Universidade Estadual de Campinas
Instituto de Computação

Renata Ghisloti Duarte de Souza

The Geometric Connected Facility Location Problem

Problema Geométrico e Conexo de Localização de Instalações

CAMPINAS
2016

Renata Ghisloti Duarte de Souza

The Geometric Connected Facility Location Problem

Problema Geométrico e Conexo de Localização de Instalações

Dissertação apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Mestra em Ciência da Computação.

Thesis presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Computer Science.

Supervisor/Orientador: Prof. Dr. Flavio Keidi Miyazawa
Co-supervisor/Coorientador: Prof. Dr. Rafael Crivellari Saliba Schouery

Este exemplar corresponde à versão final da Dissertação defendida por Renata Ghisloti Duarte de Souza e orientada pelo Prof. Dr. Flavio Keidi Miyazawa.

CAMPINAS
2016

Agência(s) de fomento e nº(s) de processo(s): CAPES, 1406904; CAPES, 1364124

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Maria Fabiana Bezerra Muller - CRB 8/6162

Souza, Renata Ghisloti Duarte de, 1987-
So89g The geometric connected facility location problem / Renata Ghisloti Duarte de Souza. – Campinas, SP : [s.n.], 2016.

Orientador: Flávio Keidi Miyazawa.
Coorientador: Rafael Crivellari Saliba Schouery.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Algoritmos de aproximação. 2. Teoria da computação. I. Miyazawa, Flávio Keidi,1970-. II. Schouery, Rafael Crivelarri Saliba. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: Problema geométrico conexo de localização de instalações
Palavras-chave em inglês:
Approximation algorithms
Theory of computing
Área de concentração: Ciência da Computação
Titulação: Mestra em Ciência da Computação
Banca examinadora:
Flávio Keidi Miyazawa [Orientador]
Eduardo Candido Xavier
Luis Augusto Angelotti Meira
Data de defesa: 25-11-2016
Programa de Pós-Graduação: Ciência da Computação

Universidade Estadual de Campinas
Instituto de Computação

INSTITUTO DE COMPUTAÇÃO

Renata Ghisloti Duarte de Souza

The Geometric Connected Facility Location Problem

Problema Geométrico e Conexo de Localização de Instalações

Campinas, 25 de novembro de 2016

Acknowledgements

First I would like to express my gratitude to CAPES for the scholarship for this masters and also to my two brilliant advisors Flavio and Rafael, who kindly shared their knowledge with me. Secondly, I want to thank my loving and caring parents Ricardo and Beatris who made this masters possible. I am also very thankful for my sister Fernanda, my brother-in-law Eduardo and my niece Bianca, three big joys in my life. And lastly but most importantly, to my beloved husband and best friend Fernando, without whom there would be no masters at all.

Resumo

Esse trabalho visa estudar problemas da família Localização de Instalações. Nesses problemas, recebemos de entrada um conjunto de clientes e um conjunto de instalações. Queremos encontrar e abrir um subconjunto de instalações, normalmente, pagando um preço por cada instalação aberta. Nosso objetivo é conectar clientes a instalações abertas, pagando o menor custo possível. Esse problema tem grandes aplicações na área de Pesquisa Operacional e Telecomunicações.

Estamos especialmente interessados no problema de Localização de Instalações Geométrico e Conexo. Nessa versão do problema, as instalações podem ser abertas em qualquer lugar de um plano de dimensão d, e pagamos um preço fixo f por cada instalação aberta. Também devemos conectar as instalações entre si formando uma árvore. Essa árvore normalmente recebe uma ponderação maior, uma vez que suas conexões agregam atendimento para quantidade maior de recursos. Para representar tal ponderação seus custos são multiplicados por um parametro $M > 0$ dado como parte da entrada. Apresentamos um Esquema de Aproximação Polinomial para a versão euclidiana do problema. Nosso trabalho foi publicado no journal Theory of Computing Systems.

Abstract

In this work we study problems from the facility location family. In this set of problems, we want to find and open a subset of given facilities. Usually, a price is paid for each opened facility. Our goal is to connect given clients to the closest opened facilities incurring in the smallest cost possible. This problem has several practical applications in Operations Research and Telecommunication.

We are specially interested in the Geometric Connected Facility Location problem. In this version, facilities can be anywhere in the d-dimensional plane, and we have to pay a fixed price f to open each facility. We also have the requirement of connecting facilities among themselves forming a tree. This tree is usually weighted by a given parameter $M > 0$. We present a Polynomial-Time Approximation Scheme for the two-dimensional version of this problem. Our work has been published in the Theory of Computing Systems journal.

List of Figures

Contents

Chapter 1

Introduction

In Theoretical Computer Science we say an algorithm is *efficient* if it can solve a problem with an instance of size n in time $O(p(n))$, for a polynomial function p. A problem whose answer is "yes" or "no" is said to be a *decision problem.* The class of all decision problems that can be solved efficiently is called P *(polynomial time)*. We denote by NP *(non-deterministic polynomial time)* the class of decision problems for which the instances whose answer is "yes" can be verified in polynomial time. It is straightforward to see that $\mathrm{P} \subseteq \mathrm{NP}$. The great open question in the area is if $\mathrm{P} = \mathrm{NP}$.

We say that a problem is NP-hard if it is at least as hard to be solved as any other problem in NP. When a problem is NP-hard and it belongs to the NP class, we call it NP-complete. If the conjecture $\mathrm{P} \neq \mathrm{NP}$ is true, then finding an optimal solution to an NP-hard problem in polynomial time is impossible. Nonetheless, over the years several approaches have been suggested to deal with NP-hard problems. One way to address these problems is through the development of *approximation algorithms.*

1.1 Approximation Algorithms

The goal of *Combinatorial Optimization* problems is to find the optimal (maximum or minimum) value of a function defined over a finite domain [31]. Even though the domain is finite, the naive idea of enumerating all possible results is not practical for most problems, since the domain is usually exponential in the size of the instance. Many of these problems are NP-hard, such as the Traveling Salesman Problem, the Knapsack Problem and others [16].

Approximation algorithms is a sub-area of Combinatorial Optimization, formalized in the 70's by Garey, Graham and Ullman [15] and by Johnson [22]. In this area, we approach NP-hard problems by relaxing the requirement of optimality to gain efficiency. An *approximation algorithm* is an efficient procedure that for every instance of a problem returns a solution whose value is within a guaranteed distance of the value of the optimal solution. Let $\mathrm{OPT}(I)$ denote the value of an optimal solution to an instance (I) of a certain problem. When the instance (I) is clear by the context, we use only OPT to refer to the optimal solution. We say we have an *α-approximation algorithm* for an optimization problem if for any instance (I) it produces in polynomial time a solution whose value is within a factor of α of OPT(I) [39]. As the problems considered in this work are minimization, we will restrict further definitions to such problems. Note that $\alpha \geq 1$ for minimization problems and $\alpha \leq 1$ for maximization problems.

Another important notion in this area is of a *Polynomial Time Approximation Scheme* (PTAS). For a rational number $\varepsilon > 0$, we say an algorithm is a PTAS for a problem, if for every fixed ε, the algorithm produces a solution of value $(1 + \varepsilon)$OPT(I) efficiently, for every instance (I) of the problem. Note that the running time of a PTAS must be polynomial in the size of the instance, but not necessarily on $1/\varepsilon$. When the algorithm is also polynomial in $1/\varepsilon$ we say we have a *Fully Polynomial Time Approximation Scheme* (FPTAS).

In the next section we will introduce the Facility Location problem and other problems relevant to the present work, and discuss our own contribution of a PTAS for the Geometric Connected Facility Location problem.

1.2 The Steiner Tree Problem

One important NP-hard problem is the Steiner Tree problem [24]. Given a set of n input nodes, we want to find a minimum cost tree connecting all nodes. Other nodes, known as *steiner nodes* or *steiner points*, can be introduced to decrease the tree length. The Steiner Tree problem appears in many practical scenarios such as circuits layouts and network design. In such problems, one can be given, for example, a set of pins to be connected using the minimum amount of wire.

When the cost of the tree edges are given by the Euclidian distance we have a Euclidian Steiner Tree problem. A PTAS was proposed to this problem by Arora [3]. The author broke the problem into distinct regions in the plane with the help of an underlying grid. He showed that it is possible to decrease the number of crossings between solution and grid with a small cost increase. This could be done by using a randomized grid. The solution could then be found by dynamic programming.

1.3 Facility Location Problems

The Facility Location problem (FLP) is also a vastly studied NP-hard problem. The first works on it go back to the 60's [6] [27] and it still draws attention until the present day [11]. In this problem, we receive a set of clients, and want to find locations to open facilities, so that each client is connected to the closest open facility. The goal is to minimize the total cost of opening facilities and connecting clients to them. This problem presents several practical applications in Operations Research, Data Mining, Image Processing and Network Design [35]. In Network Design, the applications are straightforward: one can consider the problem as computers to be connected to switches, or even switches to be connected to routers. Nevertheless, to obtain an approximation better than $O(\log(n))$ is not possible unless NP $\subseteq$ DTIME$[n^{O(\log\log n)}]$ [20].

Other variants of the Facility Location problems have been studied over the years. In the Metric Facility Location problem, where distances respect the triangle inequality, a 3-approximation has been proposed by Vazirani et al. [21] using a Primal-Dual schema. When the connecting cost is given by the Euclidian distance we have the Euclidian Facility Location problem. The PTAS [4] proposed by Arora to this problem has a big influence on our work, and it is an extension of the work proposed in [3].

The Geometric Facility Location problem (GFLP) was proposed by Meira and Miyazawa [30], and here the facilities are not given as input and we aim at finding them in the d-dimensional space. A fixed price f is paid to open each facility. This problem also has practical appeal. In Network Design for example, one may choose to set the switches anywhere on the plane instead of pre-fixed locations, aiming at reducing network costs. Meira and Miyazawa proposed a $(3 + \varepsilon)$-approximation to the Euclidian and squared Euclidian versions of the problem.

In the Connected Facility Location problem (CFLP), besides connecting clients to facilities, we also have to connect facilities among themselves. Typically, this tree of connected facilities is scaled by a factor M given as input. In Network Design, the multiplication by the factor M reflects the fact that interconnecting switches (or routers) is typically more expensive than connecting a end client to a switch [7]. All these practical applications and the academic appeal have motivated us to study a new the facility location problem.

1.4 Geometric Connected Facility Location

We are interested in the Geometric Connected FLP (GCFLP). As the name suggests, this problem is derived from the GFLP and the CFLP. In this variant, facilities are not given as input and, additionally, they have to be connected among themselves forming a weighted tree. This problem portraits realistic network scenarios, where switches can be placed in unspecified locations and inter-connecting them usually implies in a more expensive network cost.

Given a set of clients $\mathcal{C} \subset \mathbb{R}^d$, we want to select a set of locations $F \subset \mathbb{R}^d$ where to open facilities, each at a fixed cost $f \geq 0$. For each client $j \in \mathcal{C}$, one has to choose to either connect it to an open facility $\phi(j) \in F$ paying the Euclidean distance between j and $\phi(j)$, or pay a given penalty cost $\pi(j)$. The facilities must also be connected by a tree T, whose cost is $M\ell(T)$, where $M \geq 1$ and $\ell(T)$ is the total Euclidean length of edges in T. The objective is to find a solution with minimum cost.

In this work, we propose a Polynomial-Time Approximation Scheme for the two-dimensional GCFLP. Our algorithm also leads to a PTAS for the two-dimensional Geometric Connected k-medians, when $f = 0$, but only k facilities may be opened. Our work is an extension of a famous technique introduced by Arora on [3] and [2]. In this dissertation, we will cover first the bases for our contribution and then present our own PTAS. On Chapter 2 we will introduce Arora's technique, and present the PTAS for the Euclidian Steiner Tree problem. We will then discuss the Facility Location problem on Chapter 3 as well as some of its variants. In that chapter, we will cover a 3-approximation for the Metric problem and a PTAS for the Euclidian version. This PTAS is also a development of Arora's initial technique, and was of great importance to our own PTAS.

Finally, on Chapter 4 we will discuss our contribution with a PTAS for the Geometric Facility Location problem. This work has been published in the Theory of Computing Systems journal.

Chapter 2

A PTAS for the Euclidian Steiner Tree Problem

The Minimum Steiner Tree problem (ST) has been studied for decades and presents extensive applications in Network Design and VLSI routing [5]. In this problem, given a set of n nodes, our goal is to find a minimum cost tree connecting all nodes. At first glance this problem might seem similar to the well-known Minimum Spanning Tree problem (MST). In the ST however, new nodes can be added to reduce the tree cost, possibly leading to a very different structure than a Minimum Spanning Tree. Furthermore, the ST is NP-hard while the MST can be solved in polynomial time [16]. The Minimum Spanning Tree is known to give a 2-approximation to ST, such as discussed in [26] and [33].

The decision version of the Minimum Steiner Tree problem on graphs was shown to be NP-complete by Karp [24], using a reduction from the boolean satisfiability problem. Bern and Plassman proved in [9] that this problem is MAX-SNP-hard and hence there is no Polynomial-Time Approximation Scheme for it unless P = NP. In this same work, the authors showed that the Metric Steiner Tree Problem, a variation of ST where distances satisfy the triangle inequality, is also MAX-SNP-hard.

For the Euclidian Minimum Steiner Tree problem (EST) a PTAS was proposed by Arora in 1998 [3]. Latter on Arora improved the running time of his algorithm, proposing an almost linear time approximation scheme for the problem [2]. That work introduced a PTAS not only for EST but also for several other geometric problems such as the Traveling Salesman problem (TSP). In his papers, Arora described the method and proved lemmas for the TSP, giving brief details on the applications to the other problems. This chapter intends to expand this explanation and describe Arora's approximation scheme with the Steiner Tree problem perspective.

2.1 Steiner Tree Model and Notation

In the General Steiner Tree problem, we are given a graph $G = (V, E)$, a set of terminal vertices $\mathbb{T}$, with $\mathbb{T} \subseteq V$, and costs on each edge $e \in E$. We need to find a minimum cost subgraph in which any two vertices in $\mathbb{T}$ are connected. Vertices in V outside of $\mathbb{T}$

can be added to decrease the cost of the solution. These other nodes are referred to as *steiner points* or *steiner nodes*.

In the Geometric Steiner Tree problem, we are given a set V of n nodes. The goal is to find a minimum cost tree connecting all n points being that other nodes can also be added to diminish the tree cost. In this variant of the problem, the steiner nodes (V_{sp}) can be found anywhere in the geometric plane.

Let V_{all} be the union of input and steiner nodes, $V_{all} = V \cup V_{sp}$. Let T be a tree that covers all input nodes and steiner points, that is $T = (V_{all}, E)$. We define the length of any edge $e \in E$ as $length(e)$. The goal in the Minimum Geometric Steiner Tree Problem is to find a tree T that minimizes

$$length(T) = \sum_{e \in E_t} length\,(e). \tag{2.1}$$

When the points lie on $\mathbb{R}^d$ and the distances between points satisfy the triangle inequality, we say we have a Metric Steiner Tree problem. In this problem, the cost of a tree is given by the length of its edges, e.g, the sum of distance between any two connected points. The length of an edge e between two points x and y, in a dimension d, in the l_p norm is defined by:

$$length(e) = \left(\sum_{i=1}^{d} |x_i \; - \; y_i|^p \right)^{\frac{1}{p}} \tag{2.2}$$

The reader can observe that when $p = 2$ this is equivalent to the Euclidian distance.

Steiner Trees presents several important properties. A of Metric Steiner Trees used in this work is a upper bound on the maximum number of steiner nodes any tree might have. Suppose we have k steiner nodes and n input nodes in a certain Steiner Tree $T = (V, E)$. As T is a tree it must contain $|V| - 1$ edges, therefore $(k + n) - 1$ edges. As we know, the number of edges $|E|$ is also equal to half the sum of vertices' degree. Note that by definition, a steiner node has degree at least 3 (it doesn't make sense to add steiner nodes with degree 1 or 2). Also, we point out that as T has to be connected, the degree on each one of the n input nodes is at least 1. Hence, the sum of degrees in T is at least $3k + n$. Thus we have that:

$$(n + k) - 1 \geq \frac{3k + n}{2}, \tag{2.3}$$

and thus $n - 2 \geq k$. We can than state without loss of generality that a Steiner Tree with n input nodes has fewer than n steiner nodes.

The Steiner Forest problem is a generalization of the Steiner Tree problem. We are given as input a graph $G = (V, E)$, and sets of vertices $S_1, S_2, ..., S_k \in V$. The objective is to find a minimum cost subgraph in which any two vertices belonging to the same set S_i are connected and the total cost is minimum. Again, steiner nodes might be added to reduce the tree cost for any tree in the forest.

2.2 PTAS Overview

The work developed by Arora presents the first PTAS for the Geometric Euclidian Minimum Steiner Tree problem. He proposed a randomized algorithm that finds a $(1+\frac{1}{c})$-approximation in $O(n\,(\log(n))^{O(c)})$ time, when the instance is in $\mathbb{R}^2$, for every fixed $c > 1$. The algorithm can be adapted to any fixed dimension d incurring in a running time increase to $O(n\,(\log(n))^{(O(\sqrt{d}c))^{d-1}})$. It can also be derandomized with a running time increase of $O(n^d)$.

The algorithm obtains a solution by recursively partitioning the input plane into a randomized grid. Arora proved that for every $\varepsilon > 0$, there exists an approximate solution with value within $(1+\varepsilon)$ from the optimal value that crosses this randomized grid a constant number of times. To be precise, each line of the grid is crossed a number $r = O(c)$ times. This solution also crosses the grid through pre-stablished places called *portals*. The edges of the tree are "bent", so that every time they pass through the border of a square in the grid they do it so using a portal.

The problem is also recursively sub-divided so that almost independent solutions are solved separately in each grid instance. Each square of the grid is treated as an instance of the problem, and can be solved polynomially due to the discretizations described above. These individual solutions are then combined in a bottom up fashion and the best overall solution is found by dynamic programming.

In this chapter, we will focus on Arora's approach for the Euclidian version of the Steiner Tree problem. We will start by introducing some key concepts and discussing how to adapt the input nodes to make them *well-rounded*, so that all coordinates are integers. The core points of Arora's work are presented on the "Structure Theorem For Steiner Tree Problem" Section. There, we present proof of the Patching Lemma, that states one can reduce the number of crossings between a tree and a certain line without increasing its cost too much. Also in this section, we bound the number of crossings between solution and grid by the cost of the solution. We conclude our theorems by proving the Structure Theorem that bounds the entire cost increase of the approximate tree. We then finally examine the dynamic programming algorithm, that actually finds the approximate solution, and its running time.

2.3 General Definitions

As previously mentioned, Arora proposed a PTAS for the Euclidian Steiner Tree problem by recursive partitioning a geometric plane, and then using a randomized version of this partitioning. The algorithm relies on a grid structure to help solving the problem. Let a *bounding box* be a smallest square containing all input points. Let the length of each side of the bounding box be L. We define a *dissection* as a recursive partition of the bounding box in four equal parts. These four new squares, each which side $L/2$, are called *children* of the original square. Each children is also divided in another four equal parts. This partitioning process continues, until the side of the square being divided is smaller than or equal to 1.

This constructed structure can be seen as a 4-ary tree, with the bounding box as tree root. The leaves correspond to the smallest squares in the grid. Observe that since the side of the square is divided by 2 until becomes 1, the depth of this tree is $\log(L)$. Furthermore, as in each level $0 \leq i \leq \log(L)$ of the 4-ary tree we have 4^i nodes, there should be a total of $O(L^2)$ squares on the entire tree.

We will define a *quadtree* as a similar structure. In a quadtree however, the square partitioning process stops either when the square's side being divided is equal to 1 or when there is only one instance point inside the square. As consequence, the quadtree might have fewer squares than a dissection and a quite different format, as shown in Figure 4.1.

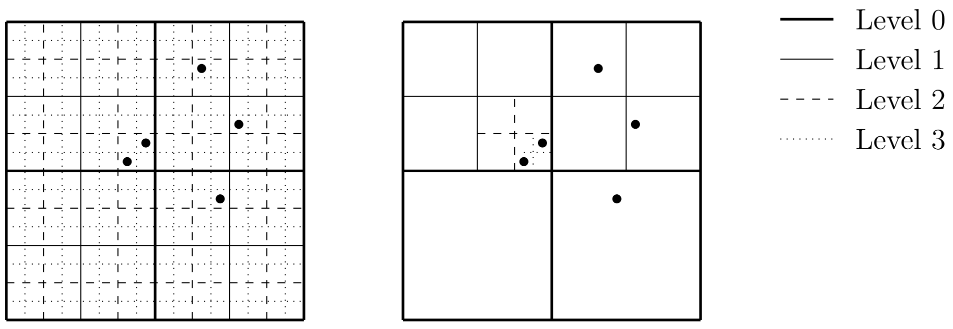

Figure 2.1: The dissection and a correspondent quadtree.

The 4-ary tree representation of the quadtree has $O(n)$ leaves, since every square leaf has either a node or it is sibling of a square containing a node. Next section, we will make sure $L = O(n)$ with the *initial perturbation*. Since the depth of the tree is also $\log(L)$, we have $O(n \log(L))$ squares overall in the quadtree, and so $O(n \log(n))$ squares.

The reader might be wondering why we require both the dissection and quadtree structures. Note that the number of squares in a quadtree is smaller than the number of squares in a dissection. This is why the authors chose to use a quadtree in the algorithm. The fact that it has $O(n \log(n))$ squares guarantees that the running time of the algorithm is $O(n \log(n)^{O(c)})$. The notion of dissection is still important however. Note that the number of vertical and horizontal lines in a quadtree depends on the input, but the number of lines in a dissection is constant and easier to determine. Since the number of lines in a dissection is an upper bound to the lines in a quadtree, the dissection is used to bound the cost increase in the approximation.

An essential notion used in this approximation is of a *random shift*. Let a and b be two integers, where $0 < a, b \leq L$. We define a random *(a,b)-shift* of a dissection or quadtree as a shifting of its vertical lines by a *modulo* L and horizontal lines by b *modulo* L, for randomly chosen values a and b. With such a shift, a vertical line on position p would be displaced to position $p + a \mod L$. Lines that "fall" out of the bounding box are "wrapped-around". Figure 2.2 exemplifies an (a, b)-shift for lines of level 0 on a quadtree.

The idea of random shift is essential to bound the cost increase of the algorithm.

Another important definition is a *level* of a square. The level of a square in a dissection is the level of partitioning. We say that a square has level i if it has been created on the i_{th} partitioning of the grid. The bounding box is defined to have level 0, since no partitioning has yet been made. Its children (first partitioning) have level 1. The children on the second partitioning have level 2 and so on. Note that for two levels j and i, with $j > i$, the squares

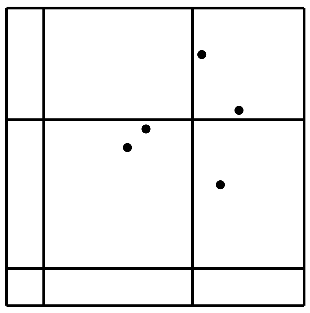

Figure 2.2: A random shift for level 0 of the quadtree.

on level j are smaller then the ones in i. We also use the concept of level for lines in the grid. We say a line l in a dissection has level i, if it contains an edge of some square of level i. Note that a single line might present several levels. We define the *maximal* level of a line i as the level of the greatest square it contains, in other words, the smallest numeric level i, for all levels this line has.

Based on the construction of the dissection, we know that for each level $i \geq 1$ it has 2^i horizontal lines and 2^i vertical lines. The lines of the bounding box are not included to simplify the notation. Let l_h be an horizontal line in an (a, b) random shifted dissection. Since the horizontal shift a was chosen uniformly at random, we have that the probability that l_h is at level i is:

$$Pr[l_h \, is \, at \, level \, i] = \frac{2^i}{L} \tag{2.4}$$

with $i \leq \log(L)$.

The idea that there are 2^i lines in a dissection can also give us information on the side length of a square. As on level i we have 2^i equally spaced lines and the bounding box side size is L, we have that the side of a square on level i has length $L/2^i$.

Let l_h be an horizontal line in an (a, b) random shifted dissection. Let i be the maximal level of line l_h. Since its maximal level is i, l_h is crossed by a vertical line at $b + p \, \frac{L}{2^i}$ mod L for every $p = 1, ..., 2^i - 1$. We define a *segment* of line l_h every fraction of line between two vertical crossings: $b + p \, \frac{L}{2^i}$ mod L and $b + (p + 1) \, \frac{L}{2^i}$ mod L. Note that these segments have size $b + (\frac{L}{2^i})$ mod L. Figure 2.3 exemplifies line segments. The same concepts can be used for horizontal lines in a similar scenario, just considering a shift of a instead of b.

2.4 Initial Perturbation

Before executing the algorithm, some adjustments have to be made on the input to ensure that the algorithm remains polynomial. We will modify the input positions so that:

a) all nodes have integer coordinates;

b) the minimum (non-zero) internode distance is at least 8 units;

c) the bounding box side has size at most $O(n)$.

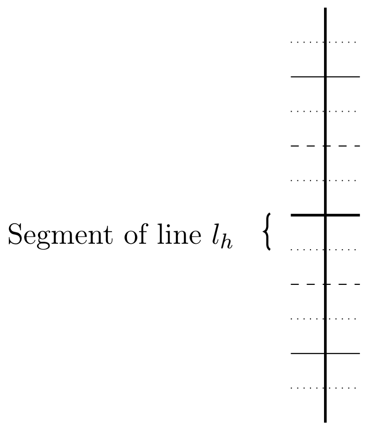

Figure 2.3: Segments of line l_h in a dissection.

We call an instance satisfying these properties *well-rounded*. We will show how to obtain such an instance with a small cost increase.

Recall that L is the side length of the bounding box. Since the bounding box is the smallest square containing all input points, and the points have to be connected by the solution, there are two points in it with distance at least L from each other. Therefore we have that $L \leq \text{OPT}$.

To make the instance well-rounded, we will place a grid of granularity $L/16nc$ on top of it, and move each node to the nearest gridpoint. Note that, with this change the minimum internode distance can be zero since two nodes could collapse to the same point. In the worst case scenario, each point will be moved a distance equivalent to the half of the diagonal of the grid square, d. Figure 2.4 shows a node being moved from the center of a grid square to its corner.

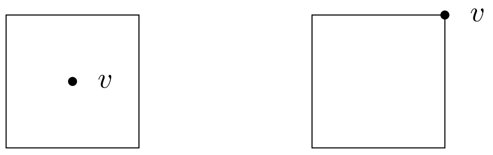

Figure 2.4: a) Node in the middle of a square. b) Node on the corner of a square.

Since $d/2$ is smaller than $L/16nc$ (the side of the square), the additional cost incurred by moving all n input points is bounded by $n\,L/16nc$.

As already discussed, in a minimum cost Steiner Tree with n input points, the maximum number of steiner nodes is n. In our dynamic programming, we will require that steiner nodes also lie on the grid and are well-rounded. We can apply the same perturbation described above to the steiner nodes. Thus, we can consider the cost incurred by moving both instance and steiner nodes to gridpoints is at most $2n\,L/16nc$, and since $L \leq \text{OPT}$, it is at most $\text{OPT}/8c$.

Now, we want to make sure that the minimum internode distance is 8. Recall that the minimum internode distance in the grid is $L/16nc$. We then need to divide all distances by $L/128nc$. With that our unit distance becomes at least $L\,128nc/16nc\,L$, and therefore 8. We just showed how to have all instances in integer coordinates and with minimum internode distance 8.

Finally, since the bounding box side length of our perturbed instance L_p is smaller than $c\,\mathrm{OPT}$ for a constant c, and $n \leq \mathrm{OPT}$, we have that $L_p = O(n)$ and our instance is well-rounded.

2.5 Structure Theorem For Steiner Tree Problem

As already discussed, the PTAS for the EST was obtained by showing that an optimal solution could be modified with a $(1/c)$OPT increase in the cost, with $c > 1$. To achieve such PTAS, the authors demonstrated that this cost increase is bounded by a fraction of the amount of times the optimal solution crosses the lines of the underlying dissection. Furthermore, it was shown that this number of crossings is proportional to the value of an optimal solution OPT. Therefore, the total cost increase is proportional to a fraction of OPT.

The authors use the notion of *bending* solution edges. Every time a bent edge crosses the boundaries of a square, it does so through a pre-specified place on the plane called *portal*. The cost of bending an edge e is associated with a line of the grid l, where e is crossed by l. We can say that we *charge* a cost over l within each modification. This charging however is only computed for l on its maximal level. Even further, the solution is made so that each line on the grid is only crossed a limited r number of times, again without much cost increase.

The polynomial running time of the algorithm is guaranteed by these ideas. With the plane “discretized” with help of portals and a constant number of crossings the best solution can be found by dynamic programming. We will iterate through each square on the portal trying all feasible possibilities and store results in a lookup table.

We begin to formalize these concepts. Let m and r be integers greater than zero. Let m portals be placed on each square edge, and 4 portals placed on its corners, leading to a total of $4m + 4$ portals per square. A solution is defined as (m, r)-light if it crosses each edge of any square of a shifted dissection at most r times, and only through portals.

Figure 2.5 bellow exemplifies the placing of m portals on the square of a dissection with a (m, r)-light path between vertices u and v.

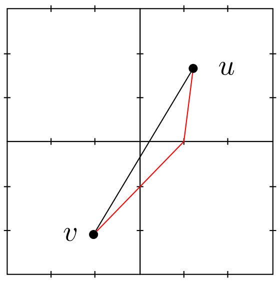

Figure 2.5: Bending edge (v, u) to make it (m, r)-light.

With this definition, we can now quote the Structure theorem for the Steiner Tree from [3]:

Theorem 2.5.1. *Let $c \geq 0$ be any constant. The following is true for every instance of Steiner tree in which the minimum (nonzero) distance between any pair of nodes (including*

Steiner nodes) is 8 *and the size of the bounding box is* L. *Let shifts* $0 \leq a, b < L$ *be picked randomly. Then with probability at least* $1/2$, *the dissection with shift* (a, b) *has an associated Steiner tree of cost at most* $(1 + \frac{1}{c})\text{OPT}$ *that is (m,r)-light, where* $m = O(c \log(L))$ *and* $r = O(c)$.

To prove this theorem, we will require two lemmas. Starting from an optimal Steiner Tree, we will modify the solution using as base Lemma 2.5.2, so that it becomes (m, r)-light. The process of creating an (m, r)-light Seiner Tree will be described in the *ModifySteinerTree* procedure. We will then use Lemma 2.5.3 to bound the cost increase generated by this process.

In the following lemma, also known as the *Patching Lemma*, we state that it is possible to change the edges of a certain solution π of the Steiner Tree problem, without adding too much cost to it, so that these solution edges will only cross each line of its underlying grid at most 2 times. Let l be any line in a grid. Let π be a Steiner Tree on this grid. We define $t(\pi, l)$ as the number of times π crosses l.

Lemma 2.5.2 (Patching Lemma for Steiner Tree). *There is a constant* $g > 0$ *such that the following is true. Let* S *be any line segment of length* s *and* π *be a tree that crosses* S *at least thrice. Then there exist line segments on* S *whose total length is at most* $g.s$ *and whose addition to* π *changes it into a tree* π' *that crosses* S *at most twice.*

Proof. To diminish the crossing points between π and S, we will substitute all $t(\pi, S)$ but two crossings by a straight line accompanying S. We will connect the $t(\pi, S) - 2$ crossing points on each side of S, forming two straight lines parallel to S. Each connecting points can be seen as a steiner node added. Note that we have to be careful to to add any cycles while performing these steps. Figure 2.6 illustrates this process. The lines added to the solution are represented in red. They are curved to better illustrate the process.

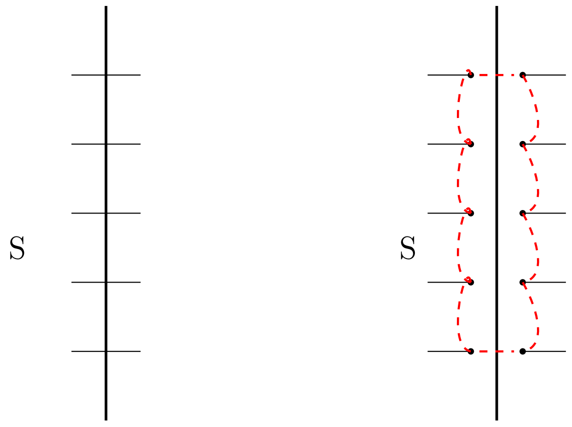

Figure 2.6: Patching Lemma.

Note that each line added is at most the length of S. This will reduce the crossings from $t(\pi, S)$ to 2, and add a cost of at most two times the length of S to the tree π. □

The next lemma will be used on the algorithm analyses. The idea behind it is to relate the cost of a Steiner Tree solution to the number of crossings between solution and underlying grid.

Lemma 2.5.3. *If the minimum internode distance is at least* 4, *and* T *is the length of* π, *then*

$$\sum_{l:vertical} t(\pi, l) + \sum_{l:horizontal} t(\pi, l) \leq 2T. \tag{2.5}$$

Proof. Let e be and edge of the solution π to the Steiner Tree problem. Let s_e be the length of edge e. Let u be the vertical projection of e, and v be its horizontal projection. If e is placed entirely upon u, then e crosses at most u lines of the grid. The equivalent is true if e is equal to its horizontal projection v. In the worst case scenario, e is a bisectrix between u and v and it crosses at most $(u+1)+(v+1)$ lines of the grid. As we know that $u^2 + v^2 = s_e^2$, we have that

$$(u+1)+(v+1) = (u+v+2) \leq \sqrt{2(u^2+v^2)} + 2 \leq \sqrt{2s_e^2} \tag{2.6}$$

Since we stablished that $s \geq 4$ (the minimum internode distance is 4), we have

$$\sqrt{2s_e^2} \leq 2s_e \tag{2.7}$$

ending our proof.

$\square$

With these two lemmas we can finish our proof of the Structure Theorem.

Proof of Theorem 2.5.1. To prove this Theorem we will use a iterative procedure that applies the Patching Lemma on each iteration. The procedure ModifySteinerTree will apply Lemma 2 bottom-up, from the smallest to the biggest squares, reducing the crossings between the solution and the grid. With that, for each level i it will ensure, for all levels $j > i$, we already have an (m, r)-light solution. Finally, we will estimate the cost added in a random fashion, using as base Lemma 2.5.3.

Let l be a vertical line in the grid, b be the vertical random shift for the dissection and i be the maximal level of line l. Let s be $12gc$, where g is the constant appearing in the Patching Lemma. We will describe the procedure ModifySteinerTree(l, i, b) and bound its cost increase for vertical lines. The same steps can be latter reproduced to demonstrate the application of the procedure on horizontal lines.

```
1 for $j = \log(L)$ down to $i$ do
2     for $p = 0, 1, ..., 2j - 1$ do
3         if segment of $l$ between the y-coordinates $(b + p\,\frac{L}{2^j} \mod L)$
          and $(b + (p+1)\,\frac{L}{2^j} \mod L)$ is crossed by the current Steiner Tree more
          than $s$ times then
4             Use the Patching Lemma to reduce the number of crossings to 4.
5         end
6     end
7 end
```

Procedure ModifySteinerTree(l, i, b)

Note that the number of crossings is being reduced to 4, instead of 2 as mentioned on the Patching Lemma. This difference comes from the fact that perhaps the line we are considering in the modify procedure could be wrapped-around in the plane, meaning that it is both the last line of the dissection and the first. For that, we have to account 2 crossings for each part of this line, which leads to a total of 4 crossings. So we bound the total reduced number of crossings to 4.

We defined on Lemma 2.5.3 that an optimal tree π crosses a line l on the grid $t(\pi, l)$ times. Note that we always apply the Patching Lemma to a segment of l that has more than s crossings and substitute these for 4 crossings. Therefore, for every line segment to which we apply it, we remove at least $s-3$ crossings. Let $c_{l,j}(b)$ be the number of segments of l to which we apply the Patching Lemma at a iteration at level j of ModifySteinerTree. Since the number of crossings removed is bounded by the total number of crossings $t(\pi, l)$, we can state that for all level $j > 1$ we have

$$\sum_{j \geq 1} c_{l,j}(b)\,(s-3) \leq t(\pi, l). \tag{2.8}$$

and therefore,

$$\sum_{j \geq 1} c_{l,j}(b) \leq \frac{t(\pi, l)}{s-3}. \tag{2.9}$$

As now we have the amount of segments being updated, and the cost of each update (given by the Patching Lemma) we can estimate the total cost increase of ModifySteinerTree. The Patching Lemma stablished that the cost increase would be a constant g times the size of the segment being crossed. Therefore we have the cost increase $cost(l, i, b)$ of ModifySteinerTree for an instance (l, i, b) is

$$cost(l, i, b) = \sum_{j \geq 1} c_{l,j}(b)\, g\, \frac{L}{2^j}. \tag{2.10}$$

where $L/2^j$ is the maximum size any segment on level j might have.

For each line l, this charging is only going to happen when i is the maximal level of l. We have already seen that for a level i, the probability of it being the maximal level of l is $2^i/L$ over the choice of horizontal shift a. We will now bound the total cost increase for all vertical lines. As already mentioned, we can use the probabilities defined for a dissection here to bound the cost increase for this procedure. This is true since a dissection might have more lines than a quadtree and so the cost incurred by it can be only equal or greater. For every vertical line l, and every random shift b, with $0 \leq b \leq L-1$, the expectation of charging l when the horizontal shift is a is

$$\begin{aligned}
E_a[\mathit{charging}\ l] &= \sum_{i\geq 1} \frac{2^i}{L}\, \mathit{cost}(l,i,b)\\
&\leq \sum_{i\geq 1} \frac{2^i}{L} \sum_{j\geq 1} c_{l,j}(b)\, g\, \frac{L}{2^j}\\
&= g \sum_{i\geq 1} 2^i \sum_{j\geq 1} \frac{c_{l,j}(b)}{2^j}\\
&\leq \frac{2gt(\pi,l)}{s-3}
\end{aligned}$$

Now, we compute the cost increase due to bending the edges and forcing the solution to cross squares on the nearest portals. If a square s with side $\frac{L}{2^i}$ has m portals, then the inter-portal distance must be $L/2^i\, m$. Instead of actually deflecting the solution edges, we will amplify the edge, adding two line segments from the crossing place to the nearest portal. Since any crossing is at most $L/2^{i+1}\, m$ distance of any portal, and the solution will travel this distance twice, the length increase is at most $L/2^i\, m$ for each crossing. Being $\frac{2^i}{L}$ the probability of line l being on level i of the dissection, $t(\pi,l)$ the number of crossings the solution π has on line l, we have that the total expected cost added due to deflecting edges for line l is:

$$\sum_{i=1}^{\log(L)} \frac{2^i}{L}\, t(\pi,l)\, \frac{L}{2^i m} = \frac{t(\pi,l)\, \log(L)}{m}. \tag{2.11}$$

If we consider $m \geq 2s\log(L)$, we obtain an upper bound of $t(\pi,l)/2s$ to the above equation.

Combining the deflecting cost of line l with the previously calculated cost of diminishing l's crossings gives a total cost increase of:

$$\frac{2gt(\pi,l)}{s-3} + \frac{t(\pi,l)}{2s} \leq \frac{3gt(\pi,l)}{s}, \tag{2.12}$$

for each vertical line l on the dissection.

As already mentioned, the same proof is also valid for horizontal lines. We can apply linearity of expectations to find the total value for all horizontal and vertical lines on the dissection. It follows that the total expected cost increase to our Steiner Tree solution is:

$$\sum_{l:\mathit{vertical}} \frac{3gt(\pi,l)}{s} + \sum_{l:\mathit{horizontal}} \frac{3gt(\pi,l)}{s}. \tag{2.13}$$

By Lemma 2.5.3, we have an upper bound of $6g\,\mathrm{OPT}/s$.

Since we defined $s \geq 12gc$ the cost increase can be stated as at most $\mathrm{OPT}/2c$. Finally, we use Markov's Inequality to have a final bound. Remember that Markov's Inequality states that for a random variable X, and a value $a > 0$,

$$P(X > a) \leq \frac{E[x]}{a} \tag{2.14}$$

Therefore, with probability 1/2 the cost increase due to the construction of an (m,r)-light solution is at most OPT/c. This result concludes our Structure Theorem proof.

□

2.6 Dynamic Programming Algorithm

In this section we will demonstrate how to use dynamic programming to find an optimal (m, r)-light Steiner Tree. Let S be a square on the shifted quadtree. We know that an optimal (m, r)-light solution crosses each edge of S at most r times. Let B be a multiset containing all portals crossed for the square S in an (m, r)-light solution. We can partition B in p sets, with $1 \leq p \leq 4r$, and each set B_i is an (m, r)-light Steiner Tree in S. Our goal in the dynamic programming is to find an optimal (m, r)-light Steiner Forest, containing all p trees. An instance of this problem is:

a) A square on the shifted quadtree,

b) a multiset B containing at most r portals for each side of the square,

c) a partition (B_1, B_2, ..., B_p) of B,

where the i^{th} tree contains all portals in set B_i, and together the p trees visit all nodes inside of S. Our goal is to find a solution to this problem with minimum cost. A small modification has to be made on the last level (bounding box) of the dynamic programming. We will require the tree to be connected and set $p = 1$, resulting in one final Steiner Tree.

Note that even if a square does not contain any instance nodes, it still has to be evaluated in the case of having a Steiner Node. The algorithm "guesses" which square might contain Steiner Nodes. To do so, since a square can only be entered $4r$ times, we can treat an empty square as a possibly Steiner Node, by considering it a sub-instance of the Minimum Steiner Tree problem with at most $4r$ nodes. In this sub-problem, the cost of pairing all $4r$ nodes to each portal will be computed, and we will therefore obtain the cost of connecting a Steiner Node to all possible $4m$ portals.

The dynamic programming algorithm works bottom-up, constructing a look-up table with all possible solutions to each square in the quadtree. Our base case are the leaves. Each leaf has at most one node, guaranteed by the initial perturbation. We will select $p \leq 4r$ portals, and try all p ways of placing the node in each of the p trees. Now let S be a square at level i of the quadtree. Suppose inductively that all squares for levels greater than i are already solved. Let S_1, S_2, S_3 and S_4 be S's four children in the quadtree. By induction, they have already been solved. The algorithm now has to combine its children's solution with the smallest cost possible. For each possibility for b) and c) on the instance problem of S, the algorithm will enumerate all (m, r)-light crossings combination for S_1, S_2, S_3 and S_4. Note that this has to be done only for the four inner edges of S_1, ..., S_4, since for the outer edges also belong to S, and will be computed next.

When we reach the root level (bounding box), we set $p = 0$, and let the goal be finding an optimal (m, r)-light Steiner Tree connecting all nodes in the square.

2.7 Running time

To calculate the algorithm running time, we have to consider the steps of the dynamic programming procedure. First, lets consider T, the number of squares in the quadtree. As already discussed, the number of squares on a shifted quadtree is $O(n \log(L))$, where L is the quadtree side length. Since the bounding box has size $O(n)$, we have that

$$T = O(n \log(n)).$$

We are going to analyze the cases on a bottom-up fashion. The leaves of the tree correspond to the base case and have at most one node. They can be solved optimally in $O(r)$ time by trying all r ways of placing the node in the $O(r)$ trees.

For the induction step suppose we have a square S on level i and all squares at level greater than i have already been solved. First lets estimate the cost in finding the best set of portals for S. In this scenario, we have to chose r multiset of portals for each side of the square. Knowing that each side has $m+2$ portals, we have $(m+2)^{4r}$ multiset of portals that can be chosen. From these, we will select $4r$ portals, being that there are $4r!$ ways of ordering them. We obtain therefore $(m+2)^{4r}\,(4r!)$ combinations to try for our dynamic programming instance S.

Now lets considers the work involved in combining the best solution for S's children. We need to enumerate and combine all S_1, ..., S_4 solutions to find the best solution for S.

For every choice of multiset of portals $\leq 4r$ for S, and for every partition $(B_1, B_2, ..., B_p)$ of B, as described in b) and in c), we will enumerate all possible ways that a set of (m,r)-light trees could cross the edges of its children S_1, ..., S_4. Remember we have $m+2$ portals in each edge, and we can only cross each one of the four inner edges r times. Hence, to enumerate a multiset of r trees in each inner edge it takes $((m+2)^r)^4$ steps. We will choose $4r$ portals from these. Again, there are $4r!$ ways of ordering these selected portals. But for each one of the $4r$ portals, we still have to choose one of the at most $4r$ trees in which it lies (from the previous iteration). We have then $4r^{4r}4r!$ ways of ordering the chosen portals so that they are traversed by an optimal (m,r)-tree.

Finally, combining all these costs together we have

$$\begin{aligned} &O(T\,((m+2)^{4r}\,(4r!))\,((m+2)^{4r}\,4r^{4r}4r!)) && (2.15)\\ &= O(T\,(m+2)^{8r}\,4r^{4r}\,(4r!)^2), && (2.16)\end{aligned}$$

which is $O(n(\log(n))^{O(c)})$, since $m = O(\log(n))$ and r is a constant.

This concludes the demonstration of the polynomial running time from this approximation.

Chapter 3

Facility Location

The Facility Location problem (FLP) arises in many practical applications in Operations Research and Network Design [35]. Suppose we have a set of cities that need to be supplied by storage establishments. Our goal is to find and open establishments, so that each city is connected to the closest establishment, paying the minimum cost possible.

Despite its apparently simplicity, the Facility Location problem is NP-hard, that is, it cannot be solved in polinomial time unless P = NP, as proved with a reduction from the Set Cover [20]. In this same work, Hochbaum presented an $O(\log(n))$-approximation for the problem, and this result is tight unless NP $\subseteq$ DTIME$[n^{O(\log\log n)}]$. The FLP has been studied since the 60's [6] [27] and continues to draw attention to this day [11].

In the most common version of the problem, the set of facility locations is some finite set given as input, and the connection cost is calculated by using a given distance function. Even in the case when the set of facilities is a subset of the set of clients the problem is still NP-hard [1].

An important special case that appears in many applications occurs when distances are constrained to metric spaces. The Metric FLP is a variant where the connecting costs respect the triangle inequality, while in the Euclidian FLP this cost is given by the Euclidian distance. The Connected Facility Location problem has great applications in telecommunication [19] [36]. Usually, the set of facilities represents network switches, which are required to be connected by a tree. This tree is scaled by a factor $M \geq 1$ to reflect the typically more expensive cost of interconnecting switches.

In the Geometric FLP [30], or Continuous FLP, facilities have to be found and opened in the d-dimensional space, and the connection cost is also given by the Euclidean distance between each client and assigned facility. Note that opposed to the Euclidian FLP, here the facilities are not given as input. The *prize-collecting* GFLP variant, when one may choose between serving a client or paying an associated penalty, is also often considered [18] [8].

In this chapter, we will study different variants of the Facility Location problem. We will discuss approximation methods for them, including some that were fundamental on the designing of our own PTAS for the Geometric Connected Facility Location problem.

3.1 Facility Location on Graphs

We will define the problem formally. We are given a set of clients C and a set of facilities F. Let f_i be the cost of opening a facility i and c_{ij} the cost of edge (i,j), for $i \in F$ and $j \in C$, where $c_{ij} \geq 0$ and $f_i \geq 0$. The objective is to open a subset of facilities $I \subseteq F$, and connect all clients to open facilities incurring the minimum cost possible.

This problem can be formulated in Integer Linear Programming as follows:

$$
\begin{aligned}
\min \sum_{i\in F}\sum_{j\in C} c_{ij}x_{ij} + \sum_{i\in F} f_i y_i & & & \quad (3.1)\\
\text{subject to } \sum_{i\in F} x_{ij} \geq 1 & & j \in C, & \quad (3.2)\\
x_{ij} \leq y_i & & i \in F, j \in C, & \quad (3.3)\\
x_{ij} \in \{0,1\} & & i \in F,\ j \in C, & \quad (3.4)\\
y_i \in \{0,1\} & & i \in F. & \quad (3.5)
\end{aligned}
$$

The first portion ($\sum_{i\in F}\sum_{j\in C} c_{ij}x_{ij}$) of the objective function is usually called *connecting cost* and the second portion ($\sum_{i\in F} f_i y_i$) is known as *opening cost* of the problem. Restriction (3.2) guarantees that all clients are connected to at least one facility. Restriction (3.3) makes sure facilities which are connected to clients are open. Restrictions (3.4) and (3.5) maintain the integrality of the variables.

Observe that if the opening cost is much smaller than the cost of connecting clients to facilities, we would open as much facilities as possible and connect clients to them. On the other hand, if the opening cost is much greater than the connecting cost, we would open as few facilities as possible.

3.2 Metric Facility Location

We say that a distance function c respects the triangle inequality if for any three nodes i_1, i_2 and i_3 it is true that $c(i_1,i_2) \leq c(i_1,i_3) + c(i_3,i_2)$. When the costs of connecting clients to facilities satisfy the triangle inequality we say we have a Metric Facility Location problem (MFLP).

Several approximation algorithms have been proposed to solve this problem. Let OPT be the cost of a MFLP optimal solution. The first constant approximation was proposed by Shmoys et al [35], with an approximation ratio of 3.16. Lin and Vitter [29] provided a well-known filtering approximation to the problem. Years latter, it was proved that the inapproximability limit for the MFLP is 1.463 unless P = NP [38]. Shi Li almost closed the approximation gap in 2011, presenting a 1.488-approximation to this problem [28].

In 1999, Vazirani et al. [21] proposed a famous greedy algorithm using a $primal - dual\ schema$ approximation. This technique has been vastly used on the development of exact algorithms, such as network flows and shortest-path, and approximation algorithms, usually providing good running times. The primal-dual schema comes from the *Strong Duality theorem* [37]. Having a primal linear minimization problem such as

$$\min \sum_{j=1}^{n} c_j x_j \tag{3.6}$$

$$\text{subject to} \sum_{j=1}^{n} a_{ij} x_j \geq b_i, \qquad i = 1, ..., m \tag{3.7}$$

$$x_j \geq 0 \qquad j = 1, ..., n \tag{3.8}$$

where a_{ij}, b_i and c_j are given rational numbers, and a dual maximization problem

$$\max \sum_{i=1}^{m} b_i y_i \tag{3.9}$$

$$\text{subject to} \sum_{i=1}^{m} a_{ij} y_i \leq c_j, \qquad j = 1, ..., n \tag{3.10}$$

$$y_i \geq 0 \qquad i = 1, ..., m, \tag{3.11}$$

the theorem states that a primal problem has an optimal solution if and only if its dual also has an optimal solution. Furthermore, the theorem states that the value of the optimal dual solution and optimal primal solution is the same. Even further, the value of any feasible solution to the dual is a lower bound to any feasible solution to the primal problem (including the optimal solution). One consequence of this theorem is the *Complementary Slackness Conditions* (CSC). The CSC is a foundation for the primal-dual schema.

Theorem 3.2.1 (Complementary Slackness Conditions)**.** *Let x and y be primal and dual feasible solutions respectively. Then, x and y are both optimal iff 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 primal-dual technique usually consists on iteratively constructing an integer solution for the primal problem, while generating a feasible solution for the dual. It is frequent that the algorithms start with a trivial solution for the primal, a null solution for example. It iteratively increases the primal variables with integral values to maintain the integrality of the system, while satisfying the Complementary Slackness Conditions. The dual's solution is consequently also updated. The result is given based on the comparison between these two solutions, since the dual is a bound to the optimal primal solution.

The primal-dual schema for approximate algorithms follows basically the same steps, but using the *Relaxed Complementary Slackness Conditions* (RCSC) described next.

Theorem 3.2.2 (Relaxed Complementary Slackness Conditions)**.** *Let x and y be primal and dual feasible solutions respectively. Then, x and y respect*

$$\sum_{j=1}^{n} c_j x_j \leq \gamma\, \delta \sum_{i=1}^{m} b_i y_i$$

if the following conditions are satisfied:

Relaxed Primal complementary slackness conditions
Let $\gamma \geq 1$.
For each $1 \leq j \leq n$: *either* $x_j = 0$ *or* $\frac{c_j}{\gamma} \leq \sum_{i=1}^{m} a_{ij} y_i \leq c_j$;

Relaxed Dual complementary slackness conditions
Let $\delta \geq 1$.
For each $1 \leq i \leq m$: *either* $y_i = 0$ *or* $b_i \leq \sum_{j=1}^{n} a_{ij} x_j \leq \delta\, b_i$.

Most approximation algorithms ensure one set of conditions are satisfied and relaxes the other. When the primal conditions are ensured we set $\gamma = 1$ and when the dual conditions are ensured we set $\delta = 1$.

As already mentioned, Vazirani et al. [21] used the primal-dual schema to provide an approximate solution to the Metric Facility Location problem. The authors started with a dual feasible solution. The algorithm increases the value of the dual variables until the RCSC are satisfied. The idea behind the algorithm is that the dual variables will "pay" for the costs of the primal solution. Let the primal formulation for MFLP be the one presented in (1), with the connecting cost respecting the triangle inequality. The relaxed dual formulation for this problem is described below:

$$\max \sum_{j \in C} \alpha_j \tag{3.12}$$

$$\text{subject to} \quad \alpha_j - \beta_{ij} \leq c_{ij}, \qquad i \in F, j \in C \tag{3.13}$$

$$\sum_{j \in C} \beta_{ij} \leq f_i \qquad i \in F, \tag{3.14}$$

$$\alpha_j \geq 0 \qquad j \in C, \tag{3.15}$$

$$\beta_{ij} \geq 0 \qquad i \in F, j \in C. \tag{3.16}$$

In the algorithm, the dual variables α can be understood as the total cost paid by clients in the MFLP, while the β variables pay for the opening cost of facilities. This notion comes from the complementary slackness conditions:

1) $\forall i \in F, j \in C : x_{ij} > 0 \Rightarrow \alpha_j - \beta_{ij} = c_{ij}$,

2) $\forall i \in F : y_i > 0 \Rightarrow \sum_{j \in C} \beta_{ij} = f_i$,

3) $\forall j \in C : \alpha_j > 0 \Rightarrow \sum_{i \in F} x_{ij} = 1$,

4) $\forall i \in F, j \in C : \beta_{ij} > 0 \Rightarrow y_i = x_{ij}$.

If $x_{ij} > 0$ we know that client j is connected to facility i. By condition 1, we can see that if $x_{ij} > 0$, α_j is at least c_{ij}. This means that when client j is connected to facility i, α_j represents at least the connecting cost paid. By condition 2, we can see that when a facility i is open, the sum of β_{ij} for all clients must be equal to the cost of opening i. Also, $\beta_{ij} > 0$ implies that $y_i = x_{ij}$. Now, if facility i is open but client j is not connected to it, $y_i \neq x_{ij}$ and therefore $\beta_{ij} = 0$ and client j is not paying to open facility i. Conditions 2 and 4 put together show that variables β pay for the opening cost of facilities. Thus, since $\alpha_j = c_{ij} + \beta_{ij}$, α_j pays not only the connection cost but all cost paid by client j.

Initially, the algorithm starts with a trivial null solution for the dual variables. The values of these dual variables will be incremented iteratively, and the primal variables will change to maintain feasibility or satisfy the RCSC.

In the beginning every client said to be *unconnected*. The algorithm starts by increasing, for each client j, the value of α_j until it is *connected* to any facility. In this process, α_j can be seen as a radius paid by client j to connect to any facility. Once $\alpha_j = c_{ij}$, we interpret that client j has "reached" facility i. Once this happens, we also start increasing the value of β_{ij}, so that client j starts to pay to open facility i. Note that increasing β_{ij}, we are also respecting $\alpha_j - \beta_{ij} = c_{ij}$.

When $\sum_j \beta_{ij} = f_i$ for some facility i, we say that it is paid and *temporarily* open. Every client j that had $\alpha_j = c_{ij}$ is connected to this facility (and its α_j stops increasing). We say facility i is temporarily open because at the end of this process, one client might have paid to open several facilities. To address this, the algorithm then uses a procedure to *permanently* open only some subset of temporarily open facilities. This is done by getting a maximal independent set of facilities from a modified graph of the clients and temporarily open facilities. The clients are then connected to the permanently open facilities and the procedure terminates.

This algorithm guarantees a 3-approximation for MFLP. There is no PTAS for this problem unless P = NP [38]. Next section we will discuss a PTAS for another Facility Location variation.

3.3 Euclidian Facility Location

The Euclidian Facility Location problem (EFLP) is a particular case of the Metric Facility Location in which clients and facilities are in the $\mathbb{R}^d$. In this problem the connecting costs in the graph are given by the Euclidian distance. For a dimension d, the Euclidian distance dist() between two points x and y is defined as

$$\operatorname{dist}(x, y) = \left(\sum_{i=1}^{d} (x_i - y_i)^2 \right)^{\frac{1}{2}}. \tag{3.17}$$

Arora et al. [4] proposed a randomized PTAS for the Euclidian Facility Location problem. It was made as an extension of the work developed in a previous approximation scheme for the TSP and other geometric problems, described in Chapter 2. Given an instance to this problem, let OPT be the value of an optimal EFLP solution. The algorithm finds a solution with value $(1 + \frac{1}{c})$OPT in $n^{O(c+1)}$ time, with $1 - o(1)$ probability. The instance points are considered to be laying on a grid with a random shift.

In this section, we will look in deeper details the ideas behind this approximation algorithm to EFLP.

3.3.1 Introduction

In [4] Arora et al. proposed the first PTAS for the Euclidian Facility Location problem on the plane and the first PTAS (and constant approximation) for the Euclidian k-Median problem on the plane.

The authors defined the EFLP as, given a set C of n input points in $\mathbb{R}^2$, a Euclidian distance function dist and a cost f_j for opening a facility at input point x_j, find a subset of facilities F that minimizes

$$\sum_{x_j \in F} f_i + \sum_{i=1}^{n} \min_{x_j \in F} \{\text{dist}(x_i, x_j)\}. \tag{3.18}$$

Note that in this definition, F is a subset of C. Nevertheless, this problem is as hard to be approximated than the regular Euclidian FLP [1].

Recall from Chapter 2 that on this first work Arora used a method to reduce the number of crossings between solution and underlying grid known as the Patching Lemma. For a solution of the Steiner Tree problem for example, this lemma says that we can remove all but two ST crossings over a line of a grid, adding a cost proportional to the size of the grid line crossed. This could be achieved by adding two edges, parallel to this grid line, connecting all crossing points, as it was shown on Figure 2.6. This lemma was an important step on the dynamic programming and analysis of the algorithm, since it provided tools for arriving in an approximate solution in polynomial time.

The Patching Lemma however does not apply to the EFLP (nor to the k-Median), since for this problem it is not possible to reduce the number of crossings between the solution and grid lines without possible increasing its cost by too much. To understand this, think of an EFLP solution, with several edges ($O(n)$) crossing a straight line l. It is not possible to just add two parallel lines to the solution, and connect all edges to them. Unlike the Steiner Tree problem, on EFLP it is not possible to open steiner nodes with no cost.

To overcome this challenge, the authors presented in their work a generalization of the charging argument, also used in [3], and detached it from the Patching Lemma. Using the *Charging Lemma* it was demonstrated that there exists an approximate solution with a very restricted interface that can be guessed in polynomial time. The final solution is also found with dynamic programming.

We will discuss the Charging Lemma in details over the next sections. We will also show how the *Structure Lemma* uses the Charging Lemma to bound the cost increase in the approximate EFLP solution. We will start by covering some definitions that are fundamental for the success of the algorithm.

3.3.2 Initial Perturbation

In this approximation, we will use a few of the same concepts defined in the PTAS for Steiner Tree Problem on Chapter 2. Just like in that chapter, our input points also lay on a randomly shifted *quadtree*, with random values (a, b). We will also consider that each one of the $O(n \log(n))$ squares of the quadtree have $4m$ portals placed on their borders.

For a square S, we will define portals(S) as the set of $4m$ portals of S. A bounding box is defined to be the smallest square containing all input points, and we say that its side length is L.

It is also necessary to ensure that all points lie on integral coordinates and L is polynomial on n. We call this a *well-rounded* solution. A different initial perturbation from Chapter 2 is required to make the instance well-rounded. We cannot use the same initial perturbation arguments since there we assumed that $L \leq \text{OPT}$ and this might not be true for the EFLP (just think of an instance with two sets of clients very far from each other, with an optimal solution which opens two facilities, one assigned for each set).

We will therefore have to develop other methods. Let WRFLP be a well-rounded solution for the EFLP. We will show how to obtain a PTAS for the EFLP from a PTAS to WRFLP.

Lemma 3.3.1. *If there is PTAS for the WRFLP then there is a PTAS for EFLP.*

Proof. Let S^* be an optimal solution to the EFLP, with value $val(S^*) = OPT$. To prove this lemma, we will reduce the EFLP to the WRFLP with a small cost increase, proportional to the value of OPT.

Suppose we have an approximate solution of value A for this instance of the Euclidian Facility Location problem, where $A \leq nOPT$. This approximation can be obtained by using, for example, the approximation algorithm of [28]. We will lay a grid with granularity $A/8kn^2$, with $k > 0$, on top of our instance of the EFLP, and move each point to the center of the nearest square. Note that the minimum non-zero internode distance now is $A/8kn^2$, with a cost increase of at most $2A/8kn^2$ per point moved. This leads to a maximum increase of $2nA/8kn^2$ for all points. Since this is at most $\text{OPT}/4k$ we conclude that moving all points to integral coordinates adds a cost of εOPT to the solution, where $\varepsilon = 1/4k$. Let S' be this modified solution.

Now we have to make sure that the size of the quadtree for S' is polynomial in n. Let L be the side length of the quadtree of S'. If $A \geq L/n^2$, then $L \leq An^2$. Recall that the minimum internode distance, d_{min}, is $A/8kn^2$. Therefore, the depth of the quadtree is $\log(L/d_{min})$, which is smaller or equal to $\log(An^2/(A/8kn^2))$. We have then that the depth of the quadtree is $O(\log(n))$, and we are done.

Suppose then that $A < L/n^2$. Note that in this situation, the size of the grid is much bigger than the size of the solution. We will then treat all squares with sizes greater than An^2 as independent instances of the problem and show that the probability the solution passes between these independent squares is very small.

We will prove bellow on Lemma 3.3.2 that the number of lines crossed by a solution and the dissection is proportional to the size of the solution, so can say that the number of dissection lines crossed by S' is $\Theta(A)$.

Lets consider the number of times the boundaries of a square of size An^2 are crossed. We can say that the probability of a certain line of the shifted dissection (with random (a, b)-shift) being a boundary of square with size An^2 is $1/An^2$ (L/An^2 boundary lines over a total of L lines). Since we have $\Theta(A)$ lines crossed on the dissection, the probability that any line of crossed lines is from a boundary of a square of size An^2 (or higher) is at least A/An^2.

Therefore, with probability at least $1 - 1/n^2$, no square bigger than An^2 is crossed by the solution. This shows that we can treat squares larger than An^2 as independent squares with a small probability of error. We can then assume that $L = O(n^4)$. Since S' has integral coordinates and its side is polynomial in n, we can say that it is an optimal solution to WRFLP. So, we have

$$val(S') = (1 + 1/\varepsilon)val(S^*).$$

Finally, we can obtain a solution to EFLP from S' by returning the instance points to their original position incurring in another cost of $(1/\varepsilon)$OPT. □

3.3.3 Charging and Structure Lemmas

As we already mentioned, the Patching Lemma does not apply to the Facility Location problem so, the authors generalized the charging argument from Lemma 2.5.3 creating the Charging Lemma for the Facility Location. In contrast to the Patching Lemma, the Charging Lemma does not care if a solution S has crossed the lines of the grid a small number of times; it simply bounds these crossings by the cost of an optimal solution. For the Charging Lemma we have to pay attention to "how" and "where" the solution crosses the lines of the grid. We will have to define functions indicating where are the closest facilities on the plane. These functions will be very useful on the dynamic programming.

Lets say we have a solution S for the EFLP. We define $cost(S)$ as the sum of lengths of the solution S. Let l be any line. The authors defined an *R-charging process* as a penalty that can be paid proportional to the number of times S crosses the line l. The charging Lemma then says that if the number of crossings between S and l is proportional to $cost(S)$ (which will be proved on Lemma 3.3.2) and we charge a value that is at most $\varepsilon/\log(n)$ from each line crossed by the solution, then the total expected amount one will pay for this charging process is a fraction of $O(\varepsilon\, cost(S))$, for $\varepsilon > 0$.

Since we do not care about limiting the crossings between solution and lines, instead of looking for an (m, r)-light solution as it was done in Chapter 2, we aim at finding an *m-light* solution. A solution is said to be m-light if every time one of its edges crosses the boundary of a square on the grid it does so through a portal.

We will now state the lemma that bounds the number of crossing between solution and underlying grid by the cost of the solution, a somewhat similar thing from what we have done in Chapter 2. Let $t(l, S)$ be the number of lines in the solution S that crosses a line l on the grid.

Lemma 3.3.2. *If the minimum length of a segment in S is at least 4 units, then* $\sum_l t(S, l) = \Theta(cost(S))$*, where the summation is over all lines in the unit grid.*

The proof is the same as Lemma 2.5.3 from the Steiner Tree PTAS.

Now, we formalize the R-charging process, a fundamental step for the Charging Lemma. Let R be any integer > 0. Let k be the maximum number of times that a maximal dissection line l crosses S. Then, in a R-charging process we are allowed to charge a cost $\frac{k}{R}\cdot$ *length of* l from line l.

Finally, we can enunciate the Charging Lemma:

Lemma 3.3.3 (Charging Lemma). *Suppose an R-charging process is allowed to charge costs only for edges from the top i levels of a grid. Then, if the shifts a,b were chosen randomly, the expected total cost (over the random choices a, b) charged with shifts a, b is $O(i\,\frac{cost(S)}{R})$.*

Proof. Let l be a horizontal line in a grid, and let j be the maximal level of l. We know that l has size $L/2^j$, so a R-charging process is allowed to charge a cost of

$$\frac{t(l,S)}{R}\frac{L}{2^j} \tag{3.19}$$

from l, where $t(l,S)$ is the number of times solution S crosses line l.

Remember the probability that line l is at level j is $2^j/L$. Since we are charging l only considering the i top levels of the grid, the expected cost charged from l is

$$\sum_{j\le i}\frac{2^j}{L}\frac{t(l,S)}{R}\frac{L}{2^j}. \tag{3.20}$$

It follows that the cost charged to l is bounded by

$$\frac{i}{R}t(l,S). \tag{3.21}$$

By linearity of expectations, for all horizontal and vertical lines we have a total expected charged cost of:

$$\frac{i}{R}\sum_{\text{all lines}} t(l,S). \tag{3.22}$$

By Lemma 3.3.2 this value is bounded by $O(\frac{i}{R}\;cost(S))$. □

Although we have considered this proof only for the top i grid levels, we will allow the charging process to occur in all $O(\log(n))$ levels of the quadtree. We can do that, by using an R value that is also $O(\log(n))$. That way the Charging Lemma bounds the cost added by an R-charging process to a cost of a constant times OPT.

Now, on the Structure Theorem for the Euclidian Facility Location problem, we will prove the existence of an $(1+\varepsilon)$OPT approximate solution for the EFLP with the help of the Charging Lemma.

Theorem 3.3.4 (Structure Theorem for Euclidian Facility Location). *Let $m > 1$ be any integer and shifts $0 \le a, b \le L$ be picked randomly. Then for any Euclidian Facility Location problem, with probability at least $1/2$, there is an m-light solution with respect to the (a, b) shifted dissection with cost at most $(1 + O(\frac{\log L}{m}))$OPT.*

Proof. Let S^* be an optimal solution for this instance of the Euclidian Facility Location problem. We will modify this solution to make it m-light with a cost increase of εOPT. To do so, we have to deflect the solution edges connecting client to facilities, so that every time one edge crosses a line of the grid, it does so through a portal.

Recall that we have placed $4m$ portals on the border of each square. We assume that m is a power of 2, without loss of generality. This means that a portal on a level i

is also a portal for all smaller squares with levels $j > i$. Therefore, when we deflect a maximal edge e on level i to make it m-light, it also becomes m-light for all levels $j > i$.

Suppose we have an edge e, such that $e \in S^*$, crossing the border of a maximal square s. Let s_e be the length of this edge. Since there are m portals on each border of s, e can be at most at $s_e/2m$ distance from a portal in s. To make e m-light, suppose that instead of actually deflecting it we will add two lines from the point where e crosses the square to the nearest portal. This will add a cost of at most of two times $s_e/2m$, and therefore, s_e/m to the solution. Thus, we can say for each edge we make m-light we will charge a cost of s_e/m from the maximal line it crosses. This what we just described is an R-charging process with $R = m$. If we repeat this process for all edges of S^*, always charging the costs to the maximal lines of the grid from all $\log(L)$ levels, by the *Charging Lemma*, we are expected to charge a total cost of $O(\log(L)/m)$OPT. Since we have $L = O(n^4)$ by the initial perturbation, if we set m to be $O(\log(n)/\varepsilon)$ we will have a total increase of εOPT to the cost of solution S^*. □

3.3.4 Dynamic Programming

In this section, we present the dynamic programming algorithm which will build an $(1 + 1/(4m))$-approximate m-light solution. Since from the Structure Lemma we know that an m-light solution value is at most $(1+O(\varepsilon))$OPT, when $m = O(\log(n))$, the approximate m-light solution is $((1 + \varepsilon/4\log(n))(1 + O(\varepsilon))\text{OPT})$, which is also $(1 + O(\varepsilon))$OPT.

The dynamic programming will work bottom-up. We will compute solutions for each square of the shifted quadtree, from leaves to the bounding box. For each square S, we will connect clients to either facilities inside of S or to one of the $4m$ portals on the border of S. On the bounding box level we will obviously only connect clients to facilities inside of it, since there are no facilities outside. When connecting clients to portals we will need to have an idea of how close they are from a facility. Enumerating the location of all facilities outside of S would take exponential time, so we will “guess” approximately how close portals are to facilities outside of S and possibly connect clients to them.

We claim that this is an approximate guessing because we might suppose one portal is at at certain distance from a facility and on a smaller level (larger square) discover that this distance is different. To do this guessing, we will assign numbers from 1 to $4m$ to portals, indicating the proximity to a facility. This assignment will be represented by the function *closest*. Two adjacent portals in S should have near closest values, meaning their distances to facilities is similar. Let p be a portal on square S, whose side length is s. The distance between two portals in S is s/m. If portal p is at distance x from a facility, then adjacent portals can be at most at a distance $x + s/m$ from a facility.

Since we always guess the distance with this additive term s/m, if we make a wrong closest assignment, on the worst case scenario, we could guess that a portal is at distance $1\,x$ from a facility when it actually was at distance $4m\,x$. In the example 3.1 below, we see a client being connected to a portal on a first assignment, being that this portal was supposed to be at distance s/m from a facility. In another level, we see that p was actually at distance $4m\,s/m$ from a facility.

As we will see latter, we only guess the closest function when we have at least one

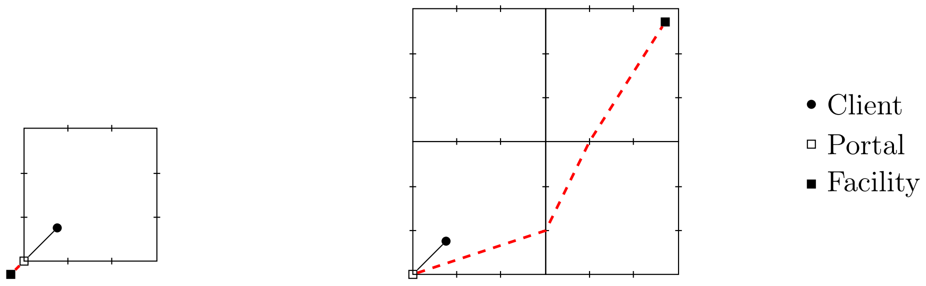

Figure 3.1: a) Approximate closest guessing of distance s/m. b) Actually facility is at most at distance $4m\, s/m$.

facility on the squares, and so we will always have a facility to connect the portal to.

Therefore, the location of a facility outside of S can be wrongly guessed by at most $4m$ of distance. This is why we arrive at a $1/4m$ approximate m-light solution.

To indicate the position of facilities inside a square we will use the function *inside*. Each portal on S will have a correspondent inside value, that can also vary from 1 to $4m$, representing how close this portal is from a facility inside the square. Two adjacent portals should also have similar inside values.

Note that since the closest facility might be inside the square, for every portal p, $closest(p) \leq inside(p)$. For each square S, we will enumerate all possible closest and inside values for all $4m$ portals and consider opening up to n facilities.

We define an instance for the dynamic programming problem as:

a) a non-empty square S in the shifted quadtree;

b) an integer number of facilities f to be opened, such that $0 \leq f \leq n$;

c) if $f \neq 0$, an inside function assignment of number $1, 2, ..., 4m$ to portals, such that if p and p' are successive portals, then $|inside(p) - inside(p')| \leq 1$;

d) if $f < n$, a closest function assignment of number $1, 2, ..., 4m$ to portals, such that if p and p' are successive portals, then $|closest(p) - closest(p')| \leq 1$ and $closest(p) \leq inside(p)$.

If $f = 0$, the goal of the subproblem is to connect clients to portals with assigned closest values. Let s be the size of a square S. If $f > 0$, the goal for subproblem for square S is to open f facilities and connect clients inside of S to either facilities in S or to portals of S, while satisfying the following properties:

1. each portal p from $portals(S)$ is at most at distance $inside(p)\frac{s}{m}$ from a facility;

2. let $n(p)$ be the number of clients connected to a portal p, we have that the length of m-light paths in S plus $\sum_{p\,\in\, portals(S)} closest(p)\, n(p)\, \frac{s}{m}$ is minimum.

The first property makes inside(p) reflect p's distance from a facility. As already discussed, it makes sense that we use this additive term s/m, since each portal is at this distance from each other.

The second property tries to connect clients with the closest facility possible. If the nearest facility inside of S is too distant from a client the m-light would be expensive, and it might be worth it to try to connect it to a facility outside of S. To do that, we have to pay its "closest" price, which will symbolize the distance from a facility outside of S. It makes sense that we multiply the closest of a portal for the number of clients connected to it, because the distance paid to connect it to facilities will we proportional to the number of clients we connect to p.

We will create a look-up table to store the optimal cost for each interface generated by an instance described above. Recall we have $4m$ represents portals per the square. Now, suppose we set an inside value for a certain portal p of S. Each following portal p' adjacent to p has only 3 option of values: $-1+inside(p)$, $0+inside(p)$ or $1+inside(p)$. The same is true for the next portal adjacent to p' and so on. Hence, there are 3^{4m} possibilities for each portal. The same reasoning applies to the closest function. Therefore, for each square S, we have $4m\,3^{4m}$ possible choices for the inside function, and $4m\,3^{4m}$ for the closest function. Being able to open up to n different facilities, for one square S of the quadtree we would have $n\,(4m\,3^{4m})^2$ entries on the table.

Now, we discuss how to compute the table. We will do this by induction. We have two base-case instances: when $f = 0$ or when $f \geq 1$ and the square is a leaf in the quadtree.

When $f = 0$, the goal of the subproblem is to connect each node u in S to a portal p of S, such that the m-light distance to the portal plus the closest value paid is minimum:

$$\min \sum_{u \in C\cap S,\, p \in portals(S)} dist_m(u,p) + closest(p)\frac{s}{m}, \tag{3.23}$$

where u is a client, $C \cap S$ is the set of clients inside square S, $dist_m(u,p)$ is the m-light distance between u and p. It is mandatory that every node u is connected to one portal of S. The minimum value satisfying these requirements will represent this subproblem cost.

When we are dealing with the leaves and $f = 1$, we will open one facility on the center of the square, and connect each inner client to it paying connection cost zero. Then, we set the inside function of each portal p of S according to their distance from the facility opened inside. We will only have to pay for the facility opening cost. Leaves instances with $f > 1$ can be left undefined.

Now lets move to the step of the induction. Let S be a square at level i of the shifted dissection. Suppose the algorithm has solved all subproblems on levels $> i$. Let S_1, S_2, S_3 and S_4 be the children of S. By induction hypothesis S_1, ..., S_4 have been already solved. To find a solution for square S we need to check and combine the best possible solutions for S_1, ..., S_4 for each interface generated by b) and c) for S. The closest and inside values of S will be influenced by the closets() and inside functions from its children.

We will consider a solution to be *admissible* if, for a number of facilities, the combination of instances of S_1, ..., S_4 respect the following rules:

1. the sum of facilities in the configurations of S_1, ..., S_4 must be equal to f,
2. for each portal p of S, there is a portal p' of one of its children such that $\text{dist}(p,p') + inside(p')(s/2m) \leq inside(p)(s/m)$,

and for each portal p of S_1, S_2, S_3 and S_4 there is

3. a. either a portal p' of S_1, S_2, S_3 and S_4 such that $\text{dist}(p,p') + (s/2m)\textit{inside}(p') \leq \textit{closest}(p)s/2m$

   b. or a portal p' of S such that $\text{dist}(p,p') + (s/m)\textit{closest}(p') \leq \textit{closest}(p)(s/2m)$.

Rule 1 ensures that f facilities are opened. Rule 2 is responsible for defining the inside function of S's portals based on its children's inside's. Its says that for each portal p of S, there should always be a portal p' from one of its children that is closer (or at equal distance) than p to a facility, in such a way that it is worth to pay the displacement from p' to p.

Finally, Rule 3 sets the closest functions for S, either based on the closest or inside functions from its children or guessing about close facilities outside.

To solve for S, we will enumerate all choices of configurations for S_1, ..., S_4. If no choice is admissible, then the entry for S is left undefined. We will choose as a solution for S an admissible configuration with minimum cost.

3.3.5 Running Time

As we discussed, the number of entries in the lookup table for a square S in the quadtree is $T = n(4m3^{4m})^2$. To compute the cost of a non-base case square on the quadtree we have to look to all entries of its four children for each entry of S. Thus, for each one of the T entries for S we run through its T^4 children's solution, leading to a running time of $O(T^5)$. Since we know that the total number of non-empty squares in the quadtree is $O(n\log(n))$, we have a running time of $O(n\log(n)T^5)$ for this algorithm. Since $m = O(c\log(n))$, we have that

$$\begin{aligned}
T^5 &= \left(n(4m\,3^{4m})^2\right)^5 \\
&= n^5\,(4m)^{10}\,3^{40m} \\
&\leq n^5\,4(\log(n)\,c)^{10}\,3^{40\log(n)\,c} \\
&\leq n^5\,4(\log(n)\,c)^{10}\,\left(2^{\log(3)}\right)^{40\log(n)\,c} \\
&\leq n^5\,4(\log(n)\,c)^{10}\,2^{(\log(3)\,40\log(n)\,c)} \\
&\leq n^5\,4(\log(n)\,c)^{10}\,n^{\log(3)\,40\,c}.
\end{aligned}$$

Since $n^5\,4(\log(n)\,c)^{10}\,n^{\log(3)\,40\,c} = n^{O(c)}$ and $n\log(n) = n^{O(1)}$, for all squares we have a total running time of $n^{O(c+1)}$ to arrive in a $(1 + 1/c)$-approximate solution.

3.4 Geometric Facility Location

The Geometric Facility Location (GFLP) or Continuous Facility Location problem is an extension of the FLPP where facilities can be located anywhere in $\mathbb{R}^d$, where d is the dimension. Let f be the constant cost to open a facility, where $f > 0$. Given a set of clients $C \in \mathbb{R}^d$, we want to find and open a non empty set of facilities $I \subset \mathbb{R}^d$ and connect each client $c \in C$ to them, paying as least as possible.

This problem was first introduced by Meira et al. [30], who proposed a $(3 + \varepsilon)$-approximation to the Euclidian and squared Euclidian versions of the problem.

In their paper, the authors used the notion of *center* of points. Let S bet a set of points $\in \mathbb{R}^d$. A *center*(S) is a point in $\mathbb{R}^q$ such that the sum of distances between all points $\in S$ to *center*(S) is minimum. They also defined $\phi(C)$ as the set of centers for all subsets of clients in C.

Let l be the distance function between two points in $\mathbb{R}^d$. Let F be a set on facilities given as input on a MFLP problem, $c_{i,j}$ be the cost of connecting a facility i to a client j and f_i the cost to open a facility i in MFLP.

The authors showed that, given an instance I_1 (C, d, l, f) to GFLP and an instance I_2 (F, C, d, c_{ij}, f_i) to MFLP, we can reduce I_1 to I_2, in such a way that every optimal solution of I_2 is also an optimal solution of I_1. On their paper, they demonstrated how it is possible to obtain this reduction by defining F as $\phi(C)$, f as f_i and l as c_{ij}.

At first sight it might seem simple to propose an approximation algorithm to GFLP. Applying this reduction and then using one approximation to MFLP appears to lead to a straightforward approximation to GFLP. However, attention should be made to the reduction step itself, since the number of centers in $\phi(C)$ might be exponential in the size of C.

The authors used a trick to consider only a polynomial number of centers in $\phi(C)$. Consider a *region* as a sphere in $\mathbb{R}^d$, with radius α_j, surrounding every client $c_j \in C$, just like Vazirani's algorithm [21]. The intuition behind this trick is that facilities located on the intersection of client regions have more chances of being opened than facilities outside the intersection of regions. The authors proved then that we need only to consider centers in these intersecting regions, and that this number is polynomial in C.

Using the already known 3-approximation to MFLP by Vazirani et al., and having a cost $(1 + \varepsilon)$-approximation [40] to compute the polynomial centers of C, the authors presented a $(3 + \varepsilon)$-approximation to GFLP.

3.5 Connected Facility Location

The Connected Facility Location problem (CFLP) is a FLP variation with a connectivity restriction. In this problem, facilities are required to be connected among themselves by a Steiner Tree. The choice of a Steiner Tree is straightforward since this is the structure that provides the connection with smallest length possible.

Let $G = (V, E)$ be a graph, $C \subseteq V$ a set of clients , $F \subseteq V$ a set of facilities, and $M \geq 1$ a given parameter, where M is a multiplying factor weighing on the Steiner Tree connecting the facilities. Let f_i be the cost of opening a facility i and c_{ij} the cost of edge (i, j), for $j \in C$ and $i \in F$, where $c_{ij} \geq 0$ and $f_i \geq 0$. Let c_e be the cost of edge e, such that $e \in T$, and T is the Steiner Tree we want to find.

Our objective is to open subset of facilities $I \subseteq F$, connect all clients to facilities, and connect the facilities among themselves involving the minimum cost possible.

This problem can be formulated as follows:

$$\min \sum_{i\in F,\, j\in C} c_{ij} + \sum_{i\in F} f_i + M \sum_{e\in T} c_e. \tag{3.24}$$

Usually, solutions that simply run a FLP approximation and then connect the facilities forming a Steiner Tree do not generate good results (when M is big), but a 4-approximation algorithm was proposed by Eisenbrand et al [14]. This problem is sometimes referenced as the Multicommodity Rent-or-Buy problem [36].

Chapter 4

A PTAS for the Geometric Connected Facility Location Problem

4.1 Introduction

In this section we consider the *Geometric Connected FLP* (GCFLP). This problem is the geometric and prize-collecting variant of CFLP where the set of clients are points in the Euclidean plane and a facility may be opened at any point of the plane. We show the existence of a PTAS for GCFLP.

We make use of a shifted dissection like it was done by Arora et al. in his PTAS for the Steiner Tree problem 2, however, no patching lemma exists for our problem. Indeed, for GCFLP the difficulty arises in two different forms, as we have to account for the costs of *connecting clients* (as in Euclidean FLP) and *installing nodes of the network*.

As we discussed in 3, Arora et al. [4] mitigate the need of a patching lemma by guessing the cost to connect clients, but, using their charging lemma alone is not sufficient, as we have to interconnect facilities. The reason guessing the distance between portals and the closest facility is not sufficient is that in a solution each facility should be directly connected to another facility, but not necessarily to the closest one.

Thus, our dynamic programming framework uses an alternative approach: for each square of the quadtree, we pay only for the network edges between connected coFmponents induced by subproblems. The key observation is that we may estimate the distance between pairs of connected components if we guess the distance from a portal and *each* internal component. While we do not have a patching lemma, and so an arbitrary number of edges may cross a portal, our approach leads to a polynomial-time algorithm since one can assume the number of connected components is constant. We remark that our dynamic programming uses a *portal*-based configuration to deal with a node-weighted problem, what was not known before [34]. This strategy helped us avoiding the use of additional structures, hopefully turning the algorithm simpler and easier to extend. Remy and Steger [34] obtained a PTAS for Node Weighted Steiner Tree (NWST) using a more complex technique, by introducing the *irregular map*, that is a partition of a square into $O(\log n)$ cells of varying size.

The Node Weighted Geometric Steiner Tree problem (NWST) on the plane is a generalization of the Steiner Tree problem on the plane. In this problem, each Steiner node has a cost, what make it hard to use the structure in [3]. Therefore, they introduced a new technique. In this approach the solution is not characterized by the amount of times or where it crosses the boundaries of the grid, but by how is the tree structure inside each square of the grid. Rather than considering edge bordered portals, they introduced (s,t)-map, an irregular partition on a quad-tree square. Grid squares are subdivided in $\mathrm{O}(\log n)$ cells so it is possible to indicate in which cells are the edges connections. Let a *tree component* be a connected internal component made of solution edge and points inside some square of the grid. It is shown that with a small addition on the cost of the optimum solution, there can be a constant amount of tree components inside of each of these squares.

We consider *r-restricted* solutions, that are similar to the standardized solutions of Remy, but rather than describing the solution structure through their map, we consider solutions whose edges cross only the square border at specified portals, in the spirit of Arora's algorithm. Thus, we show an algorithm to deal with weighted nodes that is based on the portals approach. This allows us to deal with prize-collecting more straightforwardly, and hopefully with a simpler presentation.

Our algorithm also leads to a PTAS for the geometric version of the Connected k-medians, that is similar to GCFLP, but only k facilities may be opened, and there is no cost to open a facility. Moreover, we consider the case of different network topologies, when the network is a *ring* or a *star*, rather than a tree, and give PTAS's for these cases as well.

Several other works presented approximations for geometric problems. Mitchell independently obtained a PTAS for ST and TSP [32]. Kolliopoulos and Rao [25] presented an approximation scheme for k-medians in the d-dimensinal Euclidian space for fixed $d > 2$. Das and Mathieu [12] presented a quasi-polynomial time approximation scheme for the Capacitated Vehicle Routing Problem on the plane. Borradaile et al. [10] presented an $\mathrm{O}(n \operatorname{polylog} n)$-time approximation scheme for Steiner Forest Problem. Later, Bateni and Hajiaghayi [8] consider the prize-collecting and budgeted versions of Steiner forest, and gave a PTAS for a natural special case.

This paper is organized as follows. In Sect. 4.2, we present the problem's definition, and show that we can assume that our instance is well-rounded. In Sect. 4.3, we consider near-optimal restricted solutions, that provide nice structures for our algorithm. Finally, in Sect. 4.4, we present a dynamic programming framework which finds a near-optimal restricted solution for our problem.

4.2 Preliminaries

4.2.1 Problem's definition

In the two-dimensional GCFLP, we are given a set of clients $\mathcal{C} \subset \mathbb{R}^2$, a penalty function $\pi : \mathcal{C} \to \mathbb{R}_+$, a facility cost $f \in \mathbb{R}_+$, and a scaling factor $M \in \mathbb{R}_+$, with $M \geq 1$. A solution of the problem is a tree T, whose set of leaves is $C_T \subseteq \mathcal{C}$, and the set of internal nodes is $F_T \subset \mathbb{R}^2_+$. For each leaf $v \in C_T$, we let $\phi_T(v) \in F_T$ be the vertex to which v is connected. We denote by P_T the set of pedant edges, that is, the edges that are incident to C_T. Similarly, the internal edges are denoted by S_T, that is, $S_T = E[T] \setminus P_T$. The objective of the problem is to find such a tree T that minimizes the value

$$\mathrm{val}(T) = f\,|F_T| + \pi(\mathcal{C} \setminus C_T) + \ell(P_T) + M\,\ell(S_T),$$

where $\pi(\mathcal{C} \setminus C_T)$ is the sum of penalties of clients $\mathcal{C} \setminus C_T$, and $\ell(P_T)$ and $\ell(S_T)$ are the total length of edges in P_T and S_T, respectively.

Intuitively, one may think of installing a facility at each point $v \in F_T$, paying opening cost f. Since a facility can be opened anywhere in the plane, a fixed cost is considered. Each client $u \in \mathcal{C}$ is either served by facility $\phi_T(u)$, when the length $\ell(e)$ of edge e that connects u to $\phi_T(u)$ is paid, or disregarded, when a penalty $\pi(u)$ is incurred. The set of open facilities is interconnected by a network with edges S_T, whose cost is proportional to their length, but typically more expensive than serving clients, and so it is scaled by M. We assume that $M \leq n$ since every instance with $M > n$ has an optimal solution with only one open facility, and thus can be reduced to the Geometric 1-Median Problem. Notice that, the Geometric Steiner Tree Problem is a particular case of our problem where $f = 0$, $M = 1$ and $\pi(v) = \infty$ for every $v \in \mathcal{C}$ and, thus, GCFLP is also NP-hard [16].

4.2.2 Well-rounded solutions

We consider the particular case of GCFLP when all input points have odd integral coordinates, and the length L of the bounding box of the set of points $\mathcal{C}$ is polynomially bounded. We denote this particular version by LCFLP.

Lemma 4.2.1. *If there is a PTAS for LCFLP, then there is a PTAS for GCFLP.*

To prove Lemma 4.2.1, we first need to obtain an estimate for the optimal value of the original instance. The following lemma gives a rough approximation for GCFLP. Notice that obtaining a good constant-approximation for the geometric prize-collecting connected FLP is an interesting problem on its own.

Lemma 4.2.2. *There exists a $7n$-approximation algorithm for GCFLP.*

Proof. Let OPT be the optimal value of GCFLP for instance $(\mathcal{C}, p, f, M)$, and let S be the optimal value of a (geometric) prize-collecting minimum Steiner tree with problem instance $(\mathcal{C}, p)$. Notice that an optimal solution of GCFLP induces a solution for the prize-collecting minimum Steiner tree problem of no larger cost, thus $S \leq \mathrm{OPT}$. Also, let S' be the optimal value of a prize-collecting minimum spanning tree with instance $(\mathcal{C}, p)$. It is well known that the length of a minimum spanning tree of a set of points is at most 3/2 times the length of the minimum Steiner tree [13]. This implies that the minimum prize-collecting spanning tree has cost at most 3/2 the cost of the minimum prize-collecting Steiner tree. Therefore $S' \leq (3/2)S \leq (3/2)\mathrm{OPT}$.

We use Goemans and Williamson's 2-approximation [18], and obtain a solution for the prize-collecting minimum spanning tree. Let A be the cost of such solution, and let T' be the obtained tree. We get $A \leq 2S' \leq 3\mathrm{OPT}$.

If T' is empty, then we let $T = \emptyset$, and obtain a solution for GCFLP of cost $\pi(\mathcal{C}) = A \leq 3\mathrm{OPT}$. So, we may assume that T' contains at least a vertex v. Let T be the star consisting of vertex v connected to each client $u \in V[T']$. We choose between T and the empty solution, considering the one with smaller cost. If OPT $< f$, then the optimal solution is, in fact, the empty solution. If OPT $\geq f$, notice that the length of edge $\{v, u\}$ is smaller than the length of T' for each u. Also, we have $\pi(\mathcal{C} \setminus V[T']) \leq A$. We obtain $\text{val}(T) \leq f + A + n\,A + 0 \leq \text{OPT} + 3\text{OPT} + n\,3\text{OPT} \leq 7n\text{OPT}$. □

Now we may reduce GCFLP to LCFLP.

Proof of Lemma 4.2.1. Let A be a solution for the GCFLP of cost at most $7n\,\text{OPT}$ obtained using Lemma 4.2.2, L_0 be the side length of the bounding box containing $\mathcal{C}$, and $\varepsilon' \in (0, 1]$ be a constant to be defined later.

Consider an instance of GCFLP, and a corresponding optimal solution T^*. First, let $\hat{F} \subseteq F_{T^*}$ be the set of open facilities that serve any client, that is, $\hat{F}$ is the image of function ϕ_{T^*}. Since there are only n clients, we have $|\hat{F}| \leq n$. Notice that there exists a minimum cost Steiner tree on $\hat{F}$ with at most $2|\hat{F}|$ nodes, since the maximum number of steiner nodes in a two-dimensional Steiner Tree is $n - 2$ [17]. We assume, without loss of generality, that $|F_{T^*}| \leq 2|\hat{F}| \leq 2n$, since otherwise we could replace S_{T^*} by the edges of such Steiner tree, and obtain a solution of no larger cost.

We scale facility cost, penalties, and every coordinate of instance and solution by

$$\rho = \frac{n^2 M}{\varepsilon' A},$$

and then move every point to the closest point with odd integral coordinates, so that all points are aligned in a grid of granularity 2. The bounding box of the modified instance is $L \leq \rho L_0 + 2$. The increase in the scaled optimal solution is bounded by the total distance by which points have been moved: for each client, we pay the distances by which client and assigned facility were moved, for a total of 4 units; for each edge of S_{T^*}, we pay the distance by which both endpoints were moved scaled by M, for a total of $4M$ units. Let $T^{*\prime}$ be an optimal solution for the modified instance. Since we have n clients, and T^* has at most $2n - 1$ internal edges, we obtain

$$\text{val}(T^{*\prime}) \leq \rho\,\text{val}(T^*) + 4n + (2n - 1)4M \leq \rho\,\text{val}(T^*) + 12nM.$$

Suppose that we are given a solution T' for the modified instance. By moving the points back to their original position, and scaling by $1/\rho$, we obtain a solution T for the original problem. By paying the deviated distances for each client, we obtain

$$\text{val}(T) \leq \frac{1}{\rho}\left(\text{val}(T') + 2n\right).$$

Now we show that we can use the PTAS for LCFLP to obtain a solution T' with cost $\text{val}(T') \leq (1 + \varepsilon')\text{val}(T^{*\prime})$.

If $A \geq L_0/n$, then, since $L \leq \rho L_0 + 2$, we have that $L \leq \rho A n + 2$. Therefore, $L \leq \frac{n^2 M}{\varepsilon' A} A n + 2 = \text{O}(n^4)$, as $M \leq n$. This means that the bounding box of the modified instance is polynomially bounded, and we may apply the PTAS for LCFLP, obtaining a solution T' with $\text{val}(T') \leq (1 + \varepsilon')\text{val}(T^{*\prime})$.

So, we assume that $A < L_0/n$. We consider grids of granularity $g = n^3M/\varepsilon'$ with lines lying on even integral coordinates. There are $g^2/4 = \mathrm{O}(n^8)$ different grids. We claim that there is one grid such that $T^{*\prime}$ is completely contained in one of its cells. Each edge e of $T^{*\prime}$ of length $\ell(e)$ crosses at most $\ell(e)$ lines in the 2-grid (since the lines have even integral coordinates), so the number of lines that are crossed is a lower bound on the length of $T^{*\prime}$, and so on $\mathrm{val}(T^{*\prime})$. Suppose that we pick a g-grid uniformly at random. The probability that a fixed line of the 2-grid is also a line of the g-grid is $\frac{2}{g}$, so the probability that $T^{*\prime}$ crosses a line of the g-grid is at most

$$\mathrm{val}(T^{*\prime})\frac{2}{g} \le (\rho\,\mathrm{val}(T^*) + 12nM)\,\frac{2}{g} \le \left(\frac{n^2M}{\varepsilon' A}A + 12nM\right)\frac{2}{g} = O\left(\frac{1}{n}\right).$$

Therefore, there is a grid such that an optimal solution $T^{*\prime}$ is completely contained in one cell of such a grid. We can thus enumerate all grids, consider each element of the grid as a separate instance, and apply the PTAS for LCFLP for this instance. It is easy to see that the best obtained solution induces a solution T' for the modified instance with $\mathrm{val}(T') \le (1+\varepsilon')\mathrm{val}(T^{*\prime})$.

Finally, we transform the obtained solution T' into a solution T for the original instance. By the discussion above, T has cost

$$\begin{aligned}
\mathrm{val}(T) &\le \frac{1}{\rho}\left(\mathrm{val}(T') + 2n\right) \le \frac{1}{\rho}\left((1+\varepsilon')\mathrm{val}(T^{*\prime}) + 2n\right)\\
&\le \frac{1}{\rho}\left((1+\varepsilon')\rho\,\mathrm{val}(T^*) + (1+\varepsilon')12nM + 2n\right)\\
&\le (1+\varepsilon')\mathrm{val}(T^*) + 26nM\frac{\varepsilon' A}{n^2M} \le (1+\varepsilon')\mathrm{val}(T^*) + 26\varepsilon'\frac{7n\,\mathrm{val}(T^*)}{n}\\
&= (1+183\varepsilon')\mathrm{val}(T^*),
\end{aligned}$$

where last inequality comes from Lemma 4.2.2. We set $\varepsilon' = \varepsilon/183$, where $\varepsilon \in (0,1]$ is the parameter of the PTAS for GCFLP. This completes the proof. □

4.2.3 Shifted dissections

From now on, we only consider instances of LCFLP. We will describe the *shifted dissections.* In the following, we adopt the notation used in [34]. Let L be the bounding box of the input points $\mathcal{C}$, so that every point has coordinates greater than 0 and smaller than L. Without loss of generality, assume that $L = 2^t$, for some integer $t \ge 1$. We choose shifting parameters $a, b \in \{0, 2, \ldots, L-2\}$, so that the vertical line with coordinate a, and the horizontal line with coordinate b lie in the grid of granularity 2. These two lines divide the bounding box in 4 rectangles. We extend each rectangle, so that the length of each side becomes L as in Figure 4.1, and obtain an extended square of side length $2L$, that is denoted by G_0.

A *quadtree* $QT_{a,b}$ is a recursive partitioning of G_0. Each square G is divided into four smaller subsquares, that become the children of G. We repeat this process for each child, until we obtain squares of size 2×2, that become the tree leaves. The level of the squares of side length L is defined as 0, and the level of a child of a square with level l is $l+1$.

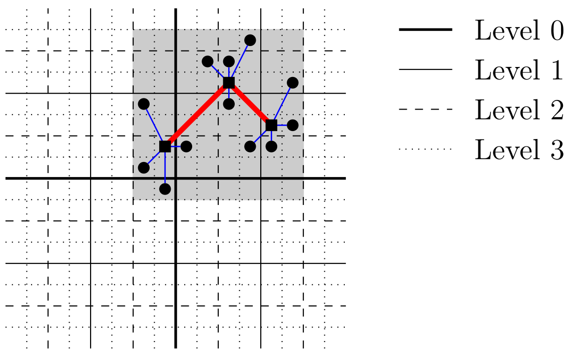

Figure 4.1: The quadtree $QT_{4,2}$, the dissection lines and the bounding box. Edges in P_T and S_T are represented as thin blue edges and as thick red edges, respectively.

Also, the level with respect to a of a vertical grid line r with coordinate $x \in (a-L, a+L)$ is the unique l for which there exists an *odd* integer i such that $x = a + i \cdot L/2^l$. The level with respect to b of a horizontal grid line r is defined analogously. See Figure 4.1.

A square with level l has side length $L/2^l$. Let e be any curve or segment contained in G_0. We denote by $\ell(e)$ the length of e. Also, suppose that the level of the smallest square $G \in QT_{a,b}$ that completely contains e is l, then the level of e is $\text{lev}(e) = l + 1$. Therefore, the largest square of $QT_{a,b}$ that is crossed by e has level $\text{lev}(e)$.

4.3 Restricted Solutions

In this section, we show how one can consider more structured solutions for LCFLP in the design of a PTAS for this problem. Those more structured solutions allows the development of a dynamic programming algorithm which is the base of our approximation algorithm.

4.3.1 m-light solutions

Let m be an even positive integer. For a given square G of the quadtree, we consider $4m$ points along the edges of G that are called *portals*. Each corner receives a portal, and the remaining $4(m-1)$ portals are uniformly distributed along the edges. The set of portals associated with G is denoted by $Q[G]$.

Notice that, if the square has level l, then two consecutive portals along the edges of G are at distance $s(l) := (L/2^l)/m$, where $L/2^l$ is the side length of G, since there are $m+1$ portals in each edge.

We consider a more relaxed definition of trees. Rather than assuming that edges are straight lines connecting two vertexes, we allow edges that are bent at portals. This is stated more precisely in the following definition.

Definition 4.3.1. *An m-*light path *is any line in the plane that only crosses the boundary of a square G at portals of G, for every G in $QT_{a,b}$. An m-*light solution *T (with respect to a, b) for LCFLP is a solution where every edge of T is an m-light path.*

Notice that the length of an m-light path connecting two points u, v may be larger than the Euclidean distance between u and v, $d(u, v)$. We denote by $\tilde{d}(u, v)$ the length of the smallest m-light path connecting u to v.

Starting with an optimal solution T^*, we will derive a series of solutions with the same set of facilities, but different edges. To bound the increase cost due to modifications in terms of the cost of T^*, we consider a transitive relation on the set of solutions, in a way similar to [34]. In the following definition, we say that $T \lhd T'$ if each edge of T can be mapped to an edge of T' of not greater level.

Definition 4.3.2. *Let T and T' be solutions of LCFLP. We say that $T \lhd T'$ if $C_T = C_{T'}$, $F_T = F_{T'}$, and there is a bijection $\gamma : E[T] \to E[T']$, such that $\mathrm{lev}(e) \geq \mathrm{lev}(\gamma(e))$ for every $e \in E[T]$.*

Now we modify an optimal solution T^*, and obtain an m-light solution T. Consider an edge e of T^*, and suppose that it crosses a dissection line r that has level l. We bend e to the closest portal in r of a square of level l. The increase in the cost for this operation is at most $s(l)$, that is charged to line r. We have just described a charging process, as defined by Arora et al. [4]. By using a version of their *charging lemma*, we may show that the expected increase in the cost due to bending edges is only a fraction $\mathrm{O}(t/m)$ of the total length of the edges in T^* (recall that t is the height of the quadtree, and $L = 2^t$). This is formalized in the next lemma.

Lemma 4.3.3. *There exists an m-light solution T, such that $T \lhd T^*$. Moreover, the expected cost of T, when a, b are chosen uniformly at random is*

$$\mathbb{E}[\mathrm{val}(T)] \leq \mathrm{val}(T^*) + \frac{2(t+1)}{m}\ell(P_{T^*}) + \frac{2(t+1)}{m} M\, \ell(S_{T^*}).$$

To prove Lemma 4.3.3, we need an auxiliary lemma, that allows us to use the charging argument. In the following, given a set of edges S of T^* and a line r of the grid, we denote by $\mathrm{cr}(S, r)$ the number of times that r is crossed by a segment of S.

Lemma 4.3.4. *For any set of edges S of T^*, we have $\sum_r \mathrm{cr}(S, r) \leq \ell(S)$, where the summation is over all lines of the grid of granularity 2.*

Proof. Notice that, since vertexes of T^* have odd integral coordinates, each edge e of S may cross at most $\ell(e)/2$ vertical lines and $\ell(e)/2$ horizontal lines of the grid of granularity 2. By summing the number of crosses for all segments in S, the lemma follows. □

Proof of Lemma 4.3.3. Let r be a line with level l. Notice that r only contains edges of squares at levels $l, \ldots, t$. Let p be a portal of level l. Since m is even, p is also a portal of each level $l+1, \ldots, t$. Thus, any path that crosses r at p also crosses r at portals of levels $l+1, \ldots, t$. Therefore, for each such r, we only need to respect portals of level l.

First, we concentrate on edges of P_{T^*}. For each line r with level l, we bend each edge of P_{T^*} that crosses r to the closest portal of level l. For each such crossing, we have an additional cost of at most $s(l) = (L/2^l)/m$. Since there are at most 2^l horizontal (vertical) lines with level l, the probability that the level of line r is l is at most $2^l/(L/2) = 2^{l+1}/L$. The expected additional cost due to bending P_{T^*} is thus

$$\sum_r \sum_{l=0}^{t} \frac{2^{l+1}}{L} \frac{L}{2^l m} \operatorname{cr}(P_{T^*}, r) = \frac{2(t+1)}{m} \sum_r \operatorname{cr}(P_{T^*}, r) \leq \frac{2(t+1)}{m} \ell(P_{T^*}).$$

By bending edges of S_{T^*}, and using analogous arguments, we also bound the expected increase in length of the edges of S_{T^*}. The lemma follows by the linearity of expectation, and the fact that bending an edge does not change its level. □

4.3.2 (m, r)-restricted solutions

For a given m-light solution T, and a square G of $QT_{a,b}$, let $k(G, T)$ be the number of connected components induced by $S_T \cap G$, that is, the number of disjoint subgraphs of the internal network of T that are completely contained in G. For an arbitrary T, this number can be very large. The following definition considers only m-light solutions such that, for each square of the quadtree, the number of internal components is bounded by a constant.

Definition 4.3.5. *An (m, r)-restricted solution is an m-light solution T such that $k(G, T) \leq r$, for every $G \in QT_{a,b}$.*

In the following, we want to transform an arbitrary m-light solution into an (m, r)-restricted solution, for some constant r. The process uses operations defined in [34]. In our case, however, we must ensure that the edges of a solution cross square edges only at portals, so we perform some extra fixing steps, that might increase the solution cost. To bound this increase, we first give the following auxiliary lemma.

Lemma 4.3.6. *Let G be a square, v be any point contained in G, and p be a portal of G. Then there is an m-light path e connecting v to p. Moreover, $\ell(e) \leq 2\, d(v, p)$.*

Proof. Without loss of generality, we assume that $v = (x_v, y_v)$ and $p = (x_p, y_p)$ such that $x_p \geq x_v$, and $y_p \geq y_v$, that is, p is at the right and upper sides of v.

We want to obtain an m-light path e connecting v and p that is a sequence of straight segments $e_1, \ldots, e_m$, such that the angle between the horizontal axis and e_i, for $i = 1, \ldots, m$, is between 0 and $\pi/2$. The claim follows by backward induction on the level l of G. If $l = t$, than G is a leaf of $QT_{a,b}$, and we simply connect v and p using a straight line. Otherwise, suppose that we may obtain such a sequence of segments for a square of level $l + 1$. In this case, we bend the line connecting v to p to the closest portal which is at the right and upper sides of v in each edge of a children of G that is crossed. Now, we use the induction hypothesis for each child (see Figure 4.2), and combine the obtained solutions.

Since the angle between each segment and the horizontal axis is between 0 and $\pi/2$, we know that total length of all segments is at most 2 times the distance between v and p. □

Lemma 4.3.7 is a consequence of Lemma 4.3.6.

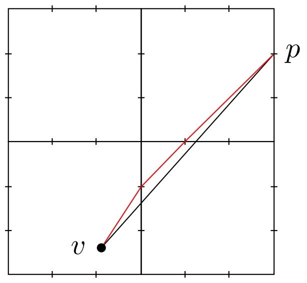

Figure 4.2: Bending an edge connecting v to p recursively.

Lemma 4.3.7. *Let G be a square with level l, and u, v be two points contained in G. Then there is an m-light path e connecting u to v. Moreover, $\ell(e) \leq 2\,(d(u,v) + s(l))$.*

Proof. Without loss of generality, we assume that G is the minimal square that contains u and v. We bend the line connecting u to v to the closest portal in each edge of a children of G that is crossed. Notice that at most 2 edges are crossed, and for each time that we bend the edge, we increase its size in at most $s(l)/2$, that corresponds to the distance between two consecutive portals of a child of G.

We obtain segments, such that each one is completely contained in one child G' of G, and has one endpoint at a portal of G'. Now we apply Lemma 4.3.6 for each child G' separately, and combine the solutions to obtain an m-light path e. The cost of e is at most 2 times the length of the segments. Thus, $\ell(e) \leq 2\,(d(u,v) + 2\,s(l)/2))$. □

Now we obtain an (m, r)-restricted solution, for some large r.

Lemma 4.3.8. *Assume r is such that $\sqrt{r} \leq m$. There exists an (m, r)-restricted solution T', such that $T' \lhd T^*$. The expected cost of T', when a, b are chosen uniformly at random is*

$$\mathbb{E}[\mathrm{val}(T')] \leq \mathrm{val}(T^*) + \frac{2(t+1)}{m}\ell(P_{T^*}) + \left(\frac{2(t+1)}{m} + \frac{32}{\sqrt{r}}\right) M\,\ell(S_{T^*}).$$

Proof. Let T be the solution obtained by Lemma 4.3.3. As in [34], we decrease the number of internal components in squares of each level one at a time, and proceed in bottom-up fashion. For each level l, we will obtain a solution T_l, such that every square of level l or greater contains at most r internal components. Since we may assume that each leaf contains at most one node (since otherwise we could connect all points paying zero), there is nothing to do in level t, and we set $T_t = T$.

Suppose we have already obtained T_{l+1}, for some l, and want to obtain T_l. We consider each square G with level l and side length A, such that $k := k(G, T_{l+1}) > r$, and perform a *reduction* operation: (i) select k vertexes of $F_{T_{l+1}}$ in distinct internal components, and obtain a Hamiltonian path, say $(u_1, \ldots, u_k)$, of cost at most $2A\sqrt{k}$ (this can be done for sufficiently large r [23]); (ii) remove $k-1$ edges, where each edge connects a distinct internal component to a point that is external to G in a way such that the resulting graph is still a tree.

Notice that, since u_1 and u_2 are in distinct internal components of tree T_{l+1}, there is a unique path between u_1 and u_2, and thus there must be an edge e crossing G in such a path. Adding edge (u_1, u_2) of the Hamiltonian path creates a cycle that contains e, that can be safely removed, and decreases the number of internal components by one. Arguing similarly for each edge (u_i, u_{i+1}), with $1 \le i < k$, shows that operation (ii) can be correctly performed.

The operation described above maintains the connectivity of the tree, and, further, does not increase the number of internal components of squares of level greater than l, since the removed edges cross the border of G.

Notice however that the edges in the Hamiltonian path may not respect the portals of the quadtree. To obtain an m-light tree, we replace each of the $k-1$ edges of the Hamiltonian path by an m-light path obtained using Lemma 4.3.7. The increase in the solution cost due to the addition of the Hamiltonian path, and the bending operation is divided among the $k-1$ removed external edges, so that each one is charged

$$\begin{aligned}\frac{1}{k-1}2\left(2A\sqrt{k}+(k-1)s(l)\right) &= \left(\frac{2\sqrt{k}}{k-1}+\frac{1}{m}\right)\frac{2L}{2^l} \le \left(\frac{3}{\sqrt{k}}+\frac{1}{m}\right)\frac{2L}{2^l}\\ &\le \left(\frac{3}{\sqrt{r}}+\frac{1}{\sqrt{r}}\right)\frac{2L}{2^l} = \frac{8L}{2^l\sqrt{r}},\end{aligned}$$

where we used $k \ge 3$ and $\sqrt{r} \le m$. Thus, each edge that was removed due to a reduction operation at level l is charged $\delta_l := \frac{8L}{2^l\sqrt{r}}$. We obtain T_l, and proceed until we obtain T_0. The final modified solution is thus $T' := T_0$.

Notice that we have not changed edges in P_T, and that we replaced each edge of E_T by an edge of larger level, so $T' \lhd T$, and thus $T' \lhd T^*$. We also want to bound the expected increase in the cost. By following the arguments of Lemma 13 by Remy and Steger [34] (the difference is that their charge on an edge of level l is $\frac{2L}{2^l\sqrt{r}}$), we obtain $\mathbb{E}[\ell(S_{T'}) - \ell(S_T)] \le \frac{32}{\sqrt{r}}\ell(S_{T^*})$.

□

4.4 Dynamic Programming

We consider the subproblem of finding an (m, r)-restricted solution in a square G of the quadtree. To describe an instance, we guess the distance of the closest facility to each portal. Enumerating all possible distances, however, would take exponential time, thus we guess each distance only approximately. For each portal of a square with level l, we guess the distance up to an additive term $s(l)$, the distance between two portals on a square. Moreover, the estimated distances from two consecutive portals to their closest facilities may differ by at most $s(l)$, so we only have to guess if the estimated distance of a portal to its closest facility is equal, larger or smaller (by $s(l)$) than the estimated distance for the previous portal. Thus only $\mathrm{O}(m)$ bits are used to describe a configuration, that may be enumerated in polynomial time when $m = \mathrm{O}(\log n)$.

4.4.1 Subproblem definition

An instance for the subproblem is a pair (G, K) formed by a square G of the quadtree, and a configuration K. A configuration K for a subproblem is described by

- an integer number z of facilities, such that $0 \le z \le 2n$;
- an integer number k of internal components, such that $\min\{1, z\} \le k \le \min\{r, z\}$;
- if $z > 0$, a function $\mathit{inside} : [k] \times Q[G] \to [4m]$, such that for each number $1 \le i \le k$ and pair of consecutive portals p, p', it holds $|\mathit{inside}(i, p) - \mathit{inside}(i, p')| \le 1$; and
- a function $\mathit{closest} : Q[G] \to [4m]$, such that for each pair of consecutive portals p, p', it holds $|\mathit{closest}(p) - \mathit{closest}(p')| \le 1$.

For a given instance, we want to select a set of facilities that are divided into k components. For each component i, the function $\mathit{inside}(i, \cdot)$ represents a requirement that there must be a facility of this component inside the square at a certain (m-light) distance to a given portal. The function *closest* represents a promise that the closest facility of the global solution is at a certain (m-light) distance to a given portal. Note that the closest facility may be either inside, or outside the square G.

Each component represents a subnetwork of the global solution. Each edge of the network may be broken into several squares, therefore the cost associated with a subproblem for G corresponds only to the network edges that are completely contained in G. The service cost associated with the clients in G, however, are completely incurred to a solution of the subproblem. There are three possibilities: the client is not served and the penalty is paid; the client is connected to an internal facility; or the client is connected to an external facility. In the latter case, the distance to the facility is estimated by using *closest*.

Let G be a square of level l, and K be a configuration. A solution for (G, K) is comprised of:

- a set of points $F \subseteq G$ with odd integer coordinates, such that $|F| = z$;
- a partition of F into sets $F_1, \ldots, F_k$, such that, for each portal p and component i, there is $v \in F_i$, and an m-light path between v and p with length at most $\mathit{inside}(i, p)\, s(l)$.
- for each $1 \le i \le k$, a spanning tree T_i on F_i that is formed by m-light edges;
- a subset $C \subseteq \mathcal{C} \cap G$; and
- an assignment $\phi : C \to F \cup Q[G]$.

Recall that $\tilde{d}(u, v)$ is the length of the smallest m-light path connecting points u and v. The service cost of a client $u \in \mathcal{C} \cap G$ is

$$\mathit{serv}(v) := \begin{cases} \pi(v) & \text{if } v \notin C, \\ \tilde{d}(v, \phi(v)) & \text{if } v \in C \text{ and } \phi(v) \in F, \\ \tilde{d}(v, \phi(v)) + \mathit{closest}(\phi(v))\, s(l) & \text{if } v \in C \text{ and } \phi(v) \in Q[G]. \end{cases}$$

The value of a given solution for the subproblem is:

$$|F|\,f + M\sum_{i=1}^{k}\ell(T_i) + \sum_{v\in\mathcal{C}\cap G} serv(v).$$

We say that a *basic* instance for a subproblem on a square G is an instance such that $z = 0$, or G is a leaf of the quadtree. Conversely, we say that a *non-basic* instance is one instance that is not basic.

4.4.2 Filling the dynamic programming table

We create a table B indexed on all valid instances of the subproblem. For an instance (G, K), we set $B(G, K)$ as an upper bound on the cost of the best solution found for (G, K). First, we solve all basic instances, then we solve the remaining instances of the table inductively, in a bottom-up manner. Suppose that we have already filled all instances of squares at level $l + 1$ or greater, then we solve each non-basic instance corresponding to a square of level l, by possibly querying the table at instances of level $l + 1$.

Basic instances: Consider a basic instance (G, K), and suppose that G has level l. If $z = 0$, then there are two possibilities for each client in G: we connect the client through a portal, or leave it unconnected and pay the penalty cost. Therefore, in this case we set

$$B(G,K) = \sum_{v\in\mathcal{C}\cap G}\ \min_{p\in Q[G]}\{\,\tilde{d}(v,p) + closest(p)\,s(l),\ \pi(v)\,\}.$$

If $z \geq 1$, then G must be a leaf. The level of G is thus $l = t$, the square has side length 2. Therefore, there exists only one point with odd integer coordinates, that is located at the center of the square, say c. So, we place z facilities at c and connect each client, if any, to a facility paying zero as connection cost. In this case, we have to check whether the requirement of the function *inside* is violated for some component, and, in such a case, mark $B(G, K)$ as infeasible. Therefore, we set

$$B(G,K) = \begin{cases}\infty & \text{if } d(c,p) > inside(i,p)\,s(l) \ \text{ for any } p\in Q[G] \text{ and } 1\leq i\leq k,\\ zf & \text{otherwise.}\end{cases}$$

Non-basic instances: Now we consider a non-basic instance (G, K). Let $l < t$ be the level of G. We will try to obtain a solution for this instance by combining solutions of subproblems for the children G_1, G_2, G_3, G_4 of G. First, we consider a choice of corresponding configurations K_1, K_2, K_3, K_4 for the children. We solve each instance (G_j, K_j), for $1 \leq j \leq 4$, and obtain a subnetwork for G_j with k_j connected components. Each connected component can be viewed as a single vertex, so that the network for square G is a forest on the set of components. We proceed as follows: for each $1 \leq j \leq 4$, we create k_j new vertexes $(j, 1), \dots, (j, k_j)$ representing components of G_j. We let $\mathcal{V}$ be the set of all vertexes created for this particular choice of configurations, and let $\mathcal{D}$ be the set of all graphs on $\mathcal{V}$.

We say a choice of configurations K_1, K_2, K_3, K_4 together with a graph $D \in \mathcal{D}$ is *admissible* for K if the following conditions hold:

(C1) D is a forest containing exactly k trees $T_1, \ldots, T_k$, where k is the number of components in K;

(C2) $B(G_1, K_1), B(G_2, K_2), B(G_3, K_3), B(G_4, K_4) < \infty$;

(C3) the sum of the number of facilities of every child configuration equals to the number of facilities in K;

(C4) for each pair p, i of portal and component of G, there exists a pair p', i' of portal and component of a child G_j such that $(j, i') \in V[T_i]$[1], and $\tilde{d}(p, p') + \mathit{inside}(i', p')\, s(l+1) \le \mathit{inside}(i, p)\, s(l)$;

(C5) for each portal p of a child, one of the following holds:

1. there is a pair p', i' of portal and component of a child such that $\tilde{d}(p, p') + \mathit{inside}(i', p')\, s(l+1) \le \mathit{closest}(p)\, s(l+1)$;
2. there is a portal p' of G such that $\tilde{d}(p, p') + \mathit{closest}(p')\, s(l) \le \mathit{closest}(p)\, s(l+1)$.

Suppose that we are given a choice of configurations, and that the algorithm has already found a solution for each child instance. If the choice of configurations is admissible, then we may obtain a feasible solution for G. Each solution of a subproblem already provides a subnetwork. The only thing missing is connecting the subnetworks. This can be done by replacing each edge of D by the minimum length m-light path connecting two nodes of the corresponding components. While we do not know a priori the locations of the installed facilities, we know that an m-light path must cross squares through portals, and we have an estimate of the distance between the portal and the closest vertex of each component. Therefore, for any two components $u, u' \in \mathcal{V}$, where $u = (j, i)$, and $u' = (j', i')$ we define the weight of edge $\{u, u'\}$ as follows (an example is illustrated in Figure 4.3):

$$\begin{aligned} w(u, u') := \min\{&\tilde{d}(p, p') + \mathit{inside}(i, p)s(l+1) \\ &+ \mathit{inside}(i', p')s(l+1) \;:\; p \in Q[G_j],\, p' \in Q[G_{j'}]\}. \end{aligned}$$

The total cost for this solution is thus

$$B(G_1, K_1) + B(G_2, K_2) + B(G_3, K_3) + B(G_4, K_4) + \sum_{e \in E[D]} w(e).$$

We enumerate all choices of configurations K_1, K_2, K_3, K_4, and graph $D \in \mathcal{D}$. If no such choice is admissible, then we set $B(G, K) = \infty$, otherwise we set $B(G, K)$ with the minimum value obtained.

[1]That is, the vertex $(j, i') \in \mathcal{V}$ that represents the connected component i' of G_j is part of connected component i of G.

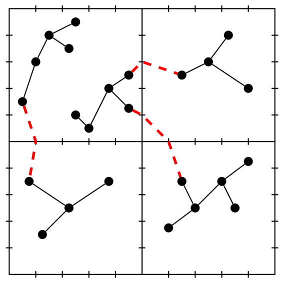

Figure 4.3: Possible configuration: costs of thick dashed red edges are estimated by *inside*$(i, \cdot)$ functions.

4.4.3 Dynamic programming algorithm obtains an approximate solution

While the dynamic programming algorithm does not calculate an optimal (m, r)-restricted solution, we argue that it obtains a "close to optimal" solution. Let T be a given (m, r)-restricted solution with respect to a fixed quadtree. We show that there is an admissible configuration which encodes T for each square of the quadtree where each edge e of T is estimated by the algorithm with an error that is not larger than $\mathrm{O}(s(\mathrm{lev}(e)))$.

By intersecting T with a square G of the quadtree, we obtain disjoint trees $T_1, \dots, T_k$ (with $k \leq r$) on vertexes of $F_T \cap G$ that correspond to internal connected components on G. Also, we obtain an assignment ϕ from each client $C_T \cap G$ to either a facility of $F_T \cap G$, or a portal of G. Therefore, T induces a solution for a subproblem defined on G.

The *interface* of T with respect to G, denoted by $K(G, T)$, is the configuration for the subproblem such that

- $z = |F_T \cap G|$;
- $k = k(G, T)$;
- if $z > 0$, then for each portal $p \in Q[G]$, and each $1 \leq i \leq k$, let v be the facility of connected component T_i with minimum m-light distance to portal p, so we choose *inside*(i, p) such that

$$s(l) \leq \mathit{inside}(i, p)s(l) - \tilde{d}(p, v) < 2s(l); \tag{4.1}$$

- for each portal $p \in Q[G]$, we choose *closest*(p) such that

$$\mathit{closest}(p) = \min_{v \in F_T} \mathit{estim}(p, v), \tag{4.2}$$

  where *estim*(p, v) is defined as follows. Let $e(p, v)$ be the m-light path with minimum length that connects p to a facility v. Also let l be the level of G, and $h = \mathrm{lev}(e(p, v))$. Then, *estim*$(p, v)$ is the unique integer number such that

$$2s(h) + \sum_{g=h}^{l-1} s(g) \leq \mathit{estim}(p, v)s(l) - \tilde{d}(p, v) < 2s(h) + \sum_{g=h}^{l} s(g). \tag{4.3}$$

Notice that $\mathit{estim}(p,v)s(l) - \tilde{d}(p,v) \leq 4s(h)$, and thus, the estimate guesses the distance up to an error of four times $s(h)$.

We remark that the interface $K(G,T)$ is well defined. Clearly, for each portal p and component i, $\mathit{closest}(p)$ and $\mathit{inside}(i,p)$ are unique. Also, either there is a facility inside G or there must be a facility in the parent of G, so one can show that $\mathit{inside}(i,p) \leq 4m$ and $\mathit{closest}(p) \leq 4m$. Now, consider consecutive portals p and p', and let v and v' be the corresponding closest facilities in T_i. Since $\tilde{d}(p',v') \leq s(l) + \tilde{d}(p,v)$, we obtain $|\mathit{inside}(i,p) - \mathit{inside}(i,p')| \leq 1$.

A similar argument is due to $\mathit{closest}(p)$. Notice, however, that the facility v that minimizes $\mathit{estim}(p,v)$ is not necessarily the facility v' that is closest to p (as $\mathit{estim}(p,v')$ could be potentially very large, depending on $\mathrm{lev}(e(p,v'))$. To see that $|\mathit{closest}(p) - \mathit{closest}(p')| \leq 1$ for consecutive portals p and p', observe that, if $\mathit{closest}(p) = \mathit{estim}(p,v)$ for some v, then $\mathit{closest}(p') \leq \mathit{estim}(p',v) \leq 1 + \mathit{estim}(p,v) = 1 + \mathit{closest}(p)$.

Lemma 4.4.1. *Let T be an (m,r)-restricted solution, G be a square of the quadtree, and $K = K(G,T)$. Then $B(G,K)$ is at most*

$$|F_T \cap G| f + \pi((\mathcal{C} \setminus C_T) \cap G) + M \sum_{e \in S_T \cap G} [\ell(e) + 4s(\mathrm{lev}(e))] + \sum_{e \in P_{T \cap G}} [\ell(e) + 4s(\mathrm{lev}(e))],$$

where $P_{T\cap G} \subseteq P_T$ is the set of edges connecting clients in G.

Proof. We show the lemma by backward induction on the level l of the square G. Note that the lemma holds for basic instances, since either no edge of S_T is contained in G, or every edge of S_T contained in G has zero length (the latter case occurs when G is a leaf of the quadtree with two or more coincident facilities). Also, for each client $u \in C_T \cap G$, let $v^* \in F_T$ be the facility of T to which u is connected, and let $e := e(p,v^*)$; then, for each portal p of G, we have $\tilde{d}(u,p) + \mathit{closest}(p)s(l) \leq \tilde{d}(u,p) + \mathit{estim}(p,v^*)s(l) \leq \tilde{d}(u,p) + \tilde{d}(p,v^*) + 4s(\mathrm{lev}(e))$, i.e., the cost of e is guessed correctly with error of at most $4s(\mathrm{lev}(e))$.

Now suppose that the lemma holds for every square of level $l+1$, and that G is a non-basic square with level l. Let G_1, G_2, G_3, G_4 be the children of G, and let K_1, K_2, K_3, K_4 be the corresponding interfaces. The solution T also induces a forest $D \in \mathcal{D}$ on the components of children of G. We claim that the choice of K_1, K_2, K_3, K_4 and D is admissible, and thus is considered by the dynamic programming algorithm. Conditions (C1) to (C3) hold trivially. We prove that conditions (C4) and (C5) hold in the following.

For Condition (C4), let $p \in Q[G]$ be a portal, T_i be a connected component of G, and v be the facility of T_i that is closest to p. Let G' be the child of G that contains v, and assume that the minimum m-light path from p to v crosses portal p' of G'. Also, let i' be the component of G' in which v is contained. By Equation (4.1), we know that $\mathit{inside}(i',p')s(l+1) < \tilde{d}(p',v) + 2s(l+1) = \tilde{d}(p',v) + s(l)$, and that $\tilde{d}(p,v) + s(l) \leq$

$\mathit{inside}(i,p)s(l)$. Therefore

$$\begin{aligned}\tilde{d}(p,p') + \mathit{inside}(i',p')s(l+1) &< \tilde{d}(p,p') + \tilde{d}(p',v) + s(l)\\ &= \tilde{d}(p,v) + s(l)\\ &\le \mathit{inside}(i,p)s(l),\end{aligned}$$

and thus condition (C4) is safe.

For Condition (C5), consider a portal p of a child G_j of G, and let v be the facility of F_T such that $\mathit{estim}(p,v)$ is minimum. Notice that, since $e(p,v)$ crosses a portal of G_j, $\mathrm{lev}(e(p,v)) \le l+1$. We have two cases, depending on the value of $h := \mathrm{lev}(e(p,v))$.

First, suppose that $h = l+1$, then $e(p,v)$ does not cross G, and so $v \in G$. Thus $e(p,v)$ crosses a portal p' of the child G' containing v. Let i' be the component of G' that contains v. By Equation (4.1) we know that $\mathit{inside}(i',p')s(l+1) < \tilde{d}(p',v)+2s(l+1) = \tilde{d}(p',v)+s(l)$, and by Equations (4.2) and (4.3) we know that $\mathit{closest}(p)s(l+1) = \mathit{estim}(p,v)s(l+1) \ge \tilde{d}(p,v) + 2s(h) + \sum_{g=h}^{(l+1)-1} s(g) = \tilde{d}(p,v) + 2s(l+1) + \sum_{g=l+1}^{l} s(g) = \tilde{d}(p,v) + s(l)$. Thus,

$$\begin{aligned}\tilde{d}(p,p') + \mathit{inside}(i',p')s(l+1) &< \tilde{d}(p,p') + \tilde{d}(p',v) + s(l)\\ &= \tilde{d}(p,v) + s(l)\\ &\le \mathit{closest}(p)s(l+1),\end{aligned}$$

and we are done in this case.

Now, assume that $h \le l$, then $v \notin G$, and thus $e(p,v)$ crosses a portal p' of G. Again, by Equations (4.2) and (4.3), we know that $\mathit{closest}(p')s(l) \le \mathit{estim}(p',v)s(l) < \tilde{d}(p',v) + 2s(h) + \sum_{g=h}^{l} s(g)$, and that $\mathit{closest}(p)s(l+1) = \mathit{estim}(p,v)s(l+1) \ge \tilde{d}(p,v) + 2s(h) + \sum_{g=h}^{(l+1)-1} s(g)$. Thus,

$$\begin{aligned}\tilde{d}(p,p') + \mathit{closest}(p')s(l) &< \tilde{d}(p,p') + \tilde{d}(p',v) + 2s(h) + \sum_{g=h}^{l} s(g)\\ &= \tilde{d}(p,v) + 2s(h) + \sum_{g=h}^{l} s(g)\\ &\le \mathit{closest}(p)s(l+1),\end{aligned}$$

and we are done also in this case. Therefore, condition (C5) is satisfied. We conclude that the choice of configurations and graph D is admissible.

The value of $B(G,K)$ is at most the sum $B(G_1,K_1) + B(G_2,K_2) + B(G_3,K_3) + B(G_4,K_4)$ plus the cost of D (see Figure 4.3). By the induction hypothesis, the sum of the values for the subproblems already accounts for all opening cost, service cost, and each edge of S_T that is completely contained in a child of G. Therefore, we only need to show that the cost of edges of D (that have level $l+1$) are guessed correctly up to an error of $4s(l+1)$. Let e be an edge of D with endpoints v and v'. Let p, p' be the portals crossed by the minimum m-light path connecting v to v'. Also, let i, i' be the internal components of v and v'. The estimated cost of e is $\tilde{d}(p,p') + \mathit{inside}(i,p)\, s(l+1) + \mathit{inside}(i',p')\, s(l+1) \le$

$\tilde{d}(v, v') + 2s(l+1) + 2s(l+1)$. This completes the proof. □

To obtain an (m, r)-restricted solution, we solve the subproblem instances on G_0 with $z = 1, \dots, 2n$, $k = 1$, *inside* $\equiv 4m$ and *closest* $\equiv 4m$. Let K_0 be a configuration for which we obtain the minimum cost. Lemma 4.4.2 is implied directly from the previous lemma, when $G = G_0$.

Lemma 4.4.2. *Let T be an (m, r)-restricted solution, then*

$$B(G_0, K_0) \le \mathrm{val}(T) + 4 \sum_{e \in P_T} s(\mathrm{lev}(e)) + 4M \sum_{e \in S_T} s(\mathrm{lev}(e)).$$

4.4.4 Approximating an optimal (m, r)-restricted solution

In this section, we show that, our algorithm does not compute an optimal (m, r)-restricted solution, in expectation it finds a solution not much more expensive than the optimal one. In order to show this, we combine Lemmas 4.3.8 and 4.4.2, obtaining the following lemma.

Lemma 4.4.3. *Suppose $\sqrt{r} \le m$. If a, b are chosen uniformly at random, then*

$$\mathbb{E}[B(G_0, K_0)] \le \mathrm{val}(T^*) + \frac{10(t+1)}{m}\ell(P_{T^*}) + \left(\frac{10(t+1)}{m} + \frac{32}{\sqrt{r}}\right) M\,\ell(S_{T^*}).$$

Proof. Let T' be as in Lemma 4.3.8. Since $T' \lhd T^*$, we have $\mathbb{E}[\sum_{e \in P_{T'}} s(\mathrm{lev}(e))] \le \mathbb{E}[\sum_{e \in P_{T^*}} s(\mathrm{lev}(e))] \le \frac{2(t+1)}{m}\ell(P_{T^*})$, where the last inequality follows by using the same arguments as in the proof of Lemma 4.3.3. Analogously, we obtain $\mathbb{E}[\sum_{e \in S_{T'}} s(\mathrm{lev}(e))] \le \frac{2(t+1)}{m}\ell(S_{T^*})$. Now, combining with Lemma 4.4.2, we obtain

$$\begin{aligned}
\mathbb{E}[B(G_0, K_0)] &\le \mathbb{E}\left[\mathrm{val}(T') + \sum_{e \in P_{T'}} 4s(\mathrm{lev}(e)) + M \sum_{e \in S_{T'}} 4s(\mathrm{lev}(e))\right] \\
&\le \mathrm{val}(T^*) + \frac{2(t+1)}{m}\ell(P_{T^*}) + \left(\frac{2(t+1)}{m} + \frac{32}{\sqrt{r}}\right) M\,\ell(S_{T^*}) \\
&\quad + \frac{4 \cdot 2(t+1)}{m}\ell(P_{T^*}) + \frac{4 \cdot 2(t+1)}{m} M\ell(P_{T^*}) \\
&\le \mathrm{val}(T^*) + \frac{10(t+1)}{m}\ell(P_{T^*}) + \left(\frac{10(t+1)}{m} + \frac{32}{\sqrt{r}}\right) M\,\ell(S_{T^*}).
\end{aligned}$$

□

Now we put all the pieces together, and obtain a PTAS for LCFLP.

Lemma 4.4.4. *There exists a PTAS for LCFLP.*

Proof. First, notice that one can easily modify the dynamic programming algorithm so that, for each square G and configuration K, we also obtain a solution for subproblem instance (G, K) of cost at most $B(G, K)$. Let $\varepsilon > 0$ be any constant. We set $r = (1/\varepsilon)^2$, and $m = 10(t+1)/\varepsilon$. If $\sqrt{r} \ge m$, then $n \in \mathrm{O}(1)$, and we solve the problem exactly by brute force in constant time. Otherwise, we use the dynamic programming algorithm to obtain a solution of expected cost at most $B(G_0, K_0)$. Lemma 4.4.3 implies that the expected cost of this solution is

$$\begin{aligned}\mathbb{E}[B(G_0,K_0)] &\le \mathrm{val}(T^*) + \frac{10(t+1)}{\frac{10(t+1)}{\varepsilon}}\ell(P_{T^*}) + \left(\frac{10(t+1)}{\frac{10(t+1)}{\varepsilon}} + \frac{32}{\frac{1}{\varepsilon}}\right) M\,\ell(S_{T^*}) \\ &\le \mathrm{val}(T^*) + \mathrm{O}(\varepsilon)(\ell(P_{T^*}) + M\ell(S_{T^*})) \le (1+\mathrm{O}(\varepsilon))\mathrm{val}(T^*).\end{aligned}$$

Now, we bound the running time of the algorithm. The quadtree has height $t = \log L = \mathrm{O}(\log n)$, and therefore it contains at most $1 + 4 + \ldots + 4^{\mathrm{O}(\log n)} = n^{\mathrm{O}(1)}$ squares, and the number of distinct configurations is at most $2n{\cdot}r{\cdot}2^{\mathrm{O}(mr)} = n^{\mathrm{O}(1/\varepsilon^3)}$, so, for constant ε, the number of entries of B is $n^{\mathrm{O}(1)}$. Also, to calculate each entry of the table, we enumerate tuples with four configurations, and for each such choice we consider all graphs of $\mathcal{D}$. The number of such graphs with up to $4r$ vertexes is at most $2^{(4r)^2} = \mathrm{O}(1)$. Therefore, the algorithm runs in polynomial time. By running the dynamic programming algorithm for each of the $\mathrm{O}(L^2) = n^{\mathrm{O}(1)}$ choices of a, b, we obtain, in polynomial time, a solution whose cost is at most the expected value. □

By combining with Lemma 4.2.1, we obtain our main result:

Theorem 4.4.5. *There exists a PTAS for GCFLP.*

4.5 Extensions

It is possible to adapt Lemma 4.2.1 for the Geometric Connected k-medians problem. Thus, by using our algorithm with $f = 0$, and choosing K_0 as the configuration with k facilities and one internal connected component, we can also obtain a PTAS for this problem.

Theorem 4.5.1. *There exists a PTAS for the Geometric Connected k-medians.*

The topology of the switches networks considered in GCFLP is a tree. We now briefly sketch how to obtain PTAS's for the case of different topologies. The topologies are depicted in Figure 4.4. First, we claim that it is enough to consider well-rounded instances, as defined in Section 4.2. For trees, this is obtained by Lemma 4.2.1. The same statements may be obtained for the variant with ring and star topologies: it is enough to recall that this lemma depends only on a rough (polynomial-factor) approximation, that is obtained by Lemma 4.2.2. Since a star is also a tree, and the length of a circuit is not smaller than the minimum spanning tree, a lower bound for the GCFLP (with tree topology) is also a lower bound on the problem with ring or star topologies. Moreover, the solution considered in the proof of Lemma 4.2.2 contains only one facility, and thus also obtains an approximation for the other considered topologies. Lemma 4.2.1 is adapted accordingly.

Now we only need to argue that there is a algorithm that finds an approximation solution for well rounded instances. For the case of the star topology, the problem reduces to Euclidean FLP. Recall that in Euclidean FLP, one is given a set of candidate facilities, each with a given cost. For the reduction, one may thus try all possibilities for the star center (since the center has integer coordinates, and the bounding box is polynomial, there are only polynomially many guesses); also, for each point with integer coordinates, we create a candidate facility with opening cost f plus the distance to the guessed center. For the case of the ring topology, one may observe that Arora's patching lemma applies, so one can consider only solutions whose network edges cross lines a constant number of times. Thus, it is possible to define an adequate subproblem with polynomially many configurations, and we obtain a PTAS also for this problem.

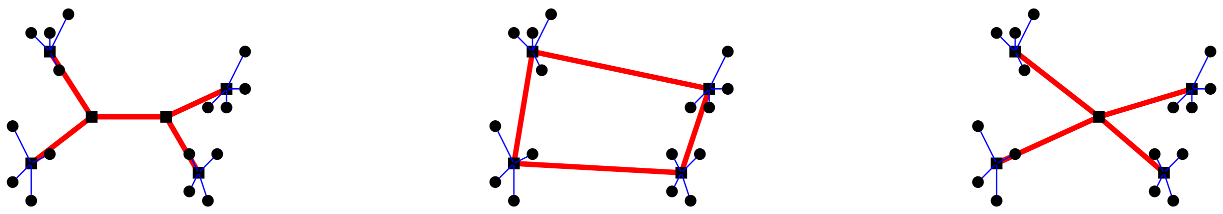
Figure 4.4: Types of internal network: *tree*, *ring* and *star* topologies.

Theorem 4.5.2. *There exist PTAS's for GCFLP with* ring *and* star *topologies.*

Chapter 5

Conclusion

The Facility Location family of problems present several practical applications. This motivated us to study the Geometric Connected Facility Location problem. In the GCFLP, facilities can be anywhere in the plane, and we have a weighted tree interconnecting facilities. In this work, we proposed a PTAS for the two-dimensional version of the problem.

To be able to present our result, we started by introducing Arora's PTAS for the Steiner Tree problem. We then discussed the Facility Location family of problems presenting a Primal-Dual approximation for the Metric Facility Location problem and Arora's PTAS for the Euclidian version. Our work is an extension of Arora's charging technique presented in these approximations, leading to the first PTAS to the GCFLP.

We considered our solution on a recursively partitioned grid. Besides having the solution crossing the grid only at pre-stablished points, we showed how the number of interconnected facilities components in each square of the grid can be made constant with a small increase to the cost of the solution. The approximate solution could then be found with dynamic programming. Our work has been published in the journal Theory of Computing Systems.

Bibliography

[1] Aaron Archer. Inapproximability of the asymmetric facility location and k-median problems, 2000.

[2] Sanjeev Arora. Nearly linear time approximation schemes for euclidean tsp and other geometric problems. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 554–563, 1997.

[3] Sanjeev Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *Journal of ACM*, 45(5):753–782, 1998.

[4] Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. Approximation schemes for euclidean k-medians and related problems. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 106–113, 1998.

[5] A. Steger B. Korte, H. J. Promel. *Paths, Flows and VLSI-Layout.* 1990.

[6] M. L. Balinski and MATHEMATICA PRINCETON N J. *On Finding Integer Solutions to Linear Programs.* 1964.

[7] M. Gisela Bardossy and S. Raghavan. Dual-based local search for the connected facility location and related problems. *INFORMS Journal on Computing*, 22(4):584–602, 2010.

[8] MohammadHossein Bateni and Mohammad Taghi Hajiaghayi. Euclidean Prize-Collecting Steiner Forest. *Algorithmica*, 62(3-4):906–929, 2012.

[9] Marshall Bern and Paul Plassmann. The steiner problem with edge lengths 1 and 2. *Information Processing Letters*, 32(4):171–176, 1989.

[10] Glencora Borradaile, Philip N. Klein, and Claire Mathieu. A Polynomial-Time Approximation Scheme for Euclidean Steiner Forest. *Foundations of Computer Science*, pages 115–124, 2008.

[11] Vincent Cohen-Addad, Philip N. Klein, and Claire Mathieu. The power of local search for clustering. *CoRR*, abs/1603.09535, 2016.

[12] Aparna Das and Claire Mathieu. A Quasi-polynomial Time Approximation Scheme for Euclidean Capacitated Vehicle Routing. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 390–403, 2010.

[13] D.-Z. Du and F.K. Hwang. A proof of the Gilbert-Pollak conjecture on the Steiner ratio. *Algorithmica*, 7(1-6):121–135, 1992.

[14] Friedrich Eisenbrand, Fabrizio Grandoni, Thomas Rothvoß, and Guido Schäfer. Approximating connected facility location problems via random facility sampling and core detouring. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1174–1183, 2008.

[15] M. R. Garey, R. L. Graham, and J. D. Ullman. Worst-case analysis of memory allocation algorithms. In *Proceedings of the Fourth Annual ACM Symposium on Theory of Computing*, pages 143–150, 1972.

[16] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness.* 1990.

[17] E. N. Gilbert and H. O. Pollak. Steiner minimal trees. *SIAM Journal on Applied Mathematics*, 16(1):1–29, 1968.

[18] Michel X. Goemans and David P. Williamson. A General Approximation Technique for Constrained Forest Problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.

[19] Anupam Gupta, Jon Kleinberg, Amit Kumar, Rajeev Rastogi, and Bulent Yener. Provisioning a Virtual Private Network: A Network Design Problem for Multicommodity Flow. In *Proc. of the 33th Annual ACM Symposium on Theory of Computing*, pages 389–398, 2001.

[20] Dorit S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing*, 11(3):555–556, 1982.

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

[22] David S. Johnson. Approximation algorithms for combinatorial problems. In *Proceedings of the Fifth Annual ACM Symposium on Theory of Computing*, pages 38–49, 1973.

[23] Howard J. Karloff. How Long Can a Euclidean Traveling Salesman Tour Be? *SIAM Journal on Discrete Mathematics*, 2(1):91–99, 1989.

[24] Richard M. Karp. *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations*, pages 85–103. 1972.

[25] Stavros G. Kolliopoulos and Satish Rao. A Nearly Linear-Time Approximation Scheme for the Euclidean k-Median Problem. *SIAM Journal on Computing*, 37(3):757–782, 2007.

[26] L. Kou, G. Markowsky, and L. Berman. A fast algorithm for steiner trees. *Acta Informatica*, 15(2):141–145, 1981.

[27] Alfred A. Kuehn and Michael J. Hamburger. A heuristic program for locating warehouses. In *Mathematical Models in Marketing: A Collection of Abstracts*, pages 406–407. 1976.

[28] Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. In *Proceedings of the 38th International Conference on Automata, Languages and Programming - Volume Part II*, pages 77–88, 2011.

[29] Jyh-Han Lin and Jeffrey Scott Vitter. ϵ-approximations with minimum packing constraint violation (extended abstract). In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 771–782, 1992.

[30] Luis A. A. Meira and Flávio K. Miyazawa. A continuous facility location problem and its application to a clustering problem. In *Proceedings of the 2008 ACM Symposium on Applied Computing*, pages 1826–1831, 2008.

[31] R Dahab P Feofiloff CG Fernandes CE Ferreira KS Guimaraes FK Miyazawa JC Pina Jr J Soares Y Wakabayashi MH Carvalho, MR Cerioli. *Uma introdução sucinta a algoritmos de aproximação.* Publicações Matemáticas do IMPA, 2001.

[32] Joseph S. B. Mitchell. Guillotine Subdivisions Approximate Polygonal Subdivisions: A Simple Polynomial-Time Approximation Scheme for Geometric TSP, k-MST, and Related Problems. *SIAM Journal on Computing*, 28(4):1298–1309, 1999.

[33] J. Plesnik. The complexity of designing a network with minimum diameter. *Networks*, 11(1):77–85, 1981.

[34] Jan Remy and Angelika Steger. Approximation Schemes for Node-Weighted Geometric Steiner Tree Problems. *Algorithmica*, 55(1):240–267, 2007.

[35] David B. Shmoys, Éva Tardos, and Karen Aardal. Approximation algorithms for facility location problems (extended abstract). In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, pages 265–274, 1997.

[36] Chaitanya Swamy and Amit Kumar. Primal–dual algorithms for connected facility location problems. *Algorithmica*, 40(4):245–269, 2004.

[37] Vijay V. Vazirani. *Approximation Algorithms.* 2001.

[38] J. Vygen. Approximation algorithms for facility location problems (lecture notes). `http://www.or.uni-bonn.de/~vygen/files/fl.pdf`, 2005.

[39] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms.* 1st edition, 2011.

[40] Guoliang Xue and Yinyu Ye. An efficient algorithm for minimizing a sum of p-norms. *SIAM Journal on Optimization*, 10(2):551–579, 1999.