_j.$$

■

Remark 3.3 *Lemma (3.4) is essential for the algorithm. If we connect a demand point j to a facility $i \in F_j$ the service cost can be bounded by v_j. Lemma* (3.4) *tells us that if we would connect j to facilities in $F_{j'}$, with $F_j \cap F_{j'} \neq \emptyset$ and $v_{j'} \leq v_j$, the service cost would increase by at most a factor of* 3. *Hence, a saturated demand point j has at least r open facilities at distance at most $3v_j$.*

The algorithm has a notion of *time*, such that every event can be associated with the time it takes place. The algorithm differs from the one developed by Jain and Vazirani in the fact that the opening of facilities is done simultaneously with the construction of the dual solution.

The algorithm can be described as follows.

Start with $v \equiv t \equiv z \equiv 0$. Initially, all the demand points are non-saturated and no facility is open. Until all the demand points become saturated, for all the not saturated demand points j proceed as follows. Increase v_j uniformly ("with unit speed"). If j does not reach a blocked facility, increase the corresponding (t, z) variables in the following way. For all the not fully paid facilities i that are reached by j, (i.e., facilities for which $c_{ij} < v_j$), start increasing t_{ij} with unit speed, until f_i is fully paid. For all the fully paid facilities i reached by j, start increasing z_{ij} with unit speed. Open all the fully paid and not blocked facilities. When a demand point j becomes saturated, stop increasing v_j, t_{ij}, z_{ij} for every $i \in F$.

In other words, the algorithm proceeds as follows:

```
UNTIL all $j \in D$ are saturated DO
    FOR all the non-saturated $j \in D$ DO
        Increase $v_j$
        FOR all facilities $i \in F$ reached by $j$
            IF $i$ is not fully paid increase $t_{ij}$
            ELSE increase $z_{ij}$.
    Open all fully paid, not blocked facilities $i \in F$
```

Remark 3.4 *Note that at any time T during the algorithm, $T = v_j$ for every non-saturated demand point j. At the end of the algorithm, v_j will indicate the time when demand point j became saturated. After a demand point j becomes directly saturated, no further facility $i \in F_j$ with $t_{ij} > 0$ will be opened, since they are all blocked.*

We summarize the properties of the dual solution (v, t, z) constructed in the following lemma.

Lemma 3.5 *(i) For every demand point j and facility i reached by j, $v_j - t_{ij} - z_{ij} = c_{ij}$. For every facility i that was not reached by j, $t_{ij} = z_{ij} = 0$.*
(ii) (v, t, z) is a feasible dual solution.
(iii) Every facility i with $\sum_{j \in D} z_{ij} > 0$ is opened.

(iv) For every open facility i and demand points j,j' for which $i \in F_{j'}$, $F_{j'} \cap F_j \neq \emptyset$ and $v_{j'} \leq v_j$ at moment v_j, the following relation is true

$$\frac{1}{3}c_{ij} \leq v_j - t_{ij} - z_{ij} \leq c_{ij}. \tag{3.7}$$

Moreover, for every demand point j, there are r open facilities i satisfying 3.7.

Proof. (i) For all the demand points j and facilities i, first we have increased the variable v_j and maintained $t_{ij} = z_{ij} = 0$ until $v_j = c_{ij}$. Then, until i became fully paid, we increased simultaneously t_{ij} and v_j, thus maintaining $v_j - t_{ij} = c_{ij}$ and $z_{ij} = 0$. If after i became fully paid, j was still non-saturated, we increased simultaneously v_j and z_{ij} and kept t_{ij} fixed.

(ii) We have actually proved in (i) that (v,t,z) satisfies (3.5). Since for every facility i we stop increasing the dual variables t_{ij} when $\sum_{j \in D} t_{ij} = f_i$, (v,t,z) satisfies (3.6) as well.
(iii) Let T be the moment when we increased for the first time a variable z_{ij}. At moment T, j was not saturated and i was fully paid, otherwise we would not have increased z_{ij}. Facility i was not blocked as well, otherwise j would have become saturated, cf. Lemma (3.3). This implies that facility i was opened at time T.
(iv) If j reached facility $i, v_j - t_{ij} - z_{ij} = c_{ij}$ and (3.7) holds. If j did not reach i, $t_{ij} = z_{ij} = 0$ and (3.7) follows from Lemma 3.4. The second part of (iv) follows from the fact that when j became saturated, there were r facilities satisfying (3.7).

■

Now we proceed to construct a primal solution (x,y). Clearly, for every facility i,

$$y_i = \begin{cases} 1, & \text{if facility } i \text{ is opened} \\ 0, & \text{otherwise} \end{cases}.$$

From Lemma 3.5 we know that there are r faclities satisfying (3.7). Assign each j to them.

Clearly, x is defined as follows:

$$x_{ij} = \begin{cases} 1, & \text{if } j \text{ is assigned to } i. \\ 0, & \text{otherwise} \end{cases}.$$

Lemma 3.6 (x,y) *is an integer feasible solution for the primal and together with* (v,t,z) *verifies* $(P1)$-$(P3)$.

Proof. Let $j \in D$. Since $f_i > 0$ for every $i \in F$, $v_j > 0$. From Lemma 3.5 (iii) follows that all facilities i with $z_{ij} > 0$ were opened. Moreover, j reached all these facilities. On the other hand, when j became saturated, and all the dual variables corresponding to j stopped increasing, j reached at most r open facilities. Hence, $(P1)$ is satisfied.

If $t_{ij} + z_{ij} > 0$ for a facility i, then j reached i. $(P2)$ follows from the fact that when a demand point becomes directly saturated, $|\{i \in F|\ y_i = 1 \text{ and } t_{ij} + z_{ij} > 0\}| \leq r$. Then, no facility reached by j is opened.

From Lemma 3.5 (iii) follows that all facilities i with $\sum\limits_{j \in D} z_{ij} > 0$ are opened. $(P3)$ is a direct consequence of $(P2)$.

■

Next, we show that the primal and the dual solution constructed satisfy $(TS1')$ and $(TS2) - (TS5)$, which together with the polynomial running time, will imply that the algorithm presented is a 3−approximation algorithm.

Since every demand point j is assigned either to facilities for which (3.7) is true, we conclude that (x, y) and (v, t, z) verify $(TS1')$. $(TS2)$ is implied by the fact that only fully paid facilities were opened. In order to prove $(TS3)$, note that since every facility i has $f_i > 0$, every demand point j will have $v_j > 0$. Moreover, the algorithm assigns each demand point to exactly r open facilities. $(TS4)$ follows from the fact that every demand point j was assigned to all open facilities i to which he was willing to contribute. Finally, since every $j \in D$ was assigned to all facilities $i \in F$ with $z_{ij} > 0$, $(TS5)$ is true as well.

Running time

A special feature of the primal-dual scheme is that it yields algorithms with good running times. Since this is true in particular for the current algorithm, we will provide some implementation details. We will adopt the following notations: $n_d = |D|$ and $n_f = |F|$ and $n = n_d + n_f$.

Sort all the edges by increasing cost. This gives the order and the time when demand points reach facilities. This operation can be done in time $O(n_d n_f \log(n_d n_f))$, or equivalently, in time $O(n^2 \log n)$.

For each facility i, we maintain the number of demand points that are currently willing to contribute towards it, and the *anticipated time*, T_i, at which it would become fully paid if no other event happens in between. Initially, all the $T_i's$ are infinite and the contribution of each demand point to each facility is equal to zero. The $T_i's$ are maintained in a binary heap so that we can update each one and find the current minimum in $O(\log n_f)$ time. Two types of events happen and they lead to the following updates.

• A demand point j reaches a facility i

-If j becomes saturated, for all the facilities that were counting j as contributor we need to decrease the number of contributors and recompute the anticipated time at which it will become fully paid.

-If j does not become saturated, then if facility i is not fully paid, it gets one more demand point willing to contribute towards its cost and we update T_i accordingly. The new T_i can be calculated in constant time. The heap can be updated in $O(\log n_f)$ time.

• A facility i becomes fully paid. For all the demand points that become saturated we will execute the first case of the previous event, i.e., update the not fully paid facilities towards which they were contributing.

Note that every edge (i, j) is considered at most twice. First when j reaches i, and then when demand point j becomes saturated. For each consideration of this edge, we will do $O(\log n_f)$ work.

Hence, we have proved that

Theorem 3.2 *The algorithm presented above constructs in $O(n^2 \log n)$ a solution (x, y) of cost $c(x) + 3f(y) \leq 3C_{TLP} \leq 3C_{TOPT}$.*

Remark 3.5 *Recall that for the case $r = 1$ the gap between C_{TOPT} and the optimal value of* (**TD**) *was of $3 - \varepsilon$. Naturally, the gap will be at least $3 - \varepsilon$ for $r > 1$. Hence, a performance guarantee of 3 is the best performance we can hope for an algorithm that uses the dual program as a lowerbound.*

Greedy improvement and scaling

In [GMM01], Guha Meyerson and Munagala show that the greedy improvement technique can also be used in order to improve the performance guarantee of approximation algorithms developed for $TFLP$. They have proved that all the results presented for $UFLP$ in Chapter 1, Section 2.2 are valid for $TFLP$ as well.

Hence, if ALG is an algorithm that obtaines a feasible solution (x, y) to ($\mathbf{TP}_{\text{int}}$) such that $c(x) + \rho f(y) \leq \rho C_{TOPT},$, by applying the following steps

- Scale the facility costs by a factor of δ
- Run for the scaled instance the algorithm ALG
- Scale back the solution and apply greedy improvement

we obtain a solution with a value at most $\min_\delta \max \left\{2 - \frac{1}{\rho\delta}, 1 + \frac{\rho-1}{\rho\delta}, 1 + \ln(\rho\delta)\right\}$ the optimal value.

Suppose we choose as ALG the algorithm for $TFLP$ presented in the previous subsection. We have proved there that it returns a feasible solution (x, y) to ($\mathbf{TP}_{\text{int}}$) such that $c(x) + 3f(y) \leq 3C_{TOPT}$. The results of Guha, Meyerson and Munagala imply that scaling and greedy improve the performance guarantee of ALG to 1.85.

Therefore,

Lemma 3.7 *The special case of $TFLP$ with $r_j = r$ for every $j \in D$ can be approximated in polynomial time within* 1.85 *times the optimum.*

3.3 Conclusions and further research

In this chapter we have presented a simple $4-$approximation algorithm based on linear programming rounding for the $TFLP$ and a $1.85-$ approximation algorithm for the special case where $r_j = r$, for all $j \in D$. The second algorithm was based on the primal-dual technique. A very interesting open problem for $TFLP$ is the design of combinatorial aproximation algorithms for the case with arbitrary r_j's. One idea would be to extend the algorithm presented for the case with equal connectivity requirements. However, it seems that for the general case, it is much more difficult to construct in polynomial time a primal solution that simultaneously satisfies conditions $(P1) - (P3)$.

Chapter 4

Approximation algorithms for the metric multilevel facility location problem

In the metric multilevel extension of the uncapacitated facility location problem, denoted by $MFLP$, there are k types of facilities to be built: one type of depots and $(k-1)$ types of transit stations. Each unit of demand must be shipped from a depot through transit stations of type $k-1, ..., 1$ to the demand points. One has to decide where to open facilities and how to assign demand points to paths along open facilities such that the total cost, i.e., the transportation cost and the opening facilities cost is minimized. We assume that the service costs are induced by a metric.

In Section 4.1 we describe the problem and formulate it as an integer program. Then we introduce the linear programming relaxation we are going to use. In Section 4.2 we review the main results on approximation algorithms for the $MFLP$, presenting in more detail the $3-$approximation algorithm developed by Aardal, Chudak and Shmoys [ACS99]. Their algorithm is based on linear programming rounding, and hence has a large running time. As an alternative, in Section 4.3 we propose a primal-dual algorithm that has a performance guarantee equal to 6. In Section 4.4 we extend the algorithm for solving a capacitated version of the problem, in which each facility i can be opened multiple times and can serve at most u_i demand points. Finally, we prove that the primal-dual algorithm proposed works also for the case in which outliers are allowed, i.e, it is allowed to let some demand points remain unconnected, at the price of a penalty. We end this Chapter with Section 4.5, in which we present our conclusions and some ideas for further research.

4.1 Preliminaries

Imagine the following scenario. Someone wants to open a new furniture factory in the regio. After being produced, the furniture is stored in depots and then sold in the factory's own warehouses. The prices of renting depots and of opening warehouses in different cities in the regio are known, as well as the average demand of furniture in each of the cities. One has to decide where to open depots and warehouses such that a certain fraction of demand is satisfied and the total cost (the cost of opening and renting facilities and the transportation cost of the furniture from factory to the client, via depots and warehouses), is minimized.

This is a typical problem that can be modeled as a multilevel facility location problem. The $MFLP$ generalizes the uncapacitated facility location problem. Traditionally it has been applied to location of warehouses and distribution centers in a hierarchical goods distribution network or supply chain (see Aardal *et al.* [Aa92]). Lately, it was used in access network design problems (see Andrews and Zhang [AZ98], Meyerson, Munagala and Plotkin [MMP00]).

Formally, the problem can be described as follows.

Consider a graph $G = (V, E)$ with $V = V_0 \cup ... \cup V_k$ and $E = \bigcup_{l=0}^{k} (V_{l-1} \times V_l)$. The set $D = V_0$ is the set of *demand points* and $F = V_1 \cup ... \cup V_k$ is the set of possible *facility locations* (at *level* $1, ..., k$). We are given a *service cost vector* $c \in R_+^{|E|}$ (the transportation cost of every unit of demand along an edge (i_{l-1}, i_l) is $c_{i_{l-1}i_l}$) and an *opening cost vector* $f \in R_+^{|F|}$ (i.e., opening a facility at $i \in F$ incurs a cost $f_i \geq 0$). We assume that c is induced by a metric on V and that there are no edges of cost 0.

Denote by P the set of paths of length $k-1$ joining some node in V_1 to some node in V_k. If $j \in D$ and $p = (v_1, ..., v_k) \in P$, we let jp denote the path $(j, v_1, ..., v_k)$. As usual, $c(p)$ and $c(jp)$ denote the length of p respectively jp (with respect to c).

The corresponding $MFLP$ can now be stated as follows. Determine for each $j \in D$ a path $p_j \in P$ (along "open facilities") so as to minimize

$$\sum_{j \in D} c(jp_j) + f(\bigcup_{j \in D} p_j).$$

Remark 4.1 *In this setting we assume that each $j \in D$ has a demand of one unit to be shipped along p_j. The results presented in the next sections easily extend to arbitrary positive demands.*

4.1.1 An integer programming formulation

To derive an integer programming formulation of the $MFLP$ problem, we introduce the $(0,1)$ variables y_i $(i \in F)$ to indicate whether $i \in F$ is open and the $0-1$ variables x_{jp} $(j \in D,\, p \in P)$ to indicate whether j is served along p.

We let

$$c(x) := \sum_{p \in P} \sum_{j \in D} c_{jp} x_{jp}$$

and

$$f(y) := \sum_{i \in F} f_i y_i.$$

The $MFLP$ is now equivalent to

$$
\begin{array}{lllr}
(\mathbf{MP}_{\text{int}}) & \text{minimize } c(x) + f(y) & & \\
& \text{subject to} \displaystyle\sum_{p \in P} x_{jp} \geq 1, & \text{for each } j \in D & (4.1) \\
& \displaystyle\sum_{p : p \ni i} x_{jp} \leq y_i, & \text{for each } i \in F,\ j \in D & (4.2) \\
& x_{jp} \in \{0,1\}, & \text{for each } p \in P,\ j \in D & \\
& y_i \in \{0,1\}, & \text{for each } i \in F. &
\end{array}
$$

Constraints (4.1) ensure that each j gets connected via some path and constraints (4.2) ensure that the paths only use open facilities.

Denote by C_{MOPT} the optimal value of ($\mathbf{MP}_{\text{int}}$).

The $LP-$relaxation of ($\mathbf{MP}_{\text{int}}$) is given by

$$
\begin{array}{ll}
(\mathbf{MP}) & \text{minimize } c(x) + f(y) \\
& \text{subject to } (1), (2) \\
& x_{jp} \geq 0 \\
& y_i \geq 0.
\end{array}
$$

Note that for any optimal solution (x^*, y^*) to $(\mathbf{MP})$ $\sum_{p \in P} x^*_{jp} = 1$, $x^*_{jp} \leq 1$ and $y^*_i \leq 1$.

Denote by C_{MLP} the optimum value to (**MP**). Clearly, $C_{MLP} \leq C_{MOPT}$.

Introducing dual variables v_j and t_{ij} corresponding to constraints (4.1) and (4.2) in (**MP**), the dual becomes

$$
\begin{aligned}
(\mathbf{MD}) \qquad & \text{maximize} \sum_{j \in D} v_j \\
& \text{subject to } v_j - \sum_{i \in p} t_{ij} \leq c(jp), \text{ for each } p \in P,\, j \in D && (4.3) \\
& \sum_{j \in D} t_{ij} \leq f_i, \quad \text{for each } i \in F && (4.4) \\
& v_j \geq 0, \quad \text{for each } j \in D \\
& t_{ij} \geq 0, \quad \text{for each } i \in F,\, j \in D.
\end{aligned}
$$

Intuitively, the dual variables v_j indicate how much $j \in D$ is willing to pay for getting connected. The value of t_{ij} indicates how much $j \in D$ is willing to contribute to the opening cost f_i (if he would be connected along a path through i).

As in the algorithms described in Chapter 2 for the one level facility location problem, the complementary slackness conditions will play an important role in designing algorithms for the multilevel facility location problem.

Let (x^*, y^*) and (v^*, t^*) be optimal solutions to (**MP**) and (**MD**). The primal and dual complementary slackness conditions for the $MFLP$ are:

$(MS1)$ $\forall\ j \in D$ and $p \in P$, $x^*_{jp} > 0$ implies $v^*_j - \sum_{i \in p} t^*_{ij} = c(jp)$

$(MS2)$ $\forall i \in F$, $y^*_i > 0$ implies $\sum_{j \in D} t^*_{ij} = f_i$

$(MS3)$ $\forall\ j \in D$, $v^*_j > 0$ implies $\sum_{p \in P} x^*_{jp} = 1$

$(MS4)$ $\forall i \in F$ and $\forall\ j \in D$ $t^*_{ij} > 0$ implies $\sum_{p: i \in p} x^*_{jp} = y^*_i$.

As the dual variables, the primal complementary slackness conditions have an intuitive interpretation as well. Namely, $(MS1)$ says that if $x^*_{jp} > 0$ for a demand point j and a path $p \in P$, then for j, the service cost along path p plus the contribution to facilities along p is v^*_j. From $(MS2)$ follows that, if $y^*_i > 0$ for a facility i, the total amount the demand points are willing to contribute for opening i is equal to f_i. The intuitive meaning of $(MS1)$ and $(MS2)$ will be very helpful in the design of a combinatorial algorithm for the $MFLP$.

4.2 Approximation algorithms

The $MFLP$ was not much studied with respect to approximation algorithms.

In [STA97] Shmoys, Tardos & Aardal extended their algorithm for one level to two levels of facilities, obtaining the first constant approximation algorithm for the two level facility location problem. Their algorithm has a performance guarantee equal to 3.16. Although the algorithm developed by Shmoys, Tardos and Aardal was stated for two levels of facilities, it can easily be extended to more levels without changing the performance guarantee.

Aardal, Chudak and Shmoys [ACS99] proposed an algorithm with a 3-performance guarantee for k levels. This is currently the best performance guarantee obtained for this problem. A drawback of their algorithm is that it requires the optimal solution of a large linear program and hence has a large running time.

The first combinatorial algorithm for the multilevel facility location problem was developed by Meyerson, Munagala and Plotkin [MMP00], and finds a solution within $O(\log |D|)$ times the optimum, where D is the set of demand points.

Later on, Bumb and Kern [BK01] proved that a primal-dual algorithm, similar to the one developed by Jain and Vazirani [JV01] for the $UFLP$ achieves a $6-$performance guarantee for the $MFLP$.

In the next sections we will present the main ideas used to design approximation algorithms for the $MFLP$.

4.2.1 An approximation algorithm based on linear programming rounding

The algorithm developed by Aardal, Chudak and Shmoys is very similar to the one developed by Chudak and Shmoys [C98a], [CS98] for one level of facilities. Again, the main idea is to round an optimal solution to (**MP**) to an integer solution of ($\mathbf{MP}_{\text{int}}$) by clustering the demand points.

Before describing the algorithm in more detail we introduce some notations. Let (x^*, y^*), respectively (v^*, t^*) be optimal solutions to (**MP**), respectively (**MD**).

We say that a path p *fractionally serves* a demand point j if $x^*_{jp} \geq 0$. The *neighborhood* of a demand point j is the set N_j of facilities included in paths that fractionally serve j.

Clearly, $\sum_{p\subseteq N_j} x^*_{jp} = 1$. Furthermore, $(MS1)$ implies that every facility $i \in N_j$ is at distance at most v^*_j from j.

Each cluster will be constructed around a cluster center. Denote by $\mathcal{C}$ the set of cluster centers. We choose the centers and construct the clusters as follows.

Initially $\mathcal{S}$ $=D$ and $\mathcal{C} = \emptyset$. Choose the demand point j in $\mathcal{S}$ which minimizes $v^*_j + \sum_{p\in P} c\,(jp)\,x^*_{jp}$. If N_j intersects N_{j_0} for some cluster centered at j_0 then add j to the cluster centered in j_0. Otherwise declare j a cluster center (add j to $\mathcal{C}$). Remove j from S. Repeat the procedure until $\mathcal{S} = \emptyset$.

Since $\sum_{p\subseteq N_j} x^*_{jp} = 1$ and $x^*_{jp} \geq 0$, x^*_{jp}'s define a probability distribution on the paths in N_j.

In every cluster centered at some j choose a path p with probability x^*_{jp} and open all the facilities on it. Assign all the demand points in this cluster to the chosen path.

```
$\mathcal{S}$ $=D$ and $\mathcal{C} = \emptyset$
UNTIL $\mathcal{S}$ $=\emptyset$ DO
    Choose $j \in \mathcal{S}$ with minimum $v^*_j + \sum_{p\in P} c\,(jp)\,x^*_{jp}$ value
    IF $N_j \cap N_{j_0} \neq \emptyset$ for some $j_0 \in \mathcal{C}$ THEN add $j$ to the cluster centered at $j_0$
    ELSE add $j$ to $\mathcal{C}$
    Set $\mathcal{S} := \mathcal{S}\backslash\,\{j\}$
FOR $j \in \mathcal{C}$
    Choose a path $p \in P$ with probability $x^*_{jp}$
    Open all the facilities on $p$
    Assign all the demand points in this cluster to $p$
```

Note that every demand point j is assigned by the algorithm to a cluster centered in some demand point j_0 with $v^*_{j_0} + \sum_{p\in P} c_{j_0p}x^*_{j_0p} \leq v^*_j + \sum_{p\in P} c\,(jp)\,x^*_{jp}$.

Since different centers have disjoint set of clients, the probability that a facility i is opened can be bounded by

$$\Pr\,(i \text{ is opened}) \leq \sum_{p:i\in p} x^*_{jp} \leq y^*_i,$$

where the second inequality follows from (4.2).

Hence, the expected cost of opening facilities can be bounded as follows:

$$E\,(\text{cost of opening facilities}) = \sum_{i\in F} f_i \Pr\,(i \text{ is opened}) \leq \sum_{i\in F} f_i y^*_i.$$

If demand point j is the center of a cluster, then it will be assigned to a path contained in its neighborhood. Thus,

$$E\left(\text{service cost of } j\right)=\sum_{p\subseteq N_j} c\left(jp\right)\Pr\left(p \text{ was chosen}\right)=\sum_{p\subseteq N_j} c\left(jp\right)x^*_{jp}=\sum_{p\in P} c\left(jp\right)x^*_{jp},$$

since $x^*_{jp}=0$ for $p\nsubseteq N_j$.

Suppose j is not a cluster center and let j_0 be the center of the cluster to which j belongs. The expected service cost of j is

$$E\left(\text{service cost of } j\right)=\sum_{p\subseteq N_{j_0}} c\left(jp\right)\Pr\left(p \text{ was chosen}\right)=\sum_{p\subseteq N_{j_0}} c\left(jp\right)x^*_{j_0p}.$$

From the construction of a cluster, there exists an $i\in N_j\cap N_{j_0}$. From the triangle inequality follows that:

$$c\left(jp\right)\leq c_{ji}+c_{j_0i}+c_{j_0p},$$

where c_{ji} and c_{j_0i} are the distances between j and i, respectively j_0 and i. Since $i\in N_j$, we have $c_{ji}\leq v^*_j$. Similarly, we have $c_{j_0i}\leq v^*_{j_0}$.

Remembering that $\sum\limits_{p\subseteq N_{j_0}} x^*_{j_0p}=1$ and that $v^*_{j_0}+\sum\limits_{p\in P} c_{j_0p}x^*_{j_0p}\leq v^*_j+\sum\limits_{p\in P} c\left(jp\right)x^*_{jp}$, the expected service cost of j can be bounded by

$$E\left(\text{service cost of } j\right)\leq\sum_{p\subseteq N_{j_0}}\left(v^*_j+v^*_{j_0}+c_{j_0p}\right)x^*_{j_0p}\leq 2v^*_j+\sum_{p\in P} c\left(jp\right)x^*_{jp}.$$

Hence, the total cost is at most

$$\begin{aligned}E\left(\text{total cost}\right)&\leq E\left(\text{cost of opening facilities}\right)+\sum_{j\in\mathcal{C}} E\left(\text{service cost of } j\right)+\\&+\sum_{j\notin\mathcal{C}} E\left(\text{service cost of } j\right)\\&\leq\sum_{i\in F} f_iy^*_i+\sum_{j\in\mathcal{C}}\sum_{p\in P} c\left(jp\right)x^*_{jp}+\sum_{j\notin\mathcal{C}}\left(2v^*_j+\sum_{p\in P} c\left(jp\right)x^*_{jp}\right)\\&\leq 3C_{MLP}.\end{aligned}$$

Since $C_{MLP}\leq C_{MOPT}$ the expected cost of the solution obtained by the algorithm is within 3 times the optimum cost. The algorithm can be derandomized by the method of conditional expectations, resulting in a deterministic $3-$approximation algorithm (for details see [ACS99]).

The question that arises is why the algorithm above has such a large performance guarantee in comparison to the very similar algorithm of Chudak and Shmoys for one level of facilities. The problem in the case of more levels is that it is difficult to open facilities such that the following two conditions are satisfied:

• for every demand point j, there is a large probability that all facilities on some path $p \subseteq N_j$ are open

• the expected cost of opening facilities can be bounded by ρC_{MLP}, where ρ is a constant.

In the one level case both conditions are satisfied if we open the facilities i that are in the neighborhood of a center with probability x^*_{ij} and the facilities i that are not in the neighborhood of any center with probability y^*_i (see [C98a]). In the $k-$level case, the first condition may be violated.

4.2.2 A primal-dual algorithm

In this section we present a 6-approximation algorithm for the $MFLP$ (cf. Bumb and Kern [BK01]). The algorithm is based on the primal-dual technique and extends the algorithm of Jain and Vazirani ([JV01]) for the case where there is only one level of facilities to the case where there are more levels.

The aim is to construct a primal feasible $(0,1)$ solution (x,y) and a feasible dual solution (v,t) such that $(MS2)$, $(MS3)$, $(MS4)$ and $(MS1')$ are satisfied, where $(MS1')$ is the following relaxation of $(MS1)$:

$(MS1')$ $\forall\ j \in D$ and $p \in P$, $x_{jp} > 0$ implies that $\frac{1}{5}c\,(jp) \leq v_j - \sum\limits_{i\in p} t_{ij} \leq c\,(jp)$ and $\sum\limits_{i\in p} t_{ij} \leq v_j$.

Suppose that we have constructed in polynomial time (x,y) and (v,t) with the above properties. Then the cost of (x,y) can be bounded as follows. The cost of opening facilities is at most

$$f\,(y) = \sum_{i\in F}\sum_{j\in D} t_{ij}(\sum_{p\ni i} x_{jp}) = \sum_{j\in D}\sum_{p\in P}(\sum_{i\in p} t_{ij})x_{jp} \leq \sum_{j\in D} v_j \sum_{p\in P} x_{jp} = \sum_{j\in D} v_j,$$

where the first equality is based on $(MS2)$ and $(MS4)$, the third inequality on $(MS1')$

and the fourth on $(MS3)$. From $(MS1')$ and $(MS3)$ we conclude

$$c(x) \leq \sum_{j\in D}\sum_{p\in P} c(jp)\, x_{jp} \leq 5\sum_{j\in D} v_j.$$

Hence, the total cost of (x, y) can be bounded by

$$c(x) + f(y) \leq 6\sum_{j\in D} v_j \leq 6C_{MLP} \leq 6C_{MOPT}.$$

This proves that the algorithm is a $6-$approximation algorithm.

Next we will describe how to construct the solutions (x, y) and (v, t) satisfying $(MS1')$ and $(MS2) - (MS4)$.

First we construct the dual solution (v, t). To this end, we introduce the following notation w.r.t. an arbitrary feasible solution (v, t) of (**MD**):

A facility $i \in F$ is *fully paid* when

$$\sum_{j\in D} t_{ij} = f_i. \tag{4.5}$$

A demand point $j \in D$ *reaches* $i_l \in V_l$ if, for some path $p = (i_1, ..., i_l)$ from V_1 to i_l, all facilities $i_1, ..., i_{l-1}$ are *fully paid* and

$$v_j = c(jp) + \sum_{i\in p} t_{ij}. \tag{4.6}$$

If, in addition, also i_l is fully paid, we say that j *leaves* i_l or, in case $l = k$, that j gets *connected* (along p to $i_k \in V_k$).

We start with $v \equiv t \equiv 0$ and increase all v_j uniformly ("at unit speed"). When some $j \in D$ reaches a not fully paid node $i \in F$, we start increasing t_{ij} with unit speed, until f_i is fully paid and j leaves i. We stop increasing v_j when j gets connected. The algorithm has the proprty that at any time T the dual variables v_j that are still being raised are all equal to T. More precisely, we proceed as described below.

```
UNTIL all $j \in D$ are connected DO
    • Increase $v_j$ for all $j \in D$ not yet connected
    • Increase $t_{ij}$ for all $i \in F$, $j \in D$ satisfying $(i) - (iii)$ :
          $(i)$ $j$ has reached $i$
          $(ii)$ $j$ is not yet connected
          $(iii)$ $i$ is not yet fully paid .
```

Let (v,t) denote the final dual solution. Before constructing a corresponding primal solution (x,y), let us state a few simple facts about (v,t).

For each fully paid facility $i \in V_l$, $l \geq 1$, denote by T_i the time when facility i became fully paid. *The predecessor* of i will be the facility on level $l-1$ via which i was reached for the first time by a demand point, i.e.,

$$Pred\,(i) = \left\{ i' \in V_{l-1} | \; i' \text{ is fully paid and } T_{i'} + c_{i'i} = \min_{\substack{i'' \in V_{l-1} \\ i'' \text{ fully paid}}} (T_{i''} + c_{i''i}) \right\}.$$

(ties are broken arbitrarily)

The predecessor of a fully paid facility $i \in V_1$ will be its closest demand point. We define the time $T_{Pred(i)} = 0$ for $i \in V_1$.

For all fully paid facilities i on the $k-th$ level, denote by $j_i p_i = (i_1, ... i_k)$ the path through the following points:

- $i_k = i$
- $i_l = Pred(i_{l+1})$, for each $1 \leq l \leq k-1$
- $j_i = Pred(i_1)$.

We will call the *set of clients* associated to i the set of demand points *contributing* to p_i i.e.,

$$D_i = \{ j \in D \mid t_{i'j} > 0 \text{ for some } i' \in p_i \}.$$

Note that for $l = 1$ the set of clients associated to a fully paid facility is exactly the set of clients introduced in the algorithm of Jain and Vazirani.

Since each $j \in D$ gets connected we may fix for each $j \in D$ a connecting path $\widetilde{p_j} \in P$ of fully paid facilities (ties are broken arbitrarily).

Lemma 4.1 *(i)* $c(j\widetilde{p_j}) \leq v_j$ *for all* $j \in D$

(ii) For all $j \in D$ *and* $i \in V_k$ *fully paid such that* $i \in \widetilde{p_j}$, *either* $v_j = T_i$ *or* $v_j > T_i$ *and* $t_{ij} = 0$.

(iii) For all fully paid facilities $i \in V_k$ *and corresponding paths* $p_i = (i_1, ..., i_k)$, *the following inequalities hold*

$$T_{i_1} \leq ... \leq T_{i_k}.$$

(iv) Let $i \in V_k$ be a fully paid facility and $p_i = (i_1, ..., i_k)$ its associated path. For all $j \in D_i$ there is a path p from V_1 to i such that

$$c(jp) + \sum_{s=1}^{k} t_{i_s j} \leq T_i.$$

In particular, $c(j_i p_i) + \sum_{s=1}^{k} t_{i_s j_i} \leq T_i$.

(v) If i, i' are two fully paid facilities in V_k, with intersecting sets of clients then for each $j' \in D$, such that $i' \in \widetilde{p_{j'}}$, $c_{j_i j'} \leq 4 \max\{T_i, v_{j'}\}$.

(vi) $\sum_{i' \in p_i} t_{i'j} \leq v_j$ for all $j \in D$.

Proof. The first claim is straightforward from (4.6) and the definition of $\widetilde{p_j}$.

The second claim is based on the observation that at time T all the $v-$values that can be increased are equal to T and that the final $v-$values reflect the times when the demand points get connected. There are two possibilities that a fully paid facility $i \in V_k$ is on the connecting path of a demand point j. One is that j reached i before T_i and got connected when i became fully paid. In this case $v_j = T_i$. The other possibility is that j reached i after i was fully paid, which means that $t_{ij} = 0$ and $v_j > T_i$.

The definition of a predecessor implies that for each fully paid $i \in F$ and each $j \in D$ with $t_{ij} > 0$,

$$c_{Pred(i)i} + T_{Pred(i)} + t_{ij} \leq T_i. \tag{4.7}$$

The third claim follows immediately .

The first part of the forth claim follows from the definition of a predecessor.

For the second part, let $j \in D_i$ and let l be the lowest level with the property that $t_{i_l j} > 0$.

By adding the inequalities (4.7) for $i_{l+1}, ..., i_{k-1}$, one obtains

$$\sum_{s=l}^{k-1} (c_{i_s i_{s+1}} + t_{i_{s+1} j}) + T_{i_l} \leq T_{i_k}. \tag{4.8}$$

Since $t_{i_l j} > 0$, there is a path p_1 from V_1 to i_l such that

$$c(jp_1) + t_{i_l j} \leq T_{i_l}. \tag{4.9}$$

Combining (4.8) and (4.9), one obtains (iv) for the path p which goes from V_1 to i_l along p_1 and then through $i_{l+1}, ..., i_k$.

For proving (v), let $j \in D_i \cap D_{i'}$. Since $j \in D_i$, by (iv), there exists a path q from V_1 to i such that $c\,(jq) \leq T_i$.

Similarly, there is a path q' from V_1 to i' such that $c\,(jq') \leq T_{i'}$.

Using the triangle inequality and (ii), we obtain

$$\begin{aligned} c_{j_i j'} &\leq c(j_i p_i) + c(jq) + c(jq') + c(j'\widetilde{p_{j'}}) \\ &\leq 2T_i + T_{i'} + v_{j'} \\ &\leq 2T_i + 2v_{j'} \\ &\leq 4 \max\{T_i, v_{j'}\}. \end{aligned}$$

Finally, for proving the statement in the last claim, it is enough to show that no demand point j could increase simultaneously two values $t_{i_l j}, t_{i_s j}$, for $i_l \neq i_s$ and $i_l, i_s \in p_i$. This follows from the definition of p_i, which implies that whenever a demand point reaches a facility on p_i, the predecessor of that facility and consequently, all the facilities of p_i situated on inferior levels should already have been paid. ■

We now describe how to construct a corresponding primal solution (x, y).

The solution (x, y) that opens all fully paid facilities and then assigns each demand point to the path with the cheapest service cost does not satisfy relation $(MS4)$. The reason is that a demand point j is allowed to "contribute" towards opening facilities on several paths, i.e., $t_{ij} > 0$ for i on different paths. Hence, we should search for a solution in which a demand point j has $t_{ij} > 0$ only for facilities situated along one path.

Suppose there are r fully paid facilities on the last level. Order them according to nondecreasing $T-$values, say

$$T_1 \leq ... \leq T_r.$$

Construct greedily a set $\mathcal{C} \subseteq V_k$ of *centers* which have pairwise disjoint set of clients and assign each $j \in D$ to some center $i_0 \in \mathcal{C}$ as follows:

```
INITIALIZE $\mathcal{C} = \emptyset$
FOR $i = 1, ...r$ DO
    IF $D_i \cap D_{i_0} \neq \emptyset$ for some $i_0 \in \mathcal{C}, i_0 \leq i$ THEN assign to $p_{i_0}$ all demand
    nodes $j \in D$ with $i \in \widetilde{p_j}$
    ELSE $\mathcal{C} = \mathcal{C} \cup \{i\}$ and assign to $p_i$ all the demand nodes $j \in D$
    with the property that $i \in \widetilde{p_j}$
```

The paths p_i $(i \in \mathcal{C})$ are called *central paths.*

Remark 4.2 *Note that each demand point j is assigned to one center. Furthermore, by construction of $\mathcal{C}$, j "contributes" to at most one central path (not necessarily the one to which it is assigned).*

The primal solution (x, y) is obtained by connecting all demand nodes along their corresponding central paths:

$$x_{jp} := \begin{cases} 1 \text{ if } p \text{ is the central path with minimal } c(jp) \text{ value} \\ 0 \text{ otherwise} \end{cases}$$

and

$$y_i := \begin{cases} 1 \text{ if } i \text{ is on a central path} \\ 0 \text{ otherwise} \end{cases}.$$

In order to prove that the algorithm above is a 6-approximation algorithm it is enough to prove that (x, y) and (v, t) satisfy $(MS1')$ and $(MS2) - (MS4)$.

It is straightforward to show that (x, y) satisfy $(MS2) - (MS4)$.

If $j \in D$ is assigned to p_{i_0} then $T_{i_0} \leq T_i$,where $\{i\} = \widetilde{p_j} \cap V_k$. Due to Lemma 4.1 (ii) and (v), we get $T_{i_0} \leq v_j$ and

$$c\left(jp_{i_0}\right) \leq c_{j_{i_0}j} + c\left(j_{i_0}p_{i_0}\right) \leq 4v_j + T_{i_0} \leq 5v_j.$$

From the feasibility of (v, t) follows that $v_j - \sum\limits_{i \in p_{i_0}} t_{ij} \leq c\left(jp_{i_0}\right)$.

The cost of opening facilities along a central path p_{i_0} can be bounded with the help of Lemma 4.1 (vi)

$$\sum_{i \in p_{i_0}} t_{ij} \leq v_j.$$

Hence, (x, y) and (v, t) satisfy $(MS1')$.

We have proved

Theorem 4.1 *The above primal solution (x, y) satisfies*

$$c(x) + f(y) \leq 6 \sum_{j \in D} v_j \leq 6C_{LP} \leq 6C_{OPT}.$$

Running time

In the following we use the notations: $n_d = |V_0|$, $n_f = \sum_{l=1}^{k} |V_l|$ and $n = n_d + n_f$.

For each facility $i_l \in V_l$, $l \geq 1$ we maintain the number of the demand points that are currently willing to contribute towards opening it, the *anticipated time* T_{i_l} at which it would be fully paid, if no other event happens on the way. For every demand point j we maintain the *anticipated time* $T_{i_l j}$ when demand point j will reach for the first time facility i_l. Initially, all T_i's are infinite and each facility has zero demand points willing to contribute towards opening it. The T_i's are maintained in a binary heap so we can update each one and find the minimum in $O\left(\log\left(n_f + n_f n_d\right)\right) = O\left(\log\left(n\right)\right)$ time.

There are three types of events that can happen and they lead to the following updates:

• A demand point j reaches a facility $i_l \in V_l$.

If i_l is not fully paid, it gets one more demand point willing to contribute towards opening it. The anticipated time for facility i_l to be fully paid can be calculated in constant time. For every demand point j' that reached i_l we update, if necessary, the anticipated time when j' reaches facilities in V_{l+1}.

• A demand point j reaches a fully paid facility $i_k \in V_k$.

In this case j is declared connected and v_j is not raised anymore. For each not fully paid facility i that was counting j as a contributor we recompute the anticipated time at which it gets fully paid.

• A facility $i_k \in V_k$ becomes fully paid. Then all demand points j that reached i_k are declared connected. For all these demand points we do the operations described for the second type of events.

Note that every pair $(j, i_l,) \in D \times V_l$, $1 \leq l \leq k$ is considered only in the following situations: when j reaches i_l, when another demand point reaches a not fully paid facility $i_{l-1} \in V_{l-1}$ reached by j and when j becomes connected. In all cases we do $O\left(\log n\right)$ work. Hence, the total work is $O\left(n^4 \log n\right)$.

We have proved the following theorem

Theorem 4.2 *The primal-dual algorithm has a* $O\left(n^4 \log n\right)$ *running time.*

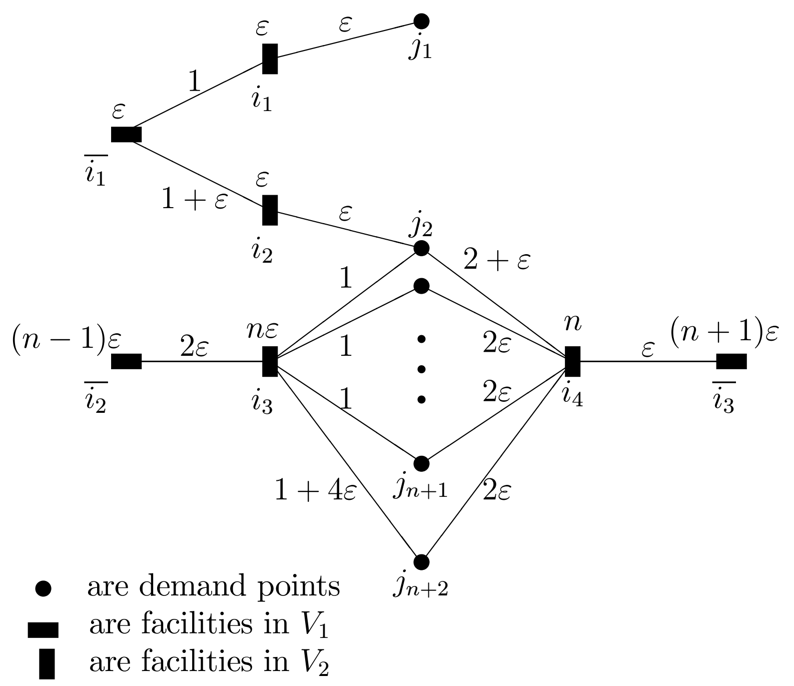


Figure 4.1: A tight example of $MFLP$

An asymptotically tight example

Consider the following family of examples. There are $n+2$ demand points $D = \{j_1, ...j_{n+2}\}$ and two levels of facilities, V_1 and V_2. On the first level there are 4 facilities, say i_s, where $1 \leq s \leq 4$ and on the second there are only 3 facilities, $\overline{i_s}$, where $1 \leq s \leq 3$. Let ε be very small. The costs of opening facilities on the first level are $f_{i_1} = f_{i_2} = \varepsilon$, $f_{i_3} = n\varepsilon$ and $f_{i_4} = n$. The costs of opening facilities on the second level are $\overline{f_{i_1}} = \varepsilon, \overline{f_{i_2}} = (n-1)\,\varepsilon$ and $\overline{f_{i_3}} = (n+1)\,\varepsilon$. The service costs from the first level to the demand points are as follows. $c_{i_1 j_1} = c_{i_2 j_2} = \varepsilon, c_{i_3 j_s} = 1$, for $2 \leq s \leq n+1$, $c_{i_4 j_2} = 2+\varepsilon, c_{i_4 j_s} = 2\varepsilon$ for $3 \leq s \leq n+1, c_{i_4 j_{n+2}} = \varepsilon$. Finally, the service costs between the two levels of facilities are $c_{i_1 \overline{i_1}} = 1, c_{i_2 \overline{i_1}} = 1+\varepsilon, c_{i_3 \overline{i_2}} = 2\varepsilon$ and $c_{i_4 \overline{i_3}} = \varepsilon$. The rest of the costs are chosen as the maximum costs for which the triangle inequality is satisfied. Hence, $c_{i_4 j_2} = 2+2\varepsilon$ and $c_{i_1 j_s} = 4+2\varepsilon$, for $3 \leq s \leq n+1$.

Suppose that we run the primal-dual algorithm on this instance. The first fully paid facility on the second level is $\overline{i_1}$. Since i_1 is its predecessor, $p_{\overline{i_1}} = \left(i_1, \overline{i_1}\right)$ is a central path and both i_1 and $\overline{i_1}$ are opened. The first two demand points get connected to $\overline{i_1}$: j_1 via the path $\left(i_1, \overline{i_1}\right)$ and j_2 via $\left(i_2, \overline{i_1}\right)$ and therefore the algorithm assigns them to the path $p_{\overline{i_1}}$.

The next facility that becomes fully paid is $\overline{i_2}$. Since $p_{\overline{i_2}} = \left(i_2, \overline{i_2}\right)$ and j_2 contributes to i_2, no facility on $p_{\overline{i_2}}$ will be opened. Moreover, because all the demand points $j_3, ..., j_{n+1}$ have the connecting path ending in $\overline{i_2}$, they will be assigned to $p_{\overline{i_1}}$. Finally, the last facility to be opened is $\overline{i_3}$. The associated path is $p_{\overline{i_3}} = \left(i_4, \overline{i_3}\right)$. The set of clients associated to $\overline{i_3}$, being $D_{\overline{i_3}} = \{j_3, ..., j_{n+2}\}$, does not intersect the set of clients associated to $\overline{i_1}$, $D_{\overline{i_3}} = \{j_1, j_2\}$. Therefore, $p_{\overline{i_3}}$ is a central path, i_4 and $\overline{i_3}$ are opened and j_{n+2} is assigned to $p_{\overline{i_3}}$. The opening facility cost of this solution is $n\left(1+\varepsilon\right)+3\varepsilon$, while the service cost is $n\left(5+2\varepsilon\right)-1+3\varepsilon$. Hence, the total cost of the solution is $6n-1+\left(3n+6\right)\varepsilon$.

The optimal solution is to open facilities on $\left(i_1, \overline{i_1}\right)$ and on $\left(i_3, \overline{i_2}\right)$ and to assign the first demand point to the first path and the others to the second path. The cost of opening facilities is $\left(2n+1\right)\varepsilon$ and the service cost is $n+2+\left(2n+7\right)\varepsilon$. This implies that the minimum cost is $n+2+\left(4n+8\right)\varepsilon$.

For a very small ε and for n approaching infinity, the solution given by the algorithm incurs a cost almost 6 times the optimal cost.

4.3 Extensions

4.3.1 A capacitated version

The following capacitated version has been considered in the literature (see [JV01]): Each $i \in F$ has an associated *node capacity* $u_i \in N$ which is an upper bound on the number of paths using i. On the other hand, we are allowed to open as many copies of i (at cost f_i each) as needed.

To formulate this, we replace the $(0,1)$ variables y_i in (P_{int}) by nonnegative integer variables $y_i \in Z_+$, indicating the number of open copies of $i \in F$. Furthermore, we add *capacity constraints*

$$\sum_{j\in D}\sum_{p\ni i} x_{jp} \leq u_i y_i, \text{ for each } i \in F. \tag{4.10}$$

Again, we let C_{LP} denote the optimum value of the corresponding $LP-$relaxation.

The idea to approach the capacitated case (also implicit in [JV01] for the one level case) is to move the capacity constraints to the objective using Lagrangian multipliers

$\lambda_i \geq 0$, for each $i \in F$. This results in an uncapacitated problem

$$\begin{aligned} C(\lambda) &:= \text{minimize } c(x) + f(y) + \sum_{i \in F} \lambda_i \left(\sum_{j \in D} \sum_{p \ni i} x_{jp} - u_i y_i \right) \\ &= \text{minimize } \widetilde{c}(x) + \widetilde{f}(y), \end{aligned}$$

with $\widetilde{f}_i = f_i - \lambda_i u_i$, for each $i \in F$ and $\widetilde{c}(e) = c(e) + \lambda_i$ if i is the endpoint of $e \in E$. Note that each $\lambda \geq 0$ gives $C(\lambda) \leq C_{LP}$.

As in the previous section we compute a primal $(0, 1)$ solution (x, y) of $C(\lambda)$ with

$$\widetilde{c}(x) + \widetilde{f}(y) \leq 6C(\lambda).$$

Note that this does not necessarily satisfy the capacity constraints (4.10). However, a clever choice of the Lagrangian multipliers $\lambda_i = \frac{1}{2} \frac{f_i}{u_i}$ $(i \in F)$ yields

$$\begin{aligned} \widetilde{c}(x) + \widetilde{f}(y) &= c(x) + \frac{1}{2} \sum_{i \in F} \frac{f_i}{u_i} \sum_{p \ni i} \sum_{j \in D} x_{jp} + \frac{1}{2} \sum_{i \in F} f_i y_i \\ &\geq c(x) + \frac{1}{2} \sum_{i \in F} f_i \overline{y_i}, \end{aligned}$$

where $\overline{y_i} := \left\lceil \frac{1}{u_i} \sum_{p \ni i} \sum_{j \in D} x_{jp} \right\rceil$ opens each facility $i \in F$ sufficiently many times. Hence $(x, \overline{y})$ is indeed a feasible solution of the capacitated problem satisfying

$$c(x) + \frac{1}{2} f(\overline{y}) \leq 6C(\lambda) \leq 6C_{LP}.$$

hhence

$$c(x) + f(\overline{y}) \leq 12C_{LP} \leq 12C_{OPT}.$$

Theorem 4.3 *The primal dual algorithm yields a performance guarantee* 12 *for the capacitated case in which multiple copies of a facility are allowed.*

4.3.2 The multilevel facility location problem with outliers

In this version of the multilevel facility location problem, to each demand point $j \in D$ is assigned a penalty α_j. For each demand point we may decide to either provide service and pay the service cost or to pay the penalty. The demand points for which the second variant is chosen are the *outliers*. This formulation is more realistic than the $MFLP$, since it avoids opening an expensive facility for serving just a few number of clients. Clearly, setting the penalties to ∞ gives the standard formulation.

Next we show how to modify the primal-dual algorithm presented for the $MFLP$ in order to obtain a $6-$approximate solution for this problem.

In formulating the problem as an integer program we maintain the same variables as in the (**MP**) and introduce new variables s_j, for every $j \in D$, to indicate whether demand point j is served ($s_j = 0$) or not ($s_j = 1$). We denote the total penalty associated with a solution (x, y, s) by $\alpha(s)$, i.e.,

$$\alpha(s) = \sum_{j \in D} \alpha_j s_j.$$

The following integer program describes the multilevel facility location problem with outliers:

$$(\mathbf{MPO}_{\text{int}}) \qquad \begin{aligned} &\text{minimize } c(x) + f(y) + \alpha(s) \\ &\text{subject to} \sum_{p \in P} x_{jp} + s_j \geq 1, && \text{for each } j \in D \\ &\sum_{p: p \ni i} x_{jp} \leq y_i, && \text{for each } i \in F,\ j \in D \\ &x_{jp}, y_i, s_j \in \{0,1\}, && \text{for each } p \in P,\ j \in D \text{ and } i \in F. \end{aligned} \tag{4.11}$$

We associate to this integer program the linear programming relaxation obtained by replacing constraints (4.11) with the condition that all variables are positive. The dual program becomes:

$$(\mathbf{MDO}) \qquad \begin{aligned} &\text{maximize} \sum_{j \in D} v_j \\ &\text{subject to } v_j - \sum_{i \in p} t_{ij} \leq c(jp), && \text{for each } p \in P,\ j \in D \\ &\sum_{j \in D} t_{ij} \leq f_i, && \text{for each } i \in F \\ &v_j \leq \alpha_j, && \text{for each } j \in D \qquad (4.12) \\ &v_j, t_{ij} \geq 0, && \text{for each } j \in D, i \in F. \end{aligned}$$

The only difference between the dual programs (**MD**) and (**MDO**) is the constraint (4.12). Hence, we can construct a dual feasible solution as in the case without outliers, with the condition to stop increasing the dual variables corresponding to a demand point

j as soon as one of the following events happens: $v_j = \alpha_j$ or j gets connected. The construction of the primal solution is done in the same way as for the $MFLP$, with the exception that demand points j with $v_j = \alpha_j$ are not assigned to any path. However, they may contribute towards opening facilities along one central path.

By doing a similar analysis, we can prove

Theorem 4.4 *The adaptation of the primal-dual algorithm to the case with outliers gives a solution with a value within* 6 *times the optimum.*

4.4 Conclusions and further research

In this chapter we have presented approximation algorithms for the multilevel facility location problem. Our main result was a combinatorial algorithm achieving a 6-performance guarantee for the uncapacitated version of the problem (and 12 for a capacitated version). This result was recently improved by Ageev [Ag02], who presented a reduction of the multilevel (uncapacitated) facility location problem to the classical uncapacitated facility location problem, such that having an $\alpha-$ approximation algorithm for the latest, we obtain a $3\alpha-$ approximation algorithm for the first. This implies that the multilevel facility location problem can now be approximated within a factor of 4.56. Moreover, using the technique of Lagrangian multipliers, Ageev also obtains a $9-$ approximation algorithm for the capacitated version of the problem described in the previous section. However, the performance guarantees of these algorithms are still large compared to the 3-performance guarantee of the LP-rounding algorithm of Aardal, Chudak and Shmoys. One direction of further research could be to obtain a performance guarantee better than 3 for this problem. Another interesting direction of research would be to look at the capacitated version of the problem, under more restrictive conditions on the number of facilities that may be built at a location. For the multilevel case, there are no approximation algorithms for this type of problem.

Chapter 5

Approximation algorithms for uncapacitated facility location problems in the maximization version

The maximization version of the uncapacitated facility location problem, denoted by $MAX\ UFLP$, can be formulated as follows: given a set of demand points, a set of possible facility locations and the profit that might be obtained if a demand point is assigned to a facility, one has to decide where to open facilities in order to maximize the total profit. The total profit is defined as the profit obtained from assigning demand points to facilities minus the cost of opening facilities.

In Section 5.1 we introduce the problem, review the known results on its computational complexity and give an integer programming formulation. In Section 5.2 we describe the main ideas of the known approximation algorithms for this problem. We will concentrate on the greedy algorithm developed by Cornuejols, Fisher and Nemhauser [CFN77] and on the linear programming based algorithm developed by Ageev and Sviridenko [AS99]. Section 5.3 is dedicated to the two level facility location problem (the two level $MAX\ UFLP$), in which there are two levels of facilities and each demand point has to be assigned to a pair of open facilities (one on each level) such that the total profit is maximized. For this problem we propose a $0.47-$approximation algorithm based on linear programming rounding. We end the chapter in Section 5.4 with conclusions and some ideas about further research.

5.1 The one level uncapacitated facility location problem in the maximization version

Reconsider the example presented at the beginning of the first chapter, about the media company that plans to place newspaper stands in a city. The company has already identified potential stand sites in a number of different neighborhoods and knows the cost of placing and maintaining a stand at each potential site. Now suppose that the company is not interested in minimizing the average travel time of the customers and the placing and maintaining costs of the stands, but is interested to maximize its profit. If the profit is measured as the average profit gained from the sales at all locations, minus the costs of placing and maintaining the stands, the problem can be modeled as an uncapacitated facility location problem in the maximization version ($MAX\ UFLP$).

The uncapacitated facility location problem in the maximization version ($MAX\ UFLP$) is formulated as follows. There are given a set of potential facility locations F and a set of demand points D. The opening of a facility at location $i \in F$ costs $f_i \in R_+$. The assignment of demand point $j \in D$ to facility $i \in F$ gives a profit $c_{ij} \in R_+$. Each demand point has to be assigned to one facility. The question is at which locations to open facilities and how to assign demand points to facilities such that the total profit (the profit obtained from the assignment minus the cost of opening facilities) is maximized.

More formally, the problem can be formulated as

$$\max_{S \subseteq F} C(S) = \max_{S \subseteq F} \left(\sum_{j \in D} \max_{i \in S} c_{ij} - \sum_{i \in S} f_i \right).$$

Like the minimization version, the maximization version of the uncapacitated facility location problem was proved not to be in $PTAS$, unless $\mathcal{P} = \mathcal{NP}$ (see Ageev and Sviridenko [AS99] for details).

However, if the problem is formulated on trees, the optimal solution can be found in polynomial time (see Cornuejols, Nemhauser and Wolsey [CNW90]).

For an extensive survey on the $MAX\ UFLP$ see Cornuejols, Nemhauser and Wolsey [CNW90].

5.1.1 An integer programming formulation

Next we will introduce the most commonly used integer programming formulation of the $MAX\ UFLP$. For every facility $i \in F$ and demand point $j \in D$, the variable y_i will indicate whether facility i is open and the variable x_{ij} will indicate whether j is assigned to i.

Denote by $c(x)$ the profit gained from the assignment described by x and by $f(y)$ the cost of opening facilities as described by y,i.e.,

$$c(x) = \sum_{j \in D} c_{ij} x_{ij} \text{ and } f(y) = \sum_{i \in F} f_i y_i.$$

The $MAX\ UFLP$ can then be described as an integer linear program as follows:

$$\begin{aligned}
(P_{int}) \qquad & \text{maximize } c(x) - f(y) & \\
& \text{subject to } \sum_{i \in F} x_{ik} = 1, \text{ for each } k \in D & (5.1)\\
& \qquad x_{ik} \le y_i \text{ , for each } i \in F, k \in D & (5.2)\\
& \qquad x_{ik}, y_i \in \{0,1\}\text{ , for each } i \in F, k \in D &
\end{aligned}$$

Constraints (5.1) ensure that each demand point k is assigned to a facility and constraints (5.2) ensure that a demand point can be assigned only to an open facility.

Denote by (P_{LP}) the linear program obtained by relaxing the variables in (P_{int}) to be non-negative. Clearly, $C_{LP} \ge C_{OPT}$, where C_{LP} is the optimal value of (P_{LP}) and C_{OPT} is the optimal value of the problem.

Next we will give another interpretation to the objective function, that will help in the analysis of the approximation algorithm that we will present.

Let (x, y) be a feasible solution to (P_{LP}). Denote $|F| = n$.

For every demand point $k \in D$ order the facilities in F such that

$$c_{i_1(k)k} \ge c_{i_2(k)k} \ge \dots \ge c_{i_n(k)k}.$$

For every $s \in \{1, ..., n\}$ and demand point $k \in D$, define the set I_{sk} as being the set of the s most profitable facilities for k:

$$I_{sk} = \{i_1(k), ..., i_s(k)\}.$$

For each set I_{sk} let the variable t_{sk} indicate the fraction in which k is assigned to facilities in I_{sk}. In other words,

$$t_{sk} = \sum_{i \in I_{sk}} x_{ik}.$$

Further, associate to each set a number w_{sk} defined by

$$w_{sk} = c_{i_s(k)k} - c_{i_{s+1}(k)k}, \text{ for } 1 \leq s \leq n-1 \tag{5.3}$$

and put

$$w_{nk} = 0. \tag{5.4}$$

Using $(5.1), (5.3)$ and (5.4) the objective function value of (P_{LP}) corresponding to (x, y) can be rewritten as

$$\begin{aligned} c(x) - f(y) &= \sum_{k \in D} (\sum_{s=1}^{n} w_{sk} t_{sk} + \min_{i \in F} c_{ik}) + \sum_{i \in F} f_i (1 - y_i) - \sum_{i \in F} f_i \\ &= \sum_{k \in D} \sum_{s=1}^{n} w_{sk} t_{sk} + \sum_{i \in F} f_i (1 - y_i) + C_R. \end{aligned}$$

Hence,

$$c(x) - f(y) - C_R = \sum_{k \in D} \sum_{s=1}^{n} w_{sk} t_{sk} + \sum_{i \in F} f_i (1 - y_i). \tag{5.5}$$

5.1.2 Approximation algorithms

Although many heuristics were proposed for the $MAX\ UFLP$, very little is known about their worst case performance. First we will discuss how one can define good measures for the quality of the solutions given by heuristics in the case of the $MAX\ UFLP$.

A commonly used measure in case of maximization problems is $\frac{C(S)}{C_{OPT}}$, where S is the solution returned by a heuristic. Since the objective value of the $MAX\ UFLP$ may take negative values, this measure is inadequate for it. We may then think to use $C_{OPT} - C(S)$, but this measure is not good, since it is sensitive to scale changes in the data. Cornuejols, Fisher and Nemhauser proposed as measure $\frac{C_{OPT}-C(S)}{C_{OPT}-C_R}$, where C_R is a lower bound on the minimum objective value of the $MAX\ UFLP$. $C_{OPT} - C_R$ can be seen as the worst absolute deviation that can be achieved by a heuristic. Then $\frac{C_{OPT}-C(S)}{C_{OPT}-C_R}$ measures the deviation for a particular heuristic to the worst possible deviation. Moreover, $\frac{C_{OPT}-C(S)}{C_{OPT}-C_R}$ has both qualities mentioned before, namely that is insensitive to scale changes in the data and both the nominator and denominator are positive.

Note that the study of $\frac{C_{OPT}-C(S)}{C_{OPT}-C_R}$ is equivalent to the study of $\frac{C(S)-C_R}{C_{OPT}-C_R}$.

In the case of the $MAX\ UFLP$, we choose $C_R = \sum_{j \in D} \min_{i \in F} c_{ij} - \sum_{i \in F} f_i$. Intuitively, C_R represents the minimum profit that can be obtained, i.e., the profit obtained by opening all facilities and assigning each demand point to the least profitable facility.

In the rest of the section we will call a $\rho-$approximation algorithm for the $MAX\ UFLP$ an algorithm that returns a solution S satisfying

$$\frac{C(S) - C_R}{C_{OPT} - C_R} \geq \rho.$$

We call ρ the performance guarantee of the algorithm.

The first approximation algorithm for the $MAX\ UFLP$ was proposed by Cornuejols, Fisher and Nemhauser [CFN77]. They proved that a simple greedy heuristic (that was also proposed by Kuehn and Hamburger [KH63] and Spielberg [Sp69]) achieves a performance guarantee equal to $\left(1-\frac{1}{e}\right)$ $(1-\frac{1}{e} \simeq 0.67)$.

There were no other results on approximation algorithms for the $MAX\ UFLP$ until recently, when Ageev and Sviridenko [AS99] improved the performance guarantee to $2\left(\sqrt{2}-1\right) \simeq 0.828$. Their algorithm is based on rounding an optimal solution of (P_{LP}) to a feasible solution of (P_{int}). Moreover, they proved that

$$\frac{C_{OPT} - C_R}{C_{LP} - C_R} \leq 2\left(\sqrt{2}-1\right),$$

which implies that $2\left(\sqrt{2}-1\right)$ is the best performance guarantee one can get by rounding a solution of (P_{LP}).

Next we give a more detailed discussion of the greedy heuristic and of the algorithm proposed by Ageev and Sviridenko.

The greedy heuristic

The greedy heuristic forms a solution by opening locations sequentially so as to maximize the increase in the objective function at each step. More precisely, it proceeds as follows.

Initially, all facilities are closed. In every iteration, let S be the set of facilities opened so far. For every closed facility i, let $gain(i) = C(S \cup \{i\}) - C(S)$ be the increase in the objective value if facility i would be opened. If there are facilities with positive gain, open the one with maximum gain. Stop when there are no facilities with positive gain.

The proof of Cornuejols, Fisher and Nemhauser makes use of a Lagrangian relaxation of (P_{LP}) in which the constraints $\sum\limits_{i \in F} x_{ik} = 1$ are moved to the objective function with Lagrangian multipliers. Since the coefficient matrix of the remaining constraints is totally unimodular, a result of Geoffrion [Ge74] implies that the value of the Lagrangian dual, denoted by L_D, is equal to C_{LP}, the optimal value of the LP-relaxation. After reformulating the greedy heuristic in terms of the Lagrangian relaxation, one can show [CFN77] that the greedy solution S satisfies $C(S) - C_R \geq \left(1 - \frac{1}{e}\right)(L_D - C_R)$. Since $L_D \geq C_{OPT}$ we conclude that $C(S) - C_R \geq \left(1 - \frac{1}{e}\right)(C_{OPT} - C_R)$ holds. Thus, the greedy algorithm is a $(1 - \frac{1}{e})-$ approximation algorithm for the $MAX\ UFLP$.

The 0.828-approximation algorithm of Ageev and Sviridenko

The first step of the algorithm proposed by Ageev and Sviridenko is to reduce the $MAX\ UFLP$ to a special case of $MAX\ SAT$ in a way that would preserve approximation guarantees. For the latter they propose an approximation algorithm similar to the one developed by Goemans and Williamson [GW94] for $MAX\ SAT$. In order to explain how the algorithm works for the $MAX\ UFLP$, we have chosen for a presentation that avoids the reduction to $MAX\ SAT$.

The idea is to obtain a feasible solution (x, y) of (P_{int}) by rounding an optimal solution to (P_{LP}) such that

$$c(x) - f(y) - C_R \geq \rho(C_{LP} - C_R) \geq \rho(C_{OPT} - C_R).$$

The second always holds, since $C_{LP} \geq C_{OPT}$.

Let $(\overline{x}, \overline{y})$ be an optimal solution to (P_{LP}) and let $\overline{t_{sk}} = \sum\limits_{i \in I_{sk}} \overline{x_{sk}}$.

The algorithm is very simple. It independently opens each facility $i \in F$ with some probability p_i and then assigns each demand point to the most profitable open facility. In this way we obtain a feasible solution to the integer program. However, the quality of the solution obtained depends on the choice of p_i.

More precisely, we proceed as follows.

Let $\lambda \in [0,1]$. At the beginning all the facilities are closed. Open independently each facility i with probability $p_i = (1-\lambda) + \lambda \overline{y_i}$. Assign each $k \in D$ to the open facility $i \in F$ with the largest profit c_{ik} among the open facilities.

Denote by $C(\lambda)$ the value returned by the algorithm. To analyze the performance of the algorithm we compare $E(C(\lambda)) - C_R$ with $C_{LP} - C_R$.

For each facility i, denote by Y_i the random variable that indicates whether facility i is open ($Y_i = 1$ with probability p_i and $Y_i = 0$ with probability $1 - p_i$). For each $k \in D$ and $s = 1, ..., n$, denote by T_{sk} the random variable that indicates that k is assigned to a facility $i \in I_{sk}$.

From (5.5) and the linearity of the expectation we find:

$$E(C(\lambda)) - C_R = \sum_{k\in D}\sum_{s=1}^{n} w_{sk} Prob(T_{sk}=1) + \sum_{i\in F} f_i (1 - Prob(Y_i = 1)).$$

One can then show (cf. [AS99] or our analysis of the 2−level version, Case 2) in Section 5.2.2.) that

$$Prob(T_{sk}=1) \geq [1 - \lambda^s \left(1 - \frac{1}{s}\right)^s]\overline{t_{sk}}.$$

Based on (5.5) we obtain

$$\begin{aligned} E(C(\lambda)) - C_R &\geq \sum_{k\in D}\sum_{s=1}^{n} [1 - \lambda^s \left(1-\frac{1}{s}\right)^s] w_{sk}\overline{t_{sk}} + \lambda \sum_{i\in F} f_i(1-\overline{y_i}) \\ &\geq \min\left\{\lambda, \min_{s\geq 1}\{1 - \lambda^s\left(1-\frac{1}{s}\right)^s\}\right\}(C_{LP} - C_R) \\ &\geq \min\left\{\lambda, \min_{s\geq 1}\{1 - \lambda^s\left(1-\frac{1}{s}\right)^s\}\right\}(C_{OPT} - C_R). \end{aligned}$$

According to the method of conditional probabilities
$\phi(\lambda) = \min\left\{\lambda, \min_{s\geq 1}\{1-\lambda^s\left(1-\frac{1}{s}\right)^s\}\right\}$ is the performance guarantee of the derandomized algorithm. Since $\max_{\lambda\in[0,1]} \phi(\lambda) = 2\left(\sqrt{2}-1\right) \simeq 0.828$ is obtained for $\lambda^* = 2\left(\sqrt{2}-1\right) \simeq 0.828$ we conclude that the algorithm presented above, applied for λ^*, is a 0.828-approximation algorithm.

In the next section we extend this algorithm to the two level problem.

5.2 The two level facility location problem in the maximization version

The multilevel problem, of course, also applies to the maximization version. We have seen in Chapter 4 that problems in which the placement of facilities must be done hierarchically can be modelled as multilevel facility location problems. In this section we will present results for the case where there are only two levels of facilities. Unlike for the minimization version, it is very difficult to generalize the results obtained for two levels to k levels of facilities, $k > 2$.

The two level uncapacitated facility location problem in the maximization version (the two level $MAX\ UFLP$) can be described as follows. There are two types of potential facility locations: the hub facilities, denoted by F and the transit facilities, denoted by E. Building (opening) a facility $i \in F$ or $j \in E$ has an associated nonnegative cost f_i, respectively e_j. There is also a set of clients, denoted by D, who should be assigned to open pairs of facilities from $F \times E$. If a client $k \in D$ is assigned to the pair (i, j), a profit c_{ijk} is obtained. The problem is to decide which facilities from F and which from E to open simultaneously (at least one from each set) and how to assign the clients to the open facilities such that the total profit (the profit obtained from assigning demand points to facilities minus the cost of opening facilities) is maximized.

Formally, the problem can be formulated in the following way:

$$\max_{S_1 \times S_2 \subseteq F \times E} C(S_1, S_2) = \max_{S_1 \times S_2 \subseteq F \times E} \sum_{k \in D} \max_{(i,j) \in S_1 \times S_2} c_{ijk} - \sum_{i \in S_1} f_i - \sum_{j \in S_2} e_j. \qquad (5.6)$$

Denote by $C_R = \sum\limits_{k \in D} \min_{(i,j) \in F \times E} c_{ijk} - \sum\limits_{i \in F} f_i - \sum\limits_{j \in E} e_j$ and by C_{OPT} the optimal value of the problem.

Clearly, $C_{OPT} \geq C(S_1, S_2) \geq C_R$, for each $S_1 \times S_2 \subseteq F \times E$. Note that the objective function $C(S_1, S_2)$ can take both positive and negative values and so there is a difficulty in the definition of measure of relative deviation for approximate solutions to (5.6) . To overcome this, we will proceed as in the one level case and consider the shifted objective function $C(S_1, S_2) - C_R$, which takes only positive values.

If either E or F is a singleton, one obtains the one level uncapacitated facility location problem, in the maximization version (the $MAX\ UFLP$). Therefore we may assume that $|F| \geq 2$ and $|E| \geq 2$.

There is considerably less literature on the two level $MAX\ UFLP$ than on the $MAX\ UFLP$ or the $UFLP$. The techniques which have been used for tackling this problem are branch-and-bound (Kaufman *et al.* [KEH77], Tcha and Lee [TL84]) Lagrangian relaxation, cutting planes (Aardal [Aa92]). Some structural properties of the problem were studied by Aardal *et al.* [ALLQ96]. To the best of our knowledge, the approximation algorithm we will present in Section 5.2.2 is the first one for the two level $MAX\ UFLP$ (we define an approximation algorithm for the two level $MAX\ UFLP$ with respect to the shifted objective values, as in the case of the $MAX\ UFLP$). This algorithm has a 0.47 performance guarantee and is based on randomized rounding. Very recently, Zhang and Ye [ZY02] developed a very simple $0.5-$ approximation algorithm for the case when the facilities are situated on k levels, where $k \in N$.

5.2.1 An integer programming formulation

To derive an integer programming formulation of the two level $MAX\ UFLP$, we introduce the $(0,1)$ variables y_i $(i \in F)$ and z_j $(j \in E)$ to indicate whether $i \in F$, respectively $j \in E$ is open and the $(0,1)$ variables x_{ijk} $(i \in F, j \in E, k \in D)$ to indicate whether demand point k is served by the pair (i,j).

We call a pair $(i,j) \in F \times E$ *open* if both i and j are open.

We let

$$c(x) = \sum_{k \in D, i \in F, j \in E} c_{ijk} x_{ijk},$$
$$f(y) = \sum_{i \in F} f_i y_i,$$

and

$$e(z) = \sum_{j \in E} e_j z_j.$$

The two level $MAX\ UFLP$ is now equivalent to

$$\begin{aligned}
(P^2_{int}) \quad & \text{maximize } c(x) - f(y) - e(z) \\
& \text{subject to } \sum_{i\in F, j\in E} x_{ijk} = 1, \text{ for each } k \in D && (5.7)\\
& \qquad x_{ijk} \le y_i \text{ , for each } i \in F,\ j \in E, k \in D && (5.8)\\
& \qquad x_{ijk} \le z_j \text{ , for each } i \in F, j \in E, k \in D && (5.9)\\
& \quad x_{ijk}, y_i, z_j \in \{0,1\} \text{ , for each } i \in F, j \in E, k \in D.
\end{aligned}$$

Constraints (5.7) ensure that each $k \in D$ is assigned to only one pair of facilities and constraints (5.8) and (5.9) ensure that only open pairs are used.

We consider the LP-relaxation of (P^2_{int}) with all variables taking values in $[0,1]$. Denote this LP-relaxation by (P^2_{LP}).

Next we will give another interpretation of the objective function of (P^2_{LP}), which is similar to (5.5). Let (x,y,z) be a feasible solution to (P^2_{LP}) and let $|F| = m$ and $|E| = n$. For each $k \in D$, we order the $p = mn$ pairs (i,j) such that

$$c_{i_1(k)j_1(k)k} \ge c_{i_2(k)j_2(k)k} \ge \ldots \ge c_{i_p(k)j_p(k)k}.$$

For every $s \in \{1,...p\}$ define the set I_{sk} as being the set of the s most profitable pairs for k :

$$I_{sk} = \{(i_1(k), j_1(k)), ..., (i_s(k), j_s(k))\}.$$

For each set I_{sk} let the variable t_{sk} indicate the fraction in which k is assigned to pairs in I_{sk}. In other words,

$$t_{sk} = \sum_{(i,j)\in I_{sk}} x_{ijk}.$$

Furthermore, we define w_{sk} by

$$w_{sk} = c_{i_s(k)j_s(k)k} - c_{i_{s+1}(k)j_{s+1}(k)k}, \text{ for } s \le p-1 \tag{5.10}$$

and put

$$w_{pk} = 0. \tag{5.11}$$

As in the one level case, the objective function value of (P^2_{LP}) corresponding to (x,y,z)

can be rewritten as

$$\begin{aligned} c(x) - f(y) - e(z) &= \sum_{k\in D}\sum_{s=1}^{p} w_{sk}t_{sk} + \min_{(i,j)\in F\times E} c_{ijk} \\ &+ \sum_{i\in F} f_i(1-y_i) + \sum_{j\in E} e_j\,(1-z_j) - \sum_{i\in F} f_i - \sum_{j\in E} e_j \\ &= \sum_{k\in D}\sum_{s=1}^{p} w_{sk}t_{sk} + \sum_{i\in F} f_i(1-y_i) + \sum_{j\in E} e_j\,(1-z_j) + C_R. \end{aligned}$$

Hence,

$$c(x) - f(y) - e(z) - C_R = \sum_{k\in D}\sum_{s=1}^{p} w_{sk}t_{sk} + \sum_{i\in F} f_i(1-y_i) + \sum_{j\in E} e_j\,(1-z_j)\,. \tag{5.12}$$

5.2.2 An approximation algorithm based on linear programming rounding

We will present a $0.47-$approximation algorithm for the two level $MAX\ UFLP$. This section is based on Bumb [Bu01].

The algorithm that we will present is similar to the one developed by Ageev and Sviridenko [AS99] for one level of facilities. Our goal is to find a feasible solution (x, y, z) to (P^2_{int}) which satisfies

$$c(x) - f(y) - e(z) - C_R \geq \rho\,(C_{OPT} - C_R)\,, \tag{5.13}$$

for some $\rho < 1$.

Let $(\widetilde{x}, \widetilde{y}, \widetilde{z})$ be an optimal solution of the LP-relaxation (P^2_{LP}), and let C^2_{LP} be its value, i.e.

$$C^2_{LP} = c(\widetilde{x}) - f(\widetilde{y}) - e(\widetilde{z}).$$

Since $C^2_{LP} - C_R \geq C_{OPT} - C_R$, it suffices to exhibit a solution (x, y, z) to (P^2_{int}) satisfying

$$c(x) - f(y) - e(z) - C_R \geq \rho\left(C^2_{LP} - C_R\right).$$

We will obtain (x, y, z) by rounding $(\widetilde{x}, \widetilde{y}, \widetilde{z})$ to an integer solution.

The algorithm independently opens facilities in F and E and then assigns each demand point to the most profitable open pair. The probabilities of opening facilities will be dependent on a parameter $\lambda \in [0, 1]$. The value of λ will be chosen at the end in order to maximize the performance guarantee of the algorithm. We call the randomized algorithm $RANDOM\,(\lambda)$. Now we describe the algorithm in more detail.

Initially, all facilities are closed. Open each facility $i \in F$ with probability $p_i = (1-\lambda) + \lambda\widetilde{y_i}$ and each facility $j \in E$ with probability $q_j = (1-\lambda) + \lambda\widetilde{z_j}$. Finally, assign each demand point $k \in D$ to the open pair (i,j) with the largest profit c_{ijk}.

```
RANDOM (λ)
FOR each $i \in F$
    Open $i$ with probability $p_i = (1-\lambda) + \lambda\widetilde{y_i}$
FOR each $j \in E$
    Open $j$ with probability $q_j = (1-\lambda) + \lambda\widetilde{z_j}$
Assign each $k \in D$ to the open pair $(i,j)$ with largest profit
```

In order to analyze the algorithm we introduce the following random variables. For every $i \in F$ the random variable Y_i will indicate whether facility i was opened by the algorithm ($Y_i = 1$ with probability p_i and $Y_i = 0$ with probability $1 - p_i$). Similarly, for every $j \in E$, Z_j will indicate whether facility j was opened by the algorithm ($Z_j = 1$ with probability q_j and $Z_j = 0$ with probability $1-q_j$). For each $k \in D$ and $s = 1, ..., p$, denote by T_{sk} the random variable that indicates whether k is assigned to a pair $(i,j) \in I_{sk}$.

Denote by $C(\lambda)$ the value returned by the algorithm (note that $C(\lambda)$ is a random variable). In a similar way as we derived (5.12), one can show that

$$C(\lambda) - C_R = \sum_{k\in D}\sum_{s=1}^{p} w_{sk}T_{sk} + \sum_{i\in F} f_i\,(1-Y_i) + \sum_{j\in D} e_j\,(1-Z_j)\,.$$

To analyze the performance of the algorithm we compare $E\,(C(\lambda)) - C_R$ with $C^2_{LP} - C_R$.

First, based on (5.12), we derive the following expression of $C^2_{LP} - C_R$,

$$C^2_{LP} - C_R = \sum_{k\in D}\sum_{s=1}^{p} w_{sk}\widetilde{t_{sk}} + \sum_{i\in F} f_i(1-\widetilde{y_i}) + \sum_{j\in E} e_j\,(1-\widetilde{z_j})\,,$$

where $\widetilde{t_{sk}} = \sum\limits_{(i,j)\in I_{sk}} \widetilde{x_{ijk}}$. From the linearity of the expectation we conclude that

$$\begin{aligned} E\,(C(\lambda)) - C_R &= \sum_{k\in D}\sum_{s=1}^{p} w_{sk}Prob(T_{sk}=1) + \sum_{i\in F} f_i(1 - Prob(Y_i = 1)) \\ &\quad + \sum_{j\in E} e_j(1 - Prob(Z_j = 1)) \\ &= \sum_{k\in D}\sum_{s=1}^{p} w_{sk}Prob(T_{sk}=1) + \lambda\sum_{i\in F} f_i(1-\widetilde{y_i}) + \lambda\sum_{j\in E} e_j(1-\widetilde{z_j}). \end{aligned}$$

Note that in order to obtain a lower bound of $E\left(C(\lambda)\right) - C_R$ it is enough to have a lower bound on the probabilities that T_{sk} takes value 1, for every s and k. To calculate these probabilities we distinguish four cases, depending on the structure of the set I_{sk}. The main idea is that the events of opening facilities are independent.

Case 1. $I_{sk} = \{(i,j)\}$.

In this case we have

$$\begin{aligned} Prob(T_{sk} = 1) &= Prob((Y_i = 1) \wedge (Z_j = 1)) = Prob(Y_i = 1)Prob(Z_j = 1) \\ &= [(1-\lambda) + \lambda\widetilde{y_i}][(1-\lambda) + \lambda\widetilde{z_j}]. \end{aligned}$$

Using the inequality

$$a + b \geq 2\sqrt{ab}, \text{ for } a, b \geq 0,$$

and (5.8) and (5.9) we obtain the following lower bound

$$Prob(T_{sk} = 1) \geq 4(1-\lambda)\lambda\sqrt{\widetilde{y_i}\widetilde{z_j}} \geq 4(1-\lambda)\lambda\widetilde{x_{ijk}}.$$

Hence,

$$Prob(T_{sk} = 1) \geq 4(1-\lambda)\lambda\widetilde{t_{sk}}.$$

Case 2. $I_{sk} = \{(i,j_1), ..., (i,j_r)\}, r \geq 2$.

We have

$$\begin{aligned} &Prob(T_{sk} = 1) = Prob(Y_i = 1)Prob((Z_{j_1} = 1) \vee ... \vee (Z_{j_r} = 1)) = \\ &Prob(Y_i = 1)(1 - Prob((Z_{j_1} = 0) \wedge ... \wedge (Z_{j_r} = 0)) = \\ &Prob(Y_i = 1)(1 - \prod_{q=1}^{r} Prob(Z_{j_q} = 0)) = \\ &[(1-\lambda) + \lambda\widetilde{y_i}][1 - \lambda^r \prod_{q=1}^{r}(1 - \widetilde{z_{j_q}})]. \end{aligned}$$

The arithmetic/geometric mean inequality, applied to $1 - \widetilde{z_{j_q}}$, $q = \overline{1,r}$ gives

$$\prod_{q=1}^{r}(1 - \widetilde{z_{j_q}}) \leq \left(1 - \frac{\sum_{q=1}^{r}\widetilde{z_{j_q}}}{r}\right)^r.$$

Hence, we obtain the following lower bound for $Prob(T_{sk} = 1)$:

$$Prob(T_{sk} = 1) \geq [(1-\lambda) + \lambda\widetilde{y_i}][1 - \lambda^r \left(1 - \frac{\sum_{q=1}^{r}\widetilde{z_{j_q}}}{r}\right)^r].$$

The function $f : [0,1] \to \mathcal{R}$ defined by

$$f(x) = 1 - a\left(1 - \frac{x}{r}\right)^r, \text{ where } a \in [0,1],$$

is concave. Observing that any $x \in [0,1]$ can be written as a convex combination of 0 and 1, the concavity of f implies

$$f(x) \geq xf(1) + (1-x)f(0).$$

Next, from

$$\begin{aligned} f(0) &= 1 - a \geq 0, \\ f(1) &= 1 - a\left(1 - \frac{1}{r}\right)^r, \end{aligned}$$

we conclude that

$$f(x) \geq [1 - a\left(1 - \frac{1}{r}\right)^r]x.$$

Substituting in this inequality $a = \lambda^r$, $x = \sum_{q=1}^{r} \widetilde{z_{j_q}}$ and taking into account that by (5.9) we have $\sum_{q=1}^{r} \widetilde{z_{j_q}} \geq \widetilde{t_{sk}}$, we obtain

$$1 - \lambda^r \left(1 - \frac{\sum_{q=1}^{r} \widetilde{z_{j_q}}}{r}\right)^r \geq [1 - \lambda^r\left(1 - \frac{1}{r}\right)^r] \sum_{q=1}^{r} \widetilde{z_{j_q}} \geq [1 - \lambda^r\left(1 - \frac{1}{r}\right)^r]\widetilde{t_{sk}}.$$

Thus, for this case the lower bound for the probability of T_{sk} being 1 is:

$$Prob(T_{sk} = 1) \geq [(1-\lambda) + \lambda\widetilde{y_i}][1 - \lambda^r\left(1 - \frac{1}{r}\right)^r]\widetilde{t_{sk}} \geq (1-\lambda)[1 - \lambda^r\left(1 - \frac{1}{r}\right)^r]\widetilde{t_{sk}}.$$

Case 3. $I_{sk} = \{(i_1, j), ..., (i_r, j)\}, r \geq 2$.

In a similar way as in the previous case, we obtain also in this case

$$Prob(T_{sk} = 1) \geq (1-\lambda)[1 - \lambda^r\left(1 - \frac{1}{r}\right)^r]\widetilde{t_{sk}}.$$

Case 4. $I_{sk} \supseteq \{(i_1, j_1), (i_2, j_2)\}$ with $i_1 \neq i_2$ and $j_1 \neq j_2$.

In this case, the event that the pair (i_1, j_1) is open is independent of the event that the pair (i_2, j_2) is open and, consequently,

$$\begin{aligned} Prob(T_{sk}=1) &\geq Prob\left[(Y_{i_1}=1 \wedge Z_{j_1}=1) \vee (Y_{i_2}=1 \wedge Z_{j_2}=1)\right] \\ &= Prob\left(Y_{i_1}=1 \wedge Z_{j_1}=1\right) + Prob\left(Y_{i_2}=1 \wedge Z_{j_2}=1\right) \\ &- Prob\left(Y_{i_1}=1 \wedge Z_{j_1}=1\right) Prob\left(Y_{i_2}=1 \wedge Z_{j_2}=1\right) \\ &= p_{i_1}q_{j_1} + p_{i_2}q_{j_2} - p_{i_1}q_{j_1}p_{i_2}q_{j_2} \\ &\geq 2\sqrt{p_{i_1}q_{j_1}p_{i_2}q_{j_2}} - p_{i_1}q_{j_1}p_{i_2}q_{j_2}. \end{aligned}$$

By the definition of p_i and q_j, $p_i \geq 1-\lambda$ and $q_j \geq 1-\lambda$ for each i and j. Hence,

$$p_{i_1}q_{j_1}p_{i_2}q_{j_2} \geq (1-\lambda)^4.$$

The function $f : \mathcal{R}_+ \to \mathcal{R}_+$ defined by $f(x) = 2\sqrt{x} - x$ is increasing on $[0,1]$ which, together with the inequality above implies that

$$Prob(T_{sk}=1) \geq 2\sqrt{(1-\lambda)^4} - (1-\lambda)^4 \geq [2(1-\lambda)^2 - (1-\lambda)^4]\widetilde{t_{sk}}.$$

From the cases $(1)-(4)$ it follows that, for each $\lambda \in [0,1]$,

$$E\left(C(\lambda)\right) - C_R \geq \rho\left(\lambda\right)\left(C_{LP}^2 - C_R\right),$$

where $\rho\left(\lambda\right) = \min\{\lambda, 4\lambda(1-\lambda), 2(1-\lambda)^2 - (1-\lambda)^4, \min_{s\geq 2}(1-\lambda)[1-\lambda^s\left(1-\frac{1}{s}\right)^s]\}$.

Hence, we have proved the following theorem

Theorem 5.1 *The expected value returned by RANDOM (λ) for every $\lambda \in [0,1]$ satisfies*

$$E\left(C(\lambda)\right) - C_R \geq \rho\left(\lambda\right)\left(C_{LP}^2 - C_R\right) \geq \rho\left(\lambda\right)\left(C_{OPT} - C_R\right).$$

Using the fact that $\left(1-\frac{1}{r}\right)^r \leq e^{-1}$, for every $r \geq 1$, we obtain that 0.47 is the maximum value of $\rho\left(\lambda\right)$ for $\lambda \in [0,1]$ and is attained for $\lambda = 0.47$

Thus we obtain

Corollary 5.1 *For $\lambda^* = 0.47$ we have $E\left(C(\lambda^*)\right) - C_R \geq 0.47\left(C_{OPT} - C_R\right)$.*

Remark 5.1 *Note that with a small probability the algorithm does not return a feasible solution. In this case, run the algorithm again.*

5.2.2.1 Derandomization

To apply the method of conditional expectations to derandomize *RANDOM* (λ), we would have to compute $Prob(T_{sk}=1)$ (conditioned on some of facilities being opened already), which is difficult because the events " (i,j) is open " and " (i',j') is open " are dependent when $i=i'$ or $j=j'$.

We overcome this property by restricting our attention to those T_{sk} where $s \le p_k$ and p_k, $p_k \ge 2$, is chosen minimal so that the corresponding $T_{p_k k}$ satisfies the condition in Case 4 of our analysis. Equivalently, we modify the profits so that

$$\widetilde{c}_{i_r(k)j_r(k)k} = \widetilde{c}_{i_p(k)j_p(k)k}, \text{ i.e., } \widetilde{w}_{i_r(k)j_r(k)k} = 0,$$

whenever $r > p_k$.

Note that our analysis of *RANDOM* (λ) actually proves that

$$\widetilde{C}(\lambda) - C_R := \sum_{k\in D}\left(\sum_{s=1}^{p} \widetilde{w}_{sk}T_{sk}\right) + \sum_{i\in F} f_i\left(1-Y_i\right) + \sum_{j\in D} e_j\left(1-Z_j\right)$$

is good enough in expected value (at least $\rho(\lambda)(C_{OPT} - C_R)$).

Remark 5.2 *Equivalently, we derandomize the algorithm that proceeds like RANDOM* (λ), *but assigns* $k \in D$ *to the most profitable pair* $(i_s(k), j_s(k))$ *in* $I_{p_k,k}$ *(in case there is such an open pair) and to* $(i_p(k), j_p(k))$ *otherwise (opening these two facilities if necessary).*

The advantage of modifying c to $\widetilde{c}$, respectively w to $\widetilde{w}$ (or, equivalently, modifying the algorithm *RANDOM* (λ) as explained in the Remark 5.2) is that it suffices to compute $Prob(T_{sk}=1)$ for the five cases listed below (in all other cases $\widetilde{w_{sk}} = 0$ and the corresponding value of T_{sk} is irrelevant) to obtain the desired derandomization.

Case 1 $I_{sk} = (i,j)$
In this case, $T_{sk} = Y_i Z_j$.

Case 2 $I_{sk} = \{(i,j_1),...,(i,j_r)\}, r \ge 2.$
In this case, $T_{sk} = Y_i\left(1 - \prod_{q=1}^{r}\left(1-Z_{j_q}\right)\right).$

Case 3 $I_{sk} = \{(i_1,j),...,(i_r,j)\}, r \ge 2$
In this case, $T_{sk} = Z_j\left(1 - \prod_{q=1}^{r}\left(1-Y_{i_q}\right)\right).$

Case 4. $I_{sk} = \{(i, j_1), ..., (i, j_r), (i', j_{r+1})\}, i \neq i'$
In this case, $T_{sk} = (1 - (1 - Y_i)(1 - Y_{i'})) \left(1 - \prod_{j_q \neq j'_q} (1 - Z_{j_q})\right).$

Case 5. $I_{sk} = \{(i_1, j), ..., (i_r, j), (i_{r+1}, j')\}$, with $j \neq j'$.
In this case, $T_{sk} = (1 - (1 - Z_j)(1 - Z_{j'})) \left(1 - \prod_{i_q \neq i'_q} (1 - Y_{i_q})\right).$

Hence, T_{sk} can always be written as a product of independent variables. Therefore we can easily compute $E\left(\widetilde{C}(\lambda)\right)$ and, similarly, we can easily compute $E\left(\widetilde{C}(\lambda) | Y_1 = y_1, ..., Y_n = y_n, Z_1 = z_1, ..., Z_l = z_l\right)$ for any given values $y_1, ..., y_n, z_1, ..., z_l$. Thus we can derandomize (the modified version of) $RANDOM(\lambda)$ by setting $y_1 := 1$ if and only if $E\left(\widetilde{C}(\lambda) | Y_1 = 1\right) \geq E\left(\widetilde{C}(\lambda) | Y_1 = 0\right)$ and continue this way until all values of y_i and z_j are fixed.

Remark 5.3 *In derandomizing $RANDOM(\lambda)$, the conditional expectations of $E\left(\widetilde{C}(\lambda)\right)$ play the role of the pessimistic estimators described in Section* 1.4.

Theorem 5.2 *By derandomizing $RANDOM(0.47)$ we obtain a deterministic approximation algorithm for the two level $MAX\ UFLP$ with a performance guarantee equal to* 0.47.

5.3 Conclusions and further research

In this chapter we have presented approximation algorithms for uncapacitated facility location problems in the maximization version.

We have seen that for the case with one level of facilities, there is a 0.828-approximation algorithm and that this is the best approximation guarantee one can hope to obtain by using the LP-relaxation (P_{LP}). One question that remains open is whether 0.828 is the best performance guarantee one can get for this problem. If this is not the case, one direction of further research on this problem would be to look for other relaxations that would help to improve the performance guarantee. We have also seen a simple greedy algorithm that achieves a $0.67-$performance guarantee. Another direction of research would be to look for combinatorial algorithms that have a better performance guarantee. Since the dual descent algorithm developed by Erlenkotter [Er78] performs very well for most of the instances, an algorithm based on this technique may also have a good worst case performance.

For the two level $MAX\ UFLP$ we have presented a $0.47-$ approximation algorithm. For the analysis of our algorithm the assumption that there are only two levels of facilities was essential. A natural question is whether the algorithm generalizes to the case where the facilities are located on k levels, with $k \geq 2$. The problem that occurs is that even if we open the facilities independently, the events corresponding to paths being opened become dependent. As a consequence, for $k > 2$ the analysis of the algorithm is much more difficult. Very recently, Zhang and Ye [ZY02] have developed a simple combinatorial algorithm achieving a $0.5-$performance guarantee, for any number k of levels. In order to improve their result, one might try different relaxations or better rounding methods then the one used in Section 5.2.2.

Summary

Facility location problems are among the most well-studied problems in optimization literature. The simplest variant is the uncapacitated facility location problem, which we denoted by the $UFLP$. In the $UFLP$, we are given a set of possible facility locations and a set of clients. The problem seeks to find a set of locations to build/open facilities such that the sum of the cost of building/opening the facilities and the cost of serving each client from exactly one open facility is minimized. This problem is $NP-$hard. Therefore, many heuristics for finding good approximate solutions were developed.

In this thesis we design approximation algorithms for several variants of the $UFLP$.

We call a ρ approximation algorithm an algorithm that finds in polynomial time a solution with a value at most (at least) ρ times the optimal value of a minimization (maximization) problem.

A very powerful tool in designing approximation algorithms is Linear Programming. In the algorithms presented in this thesis we have used two techniques based on it: linear programming rounding and the primal-dual technique. In both cases, one has to formulate the problem as an integer program and to give a linear programming relaxation of it. If we deal with a minimization (maximization) problem, the optimal value of the linear program will give a lowerbound (upperbound) on the optimal value. In linear programming rounding, one rounds the optimal solution of the linear programming relaxation to an integer one, such that the rounded solution is feasible and has an objective value of at most (at least) ρ times the optimal value of the linear programming relaxation.

The drawback of the linear programming rounding technique is that if the number of constraints or variables is large, solving a linear program is not trivial. This problem can be sometimes overcomed by using the primal-dual technique. For minimization problems, the primal-dual technique uses the following idea. The dual of the linear program is a

maximization problem, and by duality theory, any feasible solution to the dual has an objective value smaller than the objective value of any feasible solution of the primal. Hence, the objective value of any dual feasible solution can be used as a lower bound of the optimal value of the problem. If the structure of the dual linear program is simple, we can create a feasible solution combinatorially. Then we create an integral solution of the primal problem which is bounded in terms of the dual solution.

After some preliminaries in Chapter 1, in Chapter 2 we review the known approximation algorithms for the metric $UFLP$, i.e., the variant of the $UFLP$ in which the service costs are induced by a metric. In particular, we concentrate on the algorithms developed by Chudak and Shmoys [CS98] and by Jain and Vazirani [JV01]. These algorithms are good examples of how linear programming rounding and the primal-dual technique can be used in the context of the $UFLP$. At the end of the chapter, we review the main ideas of an useful procedure developed by Guha and Khuller [GK99] and Charikar and Guha [CG99], which in many cases leads to a considerable improvement of the performance guarantee of the approximation algorithms for facility location problems.

Chapter 3 is dedicated to the fault tolerant facility location problem, the $TFLP$. The $TFLP$ is similar to the metric $UFLP$, the only difference being that each demand point j has to be assigned to r_j different open facilities. Unlike for the metric $UFLP$, there are no combinatorial approximation algorithms for the $TFLP$. The only approximation algorithm for this problem, developed by Guha, Meyerson and Munagala, [GMM01] makes use of linear programming rounding and has a performance gurantee 4. By using filtering and greedy improvement, the performance guarantee can be reduced to 2.47. We propose a simpler $4-$ approximation algorithm for the $TFLP$, based on linear programming rounding as well. However, we are not able to further improve the performance guarantee. In the last part of Chapter 3 we focus on the case when $r_j = r, r > 1$ for each demand point j. For this problem, there is a $2-$approximation algorithm based on dual fitting, developed by Mahdian *et al.* [MMSV01]. We show how an extension of the primal-dual algorithm developed by Jain and Vazirani [JV01] for the $UFLP$, combined with greedy improvement and scaling, give a performance guarantee 1.85.

In Chapter 4 we focus on the multilevel facility location problem, the $MFLP$. In this problem there are k types of facilities to be opened: one type of depots and $(k-1)$ types of transit stations. Each demand must be shipped from a depot through transit stations of type $k-1, ..., 1$ to the demand points. We call level l of facilities all the facilities of type l. The transportation costs between demand points and facilities on the first level, as well as transportation costs between facilities on consecutive levels are known and they satisfy the triangle inequality. The goal is to decide which facilities to open and to assign demand points to paths along open facilities such that the total cost (the cost of opening facilities plus the assignment cost) is minimized. For this problem, Aardal, Chudak and Shmoys [ACS99] developed a $3-$approximation based on linear programming rounding. Their algorithm has a large running time, since the number of variables is proportional with the number of paths from demand points to the last level, passing levels $1, ..., k-1$ in this order. As an alternative, we propose a combinatorial algorithm using the primal-dual method. The running time of our algorithm is better in comparison with the algorithm of Aardal, Chudak and Shmoys, but the performance guarantee is larger, namely 6. At the end of the chapter we extend our algorithm to two variants of $MFLP$, to a capacitated variant and to the $MFLP$ with outliers. In the capacitated variant, multiple copies of a each facility i can be opened and each copy can serve at most u_i demand points. The $MFLP$ with outliers, has a similar formulation as the $MFLP$ with the difference that some demand points, the outliers, may remain unassigned to facilities, if a penalty is paid instead.

In Chapter 5 we study facility location problems in the maximization version. In this type of problems, we have given a set of possible facility locations, a set of demand points and the profit obtained by a client if it would be assigned to a facility. The goal is to decide which facilities to open and how to assign demand points to facilities, such that the total profit, i.e., the profit obtained from the assignment minus the cost of opening facilities is maximized. After reviewing the main results for the case where the facilities are situated on only one level, we propose a 0.47 approximation algorithm for the case where the facilities are situated on 2 levels and each demand point has to be assigned to a pair of open facilities, one in each level. The technique we use is randomized rounding.

Bibliography

[Aa92] K. Aardal. *On the solution of one and two level capacitated facility location problems by cutting plane approach, Ph.D. thesis,* Université Catholique de Louvain, Louvain-la-Neuve, Belgium

[ACS99] K. I. Aardal, F. Chudak and D. B. Shmoys. A 3-approximation algorithm for the k-level uncapacitated facility location problem. *Information Processing Letters*, 72, pages 161-167, 1999

[ALLQ96] K. Aardal, M. Labbé, J. Leung, M. Queyranne. On the two-level uncapacitated facility location problem. *INFORMS Journal on Computing, 8,* pages 289-301, 1996

[Ag02] A. Ageev. *Improved approximation algorithms for multilevel facility location problems.* Manuscript

[AB90] A. Ageev and V. Beresnev. Polynomially solvable special cases of the simple plant location problem. In *Proceedings of the First Integer Programming and Combinatorial Optimization Conference*, pages 1-6, 1990

[AS99] A. Ageev, M. Sviridenko. A 0.828-approximation algorithm for the uncapacitated facility location problem, *Discrete Applied Mathematics*, 93, pages 289-296, 1999

[ASE92] N. Alon, J.H. Spencer and P. Erdös. *The probabilistic method*, John Wiley and Sons, New York, 1992

[AZ98] M. Andrews and L. Zhang. The access network design problem. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science,* pages 40-49, 1998

[AL96] S. Arora and C. Lund. Hardness of Approximations. In D. Hochbaum, ed., *Approximation Algorithms for NP-hard problems.* PWS Publishing, Boston, 1996

[AL *et al.* 98] S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of ACM,* 45(3), pages 501-555, 1998

[ARR98] S. Arora, P. Raghavan and S. Rao. Approximation schemes for Euclidean k-medians and related problems. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 106-113, 1998

[AG *et al.* 01] V. Arya, N. Garg, R. Khandekar, V. Pandit, A. Meyerson, K. Munagala. Local search heuristics for k-median and facility location problems. In *Proceedings of the 33rd ACM Symposium on Theory of Computing,* pages 21-29, 2001

[AC *et al.* 99] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela and M. Protasi. *Complexity and Approximation*, Springer, Berlin, 1999

[BR01] I. Baev and R. Rajaraman. Approximation algorithms for data placement in arbitrary networks. In *Proceedings of the 12-th Annual ACM-SIAM Symposium on Discrete Algorithms,* pages 661-670, 2001

[BEW86] I. Bárány, J. Edmonds and L. A. Wolsey. Packing and covering a tree by subtrees. *Combinatorica* 6, pages 245-257, 1986

[Be01] V. L. Beresnev. An efficient algorithm for the uncapacitated facility location problem with totally balanced matrix. *Discrete Applied Mathematics* 114, pages 13-22, 2001

[BCV97] D. Bertsimas, T. Chungpiaw, R. Vohra. On dependent randomized rounding algorithms. *Operations Research Letters*, 21 (3), pages 123-132, 1997

[Bu01] A. F. Bumb. An approximation algorithm for the maximization version of the two level uncapacitated facility location problem. *Operations Research Letters*, 29, pages 155-161, 2001

[BK01] A. F. Bumb and W. Kern A simple dual ascent algorithm for the multilevel facility location problem. In *Proceedings of the 4th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, Springer-Verlag LNCS Vol. 2129, pages 55-63, 2001

[BT97] D. Bertsimas, J. N. Tsitsiklis. *Introduction to Linear Optimization.* Athena Scientific, Belmont, Massachusetts, 1997

[CG99] M. Charikar and S. Guha. Improved combinatorial algorithms for the facility location problem and k-median location problems. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science,* pages 378-388, 1999.

[CKMN01] M. Charikar, S. Khuller, D. Mount and G. Narasimhan. Algorithms for facility location problems with outliers. In *Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms,* pages 642-651, 2001

[C98a] F. Chudak. *Improved Approximation Algorithms for the Uncapacitated Facility Location Problem.* PhD thesis, Cornell University, 1998

[C98b] F. A. Chudak. Improved approximation algorithms for uncapacitated facility location problem. In R. E. Bixby, E. A. Boyd and R. Z. Rios-Mercado, eds., *Integer Programming and Combinatorial Optimization*, Springer-Verlag LNCS Vol. 1412, pages 180-194, 1998

[CS98] F. Chudak and D. B. Shmoys. *Improved approximation algorithms for uncapacitated facility location.* Unpublished Manuscript. 1998

[Ch83] V. Chvátal. *Linear Programming.* W. H. Freeman, New York, NY, 1983

[Co71] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of Third Annual ACM Symposium on Theory of Computing,* pages 151-158, 1971

[CFN77] G. Cornuejols, M. L. Fisher and G. L. Nemhauser. Location of bank accounts to optimize float: an analytic study of exact and approximate algorithms, *Management Science,* 23, pages 789-810, 1977

[CNW90] G. Cornuejols, G. L. Nemhauser and L. A. Wolsey. The uncapacitated facility location problem. In P. Mirchandani and R. Francis, editors, *Discrete Location Theory*, John Wiley and Sons, New York, pages 119-171, 1990

[Er78] D. Erlenkotter. A dual based procedure for uncapacitated facility location. *Operations Research* 26, 992-1009, 1978

[ES73] P. Erdös and J. L. Selfridge. On a combinatorial game. *Journal of Combinatorial Theory*, Series A, 14, pages 298-301, 1973

[Ge74] A. M. Geoffrion. Lagrangian relaxation and integer programming. *Mathematical Programming Study*, 2, pages 82-114, 1974

[GJ84] M. R. Garey and D. S. Johnson *Computers and Intractibility:A Guide to the Theory of NP-Completeness* , Freeman, New York, 1984

[GW94] M. X. Goemans, D. P. Williamson, A new 3/4-approximation algorithm for the Maximum Satisfiability Problem, *SIAM journal on Discrete Mathematics,* 7, pages 656-666, 1994

[GW97] M. X. Goemans, D. P. Williamson, The primal-dual method for approximation algorithms and its applications to network design problems. In D. Hochbaum (Ed.): *Approximation Algorithms for NP-Hard Problems.* PWS Publishing, Boston, 1997

[Gu00] S. Guha, Approximation algorithms for facility location problems, Ph.D. thesis, Stanford, 2000

[GK99] S. Guha and S. Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms* 31 (1), pages 228-248, 1999

[GMM01] S. Guha, A. Meyerson, K. Munagala. Improved algorithms for fault tolerant facility location. In *Proceedings of 12th ACM-SIAM Symposium on Discrete Algorithms,* 2001

[Ho82] D. Hochbaum. Heuristics for the fixed cost median problem. *Mathematical Programming*, 22, pages 148-162, 1982

[JMS01] K. Jain, M. Mahdian, A. Saberi. *A new greedy approach for facility location problem*, Manuscript, 2001

[JV00] K. Jain and V. Vazirani. An approximation algorithm for the fault tolerant facility location problem. In *Proceedings of the 3th International Workshop on Approximation Algorithms for Combinatorial Optimization*, Springer-Verlag LNCS Vol 1913, pages 177-183, 2000

[JV01] K. Jain and V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation. *Journal of the ACM*, 48, pages 274-296, 2001

[KEH77] L. Kaufman, M. van den Eede and P. Hansen. A plant and warehouse location problem. *Operational Research Quarterly*, 28, pages 547-557, 1977

[Ko83] A. Kolen. Solving covering problems and the uncapacitated plant location problem on trees. *European Journal of Operations Research,* 12, pages 266-278, 1983

[KR99] S. Kolliopoulos and S. Rao. A nearly linear-time approximation scheme for the Euclidean k-median problem, In *Proceedings of the 7th Annual European Symposium on Algorithms*, Springer Verlag LNCS vol. 1643, pages 378-389, 1999

[KPR 98] M. R. Korupolu, C. G. Plaxton and R. Rajaraman. Analysis of a local search heuristic for facility location problems. In *Proceedings of the 9-th Annual ACM-SIAM Symposium on Discrete Algorithms,* pages 1-10, 1998

[KB77] J. Krarup and O. Bilde. Plant location, set covering and economic lot sizing: an $O(mn)$ algorithm for structered problems. In L. Collatz *et al.*,editors, *Optimierung bei graphentheoretischen und ganzzahligen Probleme,* pages 155-180, 1977.

[KH63] A. A. Kuehn and M. J. Hamburger. A heuristic program for locating warehouses. *Management science*, 9, pages 643-666, 1963

[LV92] J. H. Lin and J. S. Vitter. ϵ-approximation with minimum packing constraint violation. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 771-782, 1992

[MMSV01] M. Mahdian, E. Markakis, A. Saberi. V.V. Vazirani. A greedy facility location algorithm analyzed using dual fitting. In *Proceedings of the 4th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, Springer-Verlag LNCS Vol. 2129, pages 127-137, 2001

[MYZ02] M. Mahdian, Y. Ye, J. Zhang. *A 1.52 approximation algorithm for the uncapacitated facility location problem*, Manuscript

[Ma64] A. Manne. Plant location and economy of scale decentralization and computation. *Management Science*, 11, pages 213-235, 1964

[MP00] R. R. Mettu and C. G. Plaxton. The online median problem. In *Proceedings of the 41st IEEE Symposium on Foundation of Computer Science,* pages 339-348, 2000

[MMP01] A. Meyerson, K. Munagala and S. Plotkin. Web caching using access statistics. In *Proceedings of the 12-th Annual ACM-SIAM Symposium on Discrete Algorithms,* pages 354-363, 2001

[MMP00] A. Meyerson, K. Munagala, S. Plotkin. Cost distance: Two metric network design. In *Proceedings of the 41th IEEE Symposium on Foundation of Computer Science,* 2000

[MF90] P. Mirchandani and R. Francis, eds. *Discrete location theory.* John Wiley and Sons, Inc., New York, 1990

[MC79] J. M. Mulvey and H. L. Crowder. Cluster Analysis: An application of Lagrangian Relaxation. *Management Science* 25, pages 329-340, 1979

[PY91] C. Papadimitriou and M. Yannakakis. Optimization, approximation and complexity classes. *Journal of Computer and System Sciences* 43, pages 425-440, 1991

[Ra98] P. Raghavan. Probabilistic construction of deterministic algorithms: approximating packing integer programs, *Journal of Computer and Systems Sciences* 37, pages 130-143, 1988

[RT87] R. Raghavan and C. D. Thompson. Randomized rounding, *Combinatorica,* 7, pages 365-374, 1987

[SG76] S. K. Sahni and T. F. Gonzalez. P-complete approximation problems. *Journal of ACM*, 23, pages 555-565, 1976

[Sc87] A. Schrijver. *Theory of linear and integer programming,* John Wiley&Sons, 1987

[STA97] D. Shmoys, E. Tardos and K. Aardal. Approximation algorithms for facility location problems. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 265-274, 1997

[Sp69] K. Spielberg. Algorithms for the simple plant location problem with some side conditions. *Operations Research,* 17, pages 85-111, 1969

[St63] J. Stollsteimer. A working model for plant numbers and locations. *Journal of Farm Economics*, 45, pages 631-645, 1963

[Sv99] M. Sviridenko. Personal communication. Cited in S. Guha, *Approximation algorithms for facility location problems*, PhD thesis, Stanford, 2000

[TL84] D. Tcha and B. Lee. A branch and bound algorithm for the multilevel uncapacitated facility location problem. *European Journal of Operations Research*, 18, pages 35-43, 1984

[Va01] V. V. Vazirani. *Approximation algorithms,* Springer Verlag, Berlin, 2001

[Va02] V. V. Vazirani. Primal-dual schema based approximation algorithms. In: G. B. Khosrovshahi *et al* (Eds.): *Theoretical Aspects of Computer Science* ,Springer Verlag LNCS 2292, pages 198-207, 2002

[Ye91] Y. Ye. An $O\left(n^3L\right)$ potential reduction algorithm for linear programming, *Mathematical Programming*, 50, pages 239-258, 1991

[ZY02] J. Zhang and Y. Ye. *A note on the maximization version of multilevel facility location problems.* Submitted, 2002

About the author

Adriana Bumb was born on June 9, 1974 in Cluj-Napoca, Romania. In 1992 she began to study Mathematics at Babeş- Bolyai University in Cluj-Napoca. She graduated in 1996, with a specialization in functional analysis. One year later, she got a master degree in optimization from the same university.

During the academic year 1997/98 she attended the master class programme in Operations Research organized by the Mathematical Research Institute (MRI) in The Netherlands.

Since 1998 she is a Ph.D. student at the Faculty of Mathematical Sciences of the University of Twente. Her research concerned approximation algorithms for facility location problems and resulted in this thesis.