Geometrical and Combinatorial Optimization Problems

Habilitationsschrift
für das Fach Mathematik

eingereicht von
Dr. rer. nat. Jens Maßberg

Juli 2015
Contents

1 Introduction 1
   1.1 Steiner Trees .......................... 2
   1.2 Repeater Tree Problem .................... 2
   1.3 Further Geometrical Problems ............... 4

2 Facility Location Ratio 5
   2.1 Problem Formulation ....................... 6
   2.2 Notation and General Observations ........... 7
   2.3 The 2-dimensional $\ell_k$ Plane ............ 10
   2.4 The Euclidean Plane ....................... 13
   2.5 The Rectilinear Plane ...................... 14
   2.6 Conclusion and Future Work ................. 18

3 Steiner Tree Embedding 19
   3.1 Problem Formulation ....................... 20
   3.2 Linear Programs .......................... 21
   3.3 Moving Components ......................... 24
   3.4 Half-Integrality of Solutions .............. 27
   3.5 Dynamic Programming ....................... 30
   3.6 An Optimal Polynomial Time Algorithm ....... 32

4 Steiner Arborescences with Depth Restrictions 37
   4.1 Problem Formulation ....................... 37
   4.2 Feasibility ............................. 38
   4.3 Main Idea ............................... 39
   4.4 Reduction Overview ....................... 40
   4.5 The Tile Design .......................... 41
   4.6 NP-completeness ......................... 51

5 Binary Trees with Choosable Edge Lengths 55
   5.1 Problem Formulation ....................... 56
5.2 Core Algorithm .................................................. 57
5.3 Refined Algorithm .............................................. 59
5.4 Conclusion and Future Work ................................. 64

6 The Art Gallery Problem ............................... 65
   6.1 Preliminaries .................................................. 67
   6.2 The Rectilinear Art Gallery Theorem ................. 69
   6.3 A Bound in Terms of the Perimeter ................. 70
   6.4 Guarding $m$-polyominoes .............................. 73

7 Bibliography ....................................................... 77
Chapter 1

Introduction

In the seventeenth century Fermat asks how to connect three points in the plane by a shortest network. He observed that a network consisting of three edges connecting these points with an additional forth point $s$ may be shorter than just using edges between the original points. Torricelli was the first who proved that this network has minimal length if all angles at $s$ between any two edges are 120 degree. Later this problem has been extended to the Steiner tree problem: Given a set of points in the plane, compute a shortest tree connecting these points and additional Steiner points. This can be further generalized to Steiner trees in metric spaces or Steiner trees in graphs.

As in this prominent example there are many problems in combinatorial optimization that were originally stated in a geometrical context and are very descriptive. Later they have been generalized to more abstract settings but still today there are many important applications that can be modeled appropriately by geometrical variants of classical optimization problems.

In this thesis we consider five of such variants that are all motivated by practical applications. We give NP-completeness results, study the structure of the problems and their solutions and derive efficient algorithms for them. In the remainder of this chapter we give a short introduction to the different problems and their motivation. Some of the problems studied in this thesis are variants of the rectilinear Steiner tree problem and originally arose from applications in VLSI design. Therefore we start by giving some basic definitions and results on Steiner trees and have a closer look at so-called repeater trees.

We assume that the reader is familiar with basic concepts of optimization. During this thesis we use the notation of Korte and Vygen [44].
1.1 Steiner Trees

The Steiner tree problem is easily stated: Given a metric space \((V, d)\) and a finite set of terminals \(T \subset V\) the task is to find a tree \(G\) with \(T \subseteq V(G)\) such that the total length of the tree

\[
\sum_{\{v, w\} \in E(G)} d(v, w)
\]  

is minimized. The vertices in \(V(G) \setminus T\), that is, the vertices of the tree that are not terminals, are called Steiner points.

Due to many applications in practice the Steiner tree problem in the rectilinear plane and in the Euclidean plane are of special interest. We refer the reader to the book of Brazil and Zachariasen [6] that gives a very good overview of the theory and algorithms for Steiner tree problems in the plane.

Computing a shortest Steiner tree is NP-hard [42], even if the metric space is the rectilinear plane [18]. For the latter case Hanan [27] has shown that there is always an optimal tree where all Steiner points are placed at the vertices of the Hanan grid on the terminals where the Hanan grid on a set of terminals is the grid obtained by constructing vertical and horizontal lines through each terminal. The vertices of the grid are the intersection points of these vertical and horizontal lines.

A variant of the Steiner tree problem is the Steiner arborescence problem where one of the terminals is distinguished - called the root - and the task is to compute a shortest Steiner tree where all root-terminal-paths are shortest ones. Shi and Su [58] showed that it is NP-hard to compute a shortest rectilinear Steiner arborescence.

Several variants of the Steiner tree problem are motivated by the repeater tree problem that arises in VLSI design.

1.2 Repeater Tree Problem

Repeater trees are tree-like substructures on a chip whose task is to carry an electrical signal from a source circuit to several sink circuits. It consists of horizontal and vertical wires and additional repeaters. The repeaters are used to handle the delay of the signal on its way from the source to the sinks. They just evaluate the identity function and therefore serve no purpose related to the logic of the chip. Repeater trees have to satisfy given timing constraints, that is, the signal has to arrive at each sink not later than a given required arrival time. These timing constraints play a crucial role: If they are not met
the chip will not work with the desired frequency. In this case the chip can only work slower than specified or the design has to be further optimized.

Repeater trees can be modeled by rectilinear Steiner trees with additional constraints: An instance consists of a set of terminals $T$ with positions $p : T \to \mathbb{R}^2$ in the plane, required arrival times $\text{rat} : T \to \mathbb{R}$ and additional physical parameters. One of the terminals is the root and the other terminals are the sinks.

Bartoschek et al. [3] have shown that by adding repeater circuits at appropriate positions of the tree the delay of a signal on a path from the source to a sink is approximately proportional to the length of the path plus a constant delay for each Steiner point on the path. Thus the delay on the path from the root $r$ to a terminal $t$ in a repeater tree $S$ can be modeled as

$$\sum_{(v,w) \in E_S[r,t]} ||p(v) - p(w)||_1 + c \cdot |E_S[r,t]| \quad (1.2)$$

where $E_S[r,t]$ denotes the set of edges on the unique path from $r$ to $t$ in $S$ and $c$ is some constant.

The task in the repeater tree problem is now to build a Steiner tree satisfying the timing constraints given by the required arrival times while minimizing the power consumption of the tree $S$ which is proportional to the length

$$l(S) := \sum_{\{v,w\} \in E(S)} ||p(v) - p(w)||_1 \quad (1.3)$$

of the tree.

Repeater trees are a fruitful source for many different combinatorial optimization problems. A general approach to build such trees is given by Bartoschek et al. [3].

Now consider the problem that the topology of the Steiner tree is already given and the task is to place the Steiner points optimally. As in this case the number of Steiner points on root-terminal paths can not be changed, the timing constraints become length restrictions on those paths. In Chapter 3 we consider this rectilinear Steiner tree embedding problem with length restrictions and show that it can be solved optimally by a combinatorial polynomial time algorithm.

If the topology of the tree is not given and has to be computed, the problem becomes NP-hard as it contains the rectilinear Steiner tree problem. But does the problem become easier if we consider a more restricted version of it? In Chapter 4 we investigate the case, that the timing constraints are as tight as possible, meaning that by further tightening the constraints, the
problem becomes infeasible and does not have a solution. We prove, that even this version of the problem is NP-hard.

In practice there is sometimes a certain degree of freedom how to distribute the additional delay caused by a vertex $v$ to the branches rooted at $v$. We address to this problem in Chapter 5.

1.3 Further Geometrical Problems

There are many other nice optimization problems that have geometrical aspects. A famous one considered in Chapter 2 is the facility location problem in the plane. Given a set of clients in the plane, a set of possible facility positions and facility opening costs the task is to open a subset of the facilities and assign each client to an open facility such that the total cost of connecting the clients to their facilities plus the opening costs is minimized. We study the gap between the cost of an optimal solution if the facilities can be opened anywhere in the plane and optimal solutions where the facilities can only be opened at positions of clients.

Another well-known problem that combines combinatorial and geometrical aspects is studied in Chapter 6: the art gallery problem. In this problem we face the task to guard an art gallery whose layout is represented by a polygon. To this end we can place guards inside the gallery that are represented by points. A point $p$ of the gallery is guarded, if there exists a guard at a position $q$ such that the line segment between $p$ and $q$ does not leave the gallery. In this thesis we consider the rectilinear (or orthogonal) art gallery problem where all walls of the gallery are axis-parallel. We establish a new framework that allows us to enable new simple proofs for upper bounds on the number of guards in terms of the number of vertices or the area of the gallery. Moreover, we prove a new upper bound in terms of the diameter of the gallery.

In many geometrical optimization problems as the Steiner tree problem and the facility location problem the instances contain a set of points embedded in a metric space. Depending on the context, we will refer to these points as terminals, sinks or clients if we consider Steiner trees, chip design or facility location problems, respectively.
Chapter 2

Facility Location Ratio

The first problem we are looking at is the *metric facility location problem*. Given a set of clients in a metric space and possible facility positions, the task is to open a set of facilities and connect each client to an open facility. The goal is to minimize the connecting cost plus the cost to open the facilities, a formal definition follows in the first section. We focus on the case that the metric is the $\ell_k$-space for $k \in \mathbb{N} \cup \{\infty\}$ and in particular the rectilinear $\ell_1$- and the Euclidean $\ell_2$-plane.

In the last decade there has been much progress in finding good approximation algorithms for this problem. One class of algorithms is based on rounding an optimal fractional solution of a linear programming relaxation of the facility location problem [8, 10, 59, 61]. Moreover, there are combinatorial primal-dual approximation algorithms [37, 38] as well as algorithms based on local search [2]. The best approximation factor so far was achieved by Li [47] who presented a 1.488-approximation with polynomial runtime.

The running time of all those algorithms depends on the number of possible facility positions. In some metrics like the Euclidean plane it is not obvious which finite set of points to consider as possible facility positions. Moreover, a small number of possible facility positions heavily decreases the running time. A simple strategy to address this problem is to use only the client locations as candidates for open facilities, raising the question how much worse a solution in this restricted setting can be than an optimal solution to the problem where facilities can be opened anywhere. In this chapter we give upper and lower bounds on this ratio for several metrics and introduce the definition of the *facility location ratio*.

The results in this chapter are joint work with Dr. Jan Schneider.
2.1 Problem Formulation

The uncapacitated facility location problem is defined in the following way: Given a set $F$ of potential facilities, a finite set $C$ of clients, facility opening costs $f_j \in \mathbb{R}_+$ for all $j \in F$, and a connection cost function $d : C \times F \to \mathbb{R}_{\geq 0}$, find a set $F \subset F$ (the open facilities) and an assignment $\sigma : C \to F$ such that

$$\sum_{j \in F} f_j + \sum_{s \in C} d(s, \sigma(s))$$

is minimized. We assume that the connection costs form a metric, meaning that they are symmetric and satisfy the triangle inequality.

In the capacitated version of the problem there is a capacity limit $u_j \in \mathbb{N}$ for every facility $j \in F$. In addition to the facilities that are opened and the assignment of the clients to the facilities one has to find numbers $k_j \in \mathbb{N}$ for every $j \in F$ satisfying

$$|\{s \in C | \sigma(s) = j\}| \leq k_j u_j$$

for all $j \in F$ such that

$$\sum_{j \in F} k_j f_j + \sum_{s \in C} d(s, \sigma(s))$$

is minimized.

The facility location problem is MAX SNP-hard. In fact, it cannot be approximated within a factor of $\alpha < 1.463$ unless $P = NP$ [25, 61].

How much worse than an optimal solution can a solution be where the facilities can only be opened at client positions?

If the facility opening costs are allowed to be non-uniform, the gap can be arbitrarily large: Assume opening a facility at a client position costs $f_1 > 0$ and opening at any other position costs $f_2 = 0$. Then for large $f_1$, a solution with open facilities only at client positions can be arbitrarily worse compared to an optimum solution. Similarly, non-uniform capacity limits lead to an arbitrarily large gap. For these reasons, we henceforth focus on the variant of the problem that has uniform opening costs $f_j = f \in \mathbb{R}_{>0}$ and uniform capacities $u_j = u \in \mathbb{R}_{>0}$ for all $j \in F$.

Note that our question is closely related to the Steiner ratio, which is defined as the supremum over all instances of the ratio between the length of a minimum spanning tree and the length of a minimum Steiner tree. In this context, spanning trees are considered as special Steiner trees which have Steiner points only at terminal locations. The Steiner ratio of a metric space
has been studied extensively. In general, it is at most 2 [21]. Hwang [32] showed that the Steiner ratio of the rectilinear plane is 1.5, and Gilbert and Pollak [21] conjectured that the Steiner ratio of the Euclidean plane is \( \sqrt{3}/2 \), although no proof has yet been found [35].

A similar property of metric spaces, which we call the Steiner star ratio, is also related to the facility location ratio. It is defined as the supremum over all instances of the ratio between the lengths of a shortest star and a shortest Steiner star where a Steiner star consists of the connections of all terminals to one Steiner point, while in a star all points are connected to one of the given points. This topic was extensively studied by Fekete and Meijer [15, 16], who showed that the Steiner star ratio of the rectilinear plane is 1.5. Ismailescu and Park [34] recently improved an earlier technique by Dumitrescu et al. [13] to prove that the Steiner star ratio of the Euclidean plane is at most 1.3546.

The definition of these ratios motivates us to define the facility location ratio for a metric space as the supremum of the ratio between the cost of an optimum solution of the facility location problem where facilities can only be opened at client positions and the cost of an optimum solution where facilities can be opened anywhere. We will see that the facility location ratio is the same for both the uncapacitated and the capacitated version of the problem.

Note that in contrast to the Steiner ratio and the Steiner star ratio in the facility location ratio we have additional costs to open one or more facilities. We use upper bounds for the Steiner star ratio in order to establish upper bounds for the facility location ratio.

### 2.2 Notation and General Observations

As a preparation of our main proofs, we provide some formal definitions and basic theorems in this section. Given an instance \( \mathcal{I} := (\mathcal{F}, \mathcal{C}, f, d) \) of the uncapacitated facility location problem and a nonempty set \( F \subset \mathcal{F} \) of facilities, let

\[
    c(\mathcal{I}, F) := |F| f + \sum_{s \in \mathcal{C}} \min_{t \in F} d(s, t)
\]

(2.4)

 denote the cost of the solution of \( \mathcal{I} \) where \( F \) are the open facilities. The cost of a solution is uniquely determined by \( F \) because the assignment \( \sigma : \mathcal{C} \to F \) is trivially obtained by setting \( \sigma(s) := \arg \min_{t \in F} d(s, t) \). We further define \( c_{\text{fac}}(\mathcal{I}, F) := |F| f \), the facility opening cost of \( F \), and \( c_{\text{cl}}(\mathcal{I}, F) := c(\mathcal{I}, F) - c_{\text{fac}}(\mathcal{I}, F) \), the connection cost of \( F \). In the following, we omit \( \mathcal{I} \) in this notation if it is clear from the context.
We denote by $\kappa(\mathcal{F}, \mathcal{C}, f, d)$ the cost of an optimum solution of $\mathcal{I}$. For simplicity of notation we abbreviate $\kappa(\mathcal{F}, \mathcal{C}, f, d)$ by $\kappa(\mathcal{F}, \mathcal{C})$ if $f$ and $d$ are circumstantial. Using this notation, we define the facility location ratio as follows.

**Definition 2.1.** For a metric space $(V, d)$, the facility location ratio is defined as

$$\rho(V, d) := \sup \left\{ \frac{\kappa(\mathcal{C}, \mathcal{C}, f, d)}{\kappa(\mathcal{F}, \mathcal{C}, f, d)} : \emptyset \neq \mathcal{C} \subseteq \mathcal{F} \subseteq V, |\mathcal{C}| < \infty, f > 0 \right\}. \tag{2.5}$$

In other words, $\rho(V, d)$ equals the worst possible ratio of the cost of a best restricted solution of a facility location instance (i.e. a solution where facilities may only be opened on client locations) and its optimal solution, taken over all instances in the metric space $(V, d)$.

The first observation regarding this definition is that it is not necessary to consider instances for which all optimum solutions require 2 or more open facilities. For such an instance, one of the stars in an optimum solution would form a sub-instance for which the ratio in Definition 2.1 is not smaller than the ratio of the full instance.

**Lemma 2.2.** The value of $\rho(V, d)$ does not change if the supremum is taken only over instances having an optimum solution with exactly one open facility.

**Proof.** For an instance $\mathcal{I} = (\mathcal{F}, \mathcal{C}, f, d)$ let $F^* = \{j_1, \ldots, j_k\}$ be an optimum solution, that is, $c(\mathcal{I}, F^*) = \kappa(\mathcal{F}, \mathcal{C}, f, d)$. Let $C_i := \{s \in \mathcal{C} | \sigma(s) = j_i\}$ be the clients assigned to $j_i$. As $F^*$ is an optimum solution, $\{j_i\}$ is an optimum solution of $(\mathcal{F}, C_i, f, d)$ for $1 \leq i \leq k$. Thus $\kappa(\mathcal{F}, \mathcal{C}) = \sum_{i=1}^{k} \kappa(\mathcal{F}, C_i)$. Additionally, since splitting an instance into sub instances cannot improve the cost of an optimum solution, we have $\kappa(\mathcal{C}, \mathcal{C}) \leq \sum_{i=1}^{k} \kappa(\mathcal{C}_i, \mathcal{C}_i)$. Choosing $i^*$ as the index that maximizes $\kappa(C_i, \mathcal{C}_i)/\kappa(\mathcal{F}, \mathcal{C}_i)$, we conclude

$$\frac{\kappa(\mathcal{C}, \mathcal{C})}{\kappa(\mathcal{F}, \mathcal{C})} \leq \frac{\sum_{i=1}^{k} \kappa(\mathcal{C}_i, \mathcal{C}_i)}{\sum_{i=1}^{k} \kappa(\mathcal{F}, \mathcal{C}_i)} \leq \frac{\sum_{i=1}^{k} \kappa(\mathcal{F}, \mathcal{C}_i)}{\sum_{i=1}^{k} \kappa(\mathcal{F}, \mathcal{C}_i)} \leq \frac{\kappa(C_{i^*}, \mathcal{C}_{i^*})}{\kappa(\mathcal{F}, \mathcal{C}_{i^*})} \leq \rho(V, d). \tag{2.6}$$

By construction $(\mathcal{F}, C_{i^*}, f, d)$ is an instance for which an optimal solution with exactly one open facility exists.

Moreover, the facility location ratio is the same for the capacitated and the uncapacitated version of the facility location problem.
Lemma 2.3. If $\rho_{\text{cap}}(V,d)$ denotes the facility location ratio for the capacitated version of the problem, then $\rho_{\text{cap}}(V,d) = \rho(V,d)$.

Proof. First observe that any instance of the uncapacitated version is also an instance of the capacitated version by setting $u \equiv |C|$. Thus $\rho_{\text{cap}}(V,d) \geq \rho(V,d)$.

On the other hand, Lemma 2.2 is also true for the capacitated facility location problem. Thus we only need to consider capacitated instances where an optimum solution has only one facility. But in this case the capacity constraint is satisfied for all solutions. Thus the constraint can be omitted and we get an uncapacitated instance with the same optimal cost, both for facilities anywhere and facilities only on client positions. Hence we get $\rho_{\text{cap}}(V,d) \leq \rho(V,d)$. □

Due to this lemma, we only consider the uncapacitated version of the problem from now on. First, we establish an upper bound for the facility location ratio of general metrics.

Theorem 2.4. If $(V,d)$ is a metric space, then $\rho(V,d) \leq 2$.

Proof. Let $(V,d)$ be a metric space, $(C,F,f,d)$ with $C \subseteq F \subseteq V$ an instance of the uncapacitated facility location problem, and $F^* \subseteq F$ an optimum solution. By Lemma 2.2 we can assume w.l.o.g. that $|F^*| = 1$, say, $F^* = \{v\}$. Now let $s \in C$ be a client with $d(s,v) \leq d(s',v)$ for all $s' \in C$. If we select $s$ as the facility position, then the total cost is

$$c(\{s\}) = f + \sum_{s' \in C} d(s,s') \leq f + \sum_{s' \in C} (d(s,v) + d(v,s')) \leq f + 2 \sum_{s' \in C} d(v,s'). \quad (2.8)$$

But the cost of an optimal solution is $f + \sum_{s' \in C} d(v,s')$, so the ratio is bounded by 2. □

The bound is tight: To construct a metric space that satisfies $\rho(V,d) = 2$, let $V = F = \mathbb{N}$ with $d(0,i) = 1$ for $i \neq 0$ and $d(i,j) = 2$ for $i,j \neq 0$. Now consider for each $k \in \mathbb{N}$ the instance $C_k := \{1, \ldots, k\}$ with facility opening cost $f = 2$. The solution where one facility at position 0 is opened has cost $k+2$, while any solution where we are only allowed to open facilities at client positions has cost at least $2k$. For increasing $k$ this show $\rho(V,d) = 2$.

The following definition serves as an important tool to analyze the facility location ratio for given metric spaces.

Definition 2.5. A metric space $(V,d)$ is called $(\alpha, \beta)$-FL-optimal if for every instance $I = (C,F,f,d)$ of the uncapacitated facility location problem with $C \subseteq F \subseteq V$ and any optimal solution $F \subseteq F$ for which $|F|$ is minimal there
is a set $F^* \subseteq C$ of facilities on client positions such that
\begin{align}
c_{\text{fac}}(I, F^*) &\leq \alpha c_{\text{fac}}(I, F) \quad \text{and} \\
c_{\text{cl}}(I, F^*) &\leq \beta c_{\text{cl}}(I, F).
\end{align}

The proof of Theorem 2.4, in particular inequality (2.8), implies that any metric space is $(1, 2)$-FL-optimal. The definition is helpful because it can be used to analyze the facility location ratio of specific metric spaces.

**Lemma 2.6.** If a metric space $(V, d)$ is $(\alpha, 1)$-FL-optimal for $\alpha > 1$ and $(1, \beta)$-FL-optimal for $\beta > 1$, then $\rho(V, d) \leq \min\{\alpha, \beta, \alpha \beta - 1\}$. 

**Proof.** Obviously, $\rho(V, d) \leq \min\{\alpha, \beta\}$ holds. For an instance $(\mathcal{F}, C, f, d)$ of the uncapacitated facility location problem let $F \subseteq \mathcal{F}$ be an optimum solution where $|F|$ is minimum and $F' \subseteq C$ an optimum solution with facilities on client positions. As the metric space is $(\alpha, 1)$-FL-optimal and $(1, \beta)$-FL-optimal, we know that
\begin{align}
c(F') &\leq \min\{\alpha c_{\text{fac}}(F) + c_{\text{cl}}(F), c_{\text{fac}}(F) + \beta c_{\text{cl}}(F)\}.
\end{align}

If we substitute $x := c_{\text{fac}}(F)/c_{\text{cl}}(F)$, then the ratio (2.11) is at most equal to the infimum of $\min\{\alpha x + 1, x + \beta\}/(x + 1)$ over all $x \in \mathbb{R}_+$. This infimum is attained if $\alpha x + 1 = x + \beta$, that is, $x = (\beta - 1)/(\alpha - 1)$. We conclude $\rho(V, d) \leq (\alpha \beta - 1)/(\alpha + \beta - 2)$. \qed

### 2.3 The 2-dimensional $\ell_2$ Plane

For $k \in \mathbb{N} \cup \{\infty\}$ we consider the metric space $(\mathbb{R}^2, \ell_k)$. Given a finite set of clients $C$ and some $p \in \mathbb{R}^2$, we denote by $S_p(C) := \sum_{s \in C} \ell_k(p, s)$ the length of the star with center $p$. Moreover, we define $S_{\text{opt}}(C) := \min_{p \in \mathbb{R}^2} S_p(C)$ as the length of a minimum star spanning $C$ and $S_{\text{cl}}(C) := \min_{p \in C} S_p(C)$ as the length of the minimum star spanning $C$ with a client as its center. We call a point $p \in \mathbb{R}^2$ satisfying $S_{\text{opt}}(C) = S_p(C)$ a 1-median of $C$. Note that this point need not be unique in every metric space. We further define
\begin{equation}
\lambda^n_k := \sup_{C \subseteq \mathbb{R}^2, |C| = n} \frac{S_{\text{cl}}(C)}{S_{\text{opt}}(C)}
\end{equation}
for every $n \in \mathbb{N}$ and set $\lambda_k := \sup_{n \in \mathbb{N}} \lambda^n_k$.

Note that by definition $(\mathbb{R}^2, \ell_k)$ is $(1, \lambda_k)$-FL-optimal but not $(1, \lambda_k - \epsilon)$-FL-optimal for all $\epsilon > 0$. The next lemma shows that there exist instances with $n$ clients attaining $\lambda^n_k$ in which all stars centered on clients have the same length.
Lemma 2.7. For all \( n \in \mathbb{N} \) there is a set \( C^n \subset \mathbb{R}^2 \) with \( |C^n| = n \) such that

\[
\lambda^n_k = \frac{S_{cl}(C^n)}{S_{opt}(C^n)} \quad \text{and} \quad S_{s}(C) = S_{cl}(C) \quad \text{for all} \quad s \in C.
\]  

(2.13)

Proof. Consider some fixed \( n \in \mathbb{N} \). We first show that there exist instances \( C^n \) that satisfy (2.13). Let \( C_1, C_2, \ldots \) be a sequence of instances with \( |C_i| = n \) and

\[
\lim_{i \to \infty} \frac{S_{cl}(C_i)}{S_{opt}(C_i)} = \lambda^n_k.
\]  

(2.15)

By normalizing and translating the instances we can assume for all \( i \in \mathbb{N} \) that \( S_{opt}(C_i) = 1 \) and that \((0,0)\) is a 1-median of \( C_i \). But then \( C_i \subset B_k := \{ p \in \mathbb{R}^2 | \ell_k((0,0), p) \leq 1 \} \), that is, \( \{C_i\}_{i \in \mathbb{N}} \) is a sequence in a compact set. Thus there is a fixed point \( C^n \subseteq B_k \) of this sequence. As \( \ell_k \) is continuous,

\[
\frac{S_{cl}(C^n)}{S_{opt}(C^n)} = \lambda^n_k.
\]

By contradiction, we conclude that there is such an instance satisfying (2.14). In fact, we even show that all such instances \( C^n \) have this property. Again we assume that \((0,0)\) is a 1-median of \( C^n \). Suppose that there is a client \( s' \in C^n \) with \( S_{s'}(C^n) > S_{cl}(C^n) \). Let \( C^n_{s'} \) be the instance obtained from \( C^n \) by moving \( s' \) by \( \epsilon > 0 \) towards the origin. Then \( S_{opt}(C^n_{s'}) \leq S_{(0,0)}(C^n) = S_{opt}(C^n) - \epsilon \). Moreover, for all \( s \in C^n \) with \( s \neq s' \) the length of the star with center \( s \) decreases by at most \( \epsilon \), i.e.

\[
S_{s}(C^n_{s'}) \geq S_{s}(C^n) - \epsilon.
\]

If \( \epsilon \) is sufficiently small, we also have \( S_{s'}(C^n_{s'}) > S_{cl}(C^n_{s'}) \). We conclude

\[
\lambda^n_k = \frac{S_{cl}(C^n)}{S_{opt}(C^n)} < \frac{S_{cl}(C^n) - \epsilon}{S_{opt}(C^n) - \epsilon} \leq \frac{S_{cl}(C^n_{s'})}{S_{opt}(C^n_{s'})}.
\]

(2.16)

This contradicts the definition of \( \lambda^n_k \). \( \square \)

So it suffices to only consider instances where all stars centered on a client have the same length in order to prove that \((\mathbb{R}^2, \ell_k)\) is \((\alpha, 1)\)-FL-optimal.

We now show that if all clients in a given quadrant are considered, one can find a star centered on one of the clients that is shorter than the connections to the origin. In the proof we denote \( ||p|| := \ell_k((0,0), p) \).

Lemma 2.8. If \( C \subset \mathbb{R}^2_+ \setminus \{(0,0)\} \) is a non-empty finite set of clients with non-negative coordinates, then \( S_{(0,0)}(C) > S_{cl}(C) \).

Proof. We prove this lemma by contradiction, so suppose that \( C^* \) is a counterexample of minimum cardinality, that is,

\[
S_{(0,0)}(C^*) \leq S_{cl}(C^*).
\]  

(2.17)
If there would exist a proper subset $C' \subset C^*$ with $\sum_{s' \in C'} ||s - s'|| \leq \sum_{s' \in C'} ||s'||$ for all $s \in C^*$, then for all $s \in C^*$

\begin{align*}
S_{(0,0)}(C^* \setminus C') &= S_{(0,0)}(C^*) - \sum_{s' \in C'} ||s'|| \\
&\leq S_s(C^*) - \sum_{s' \in C'} ||s - s'|| \\
&= S_s(C^* \setminus C'),
\end{align*}

in other words $S_{(0,0)}(C^* \setminus C') \leq S_{cl}(C^* \setminus C')$. Since this is a contradiction to the minimality of $|C^*|$, we know that for every $C' \subset C^*$ there exists an $s \in C^*$ such that

\begin{equation}
\sum_{s' \in C'} ||s - s'|| > \sum_{s' \in C'} ||s'||.
\end{equation}

Let $s^t = (s^t_x, s^t_y)$ be the topmost and $s^r = (s^r_x, s^r_y)$ the rightmost client in $C^*$. If $s^t = s^r$ then $||s^t - s|| \leq ||s^t||$ for all $s \in C^*$, contradicting (2.21) for $C' = \{s^t\}$. Consequently, $s^t \neq s^r$.

Moreover, we define the three points $v' := (0, s^t_y), w' := (s^r_x, 0)$, and $v'' := (s^r_x, s^t_y)$, and the set $X := \{(0,0), v', v'', w'\}$. The definitions are illustrated in Figure 2.1. We claim that for every $p \in X$

\begin{equation}
||s^t - p|| + ||s^r - p|| \leq ||s^t|| + ||s^r||
\end{equation}

holds. For $(0,0)$ this is trivial. For $v''$ it follows from the fact that $v''$ is in the convex hull of $\{(0,0), s^t, s^r, s^t + s^r\}$ and from the convexity of the left hand side of (2.22).
For $v'$, using $w'' := (s^t_x, 0)$, we obtain

$$||s^t - v'|| + ||s^r - v'|| \leq ||s^t - v'|| + ||w' - v'|| \leq ||s^t - v'|| + ||v' - w''|| + ||w'' - w'|| \leq ||s^t|| + ||w'||$$

Equations (2.23) to (2.27)

By symmetry, (2.22) is also true for $w'$. However, all $s \in C^*$ are in the convex hull of $X$, thus also satisfying (2.22) by the convexity of its left hand side. But this fact contradicts (2.21) for $C' = \{s^t, s^r\}$, finishing the proof.

Applying this lemma once in every quadrant gives an upper bound for the FL-optimality of the $\ell_k$ plane.

**Lemma 2.9.** For all $k \in \mathbb{N} \cup \{\infty\}$ the plane $(\mathbb{R}^2, \ell_k)$ is $(4,1)$-FL-optimal.

To sum up, $(\mathbb{R}^2, \ell_k)$ is $(4,1)$-FL-optimal and $(1,2)$-FL-optimal. By Lemma 2.6 this implies a bound for the facility location ratio.

**Lemma 2.10.** For all $k \in \mathbb{N} \cup \{\infty\}$ $\rho(\mathbb{R}^2, \ell_k) \leq 1.75$ holds.

Experiments indicate that in all instances in all spaces $(\mathbb{R}^2, \ell_k)$ it is always sufficient to choose two clients as centers such that the resulting stars are not longer than an optimal star where the center is a 1-median.

**Conjecture 2.11.** For all $k \in \mathbb{N} \cup \{\infty\}$ the plane $(\mathbb{R}^2, \ell_k)$ is $(2,1)$-FL-optimal. By Lemma 2.6, this would imply $\rho(\mathbb{R}^2, \ell_k) \leq 1.5$.

### 2.4 The Euclidean Plane

For the two-dimensional Euclidean plane, Ismailescu and Park [34] were able to prove the following property:

**Lemma 2.12.** The Euclidean plane $(\mathbb{R}^2, \ell_2)$ is $(1, 1.3546)$-FL-optimal.

Combining this fact with Lemma 2.6 and Lemma 2.9 immediately yields an upper bound for the facility location ratio.

**Corollary 2.13.** The facility location ratio of the Euclidean plane $\rho(\mathbb{R}^2, \ell_2)$ is at most 1.3172.
To obtain a lower bound for $\rho(R^2, \ell_2)$, consider the instances $C^n$ for $n \in \mathbb{N}$ which contain the vertices of a regular $n$-gon arranged on a unit circle. The sum of distances from one of these vertices to all other vertices approaches $\frac{4n}{\pi}$ for sufficiently large $n$ [53]. Connecting all vertices to the origin yields a shortest Steiner star, which has a length of $n$. Hence, $(R^2, \ell_2)$ is not $(1, \beta)$-FL-optimal for $\beta < \frac{4}{\pi}$, but it may be $(1, \frac{4}{\pi})$-FL-optimal.

Moreover, taking two opposite vertices of $C^n$ as centers, the sum of lengths of the two resulting stars does not exceed $n$. This means that $(R^2, \ell_2)$ is not $(\alpha, 1)$-FL-optimal for $\alpha < 2$, but it may be $(2, 1)$-FL-optimal. Using these values of $\alpha$ and $\beta$ in Lemma 2.6 gives a lower bound for the facility location ratio.

**Corollary 2.14.** The facility location ratio of the Euclidean plane $\rho(R^2, \ell_2)$ is at least $1.2146$.

We think that $C^n$ is in fact a class of worst-case instances in the sense that this lower bound already equals the exact facility location ratio.

**Conjecture 2.15.** The Euclidean plane $(\mathbb{R}^2, \ell_2)$ is $(1, \frac{4}{\pi})$-FL-optimal and $(2, 1)$-FL-optimal and thus, by Corollary 2.14, $\rho(\mathbb{R}^2, \ell_2) = 2 - \frac{\pi}{4} \approx 1.2146$.

The first part of the conjecture was mentioned in [13], the second part has, to our knowledge, not appeared before.

### 2.5 The Rectilinear Plane

In this section we examine the special case that the metric is the rectilinear plane $(\mathbb{R}^2, \ell_1)$ with $\ell_1((x_1, y_1), (x_2, y_2)) := |x_1 - x_2| + |y_1 - y_2|$. We show that this metric is $(3, 1)$-FL-optimal as well as $(1, 1.5)$-FL-optimal. By Lemma 2.6 our main result follows.

**Theorem 2.16.** The facility location ratio of the rectilinear plane $\rho(\mathbb{R}^2, \ell_1)$ is at most $1.4$.

In this section we denote by $(s_x, s_y) \in \mathbb{R}^2$ the position of a client $s \in C$. For simplicity of notation, we identify a client with its position and vice versa when no confusion can arise. Nevertheless, note that more than one client of $C$ might lie at the same position, so $C$ can be seen as a multiset of points in $\mathbb{R}^2$.

The first part of the result is due to Fekete and Meijer [15]:

**Theorem 2.17.** The rectilinear plane $(\mathbb{R}^2, \ell_1)$ is $(1, 1.5)$-FL-optimal.
Thus it remains to show (3,1)-FL-optimality. We have already proven the (4,1)-FL-optimality for the general case \( (\mathbb{R}^2, \ell_k) \) in Lemma 2.9. Here we establish an alternative proof for the \( \ell_1 \) metric that will be extended in order to show that the rectilinear plane is even (3,1)-FL-optimal.

We assume w.l.o.g. that \( m = (0,0) \) is a 1-median of \( C \), that is, \( S_m(C) = S_{\text{opt}}(C) \). We can further assume that there exists no \( s \in C \) with \( s_x = s_y = 0 \) because otherwise \( S_{\text{opt}}(C) = S_{\text{cl}}(C) \) would hold. Now consider the four regions

\[
\begin{align*}
R_l & := \{(x, y) \in \mathbb{R}^2 \mid x \leq 0, \text{ and } -y \leq x \leq y \}, \\
R_r & := \{(x, y) \in \mathbb{R}^2 \mid x \geq 0, \text{ and } -x \leq y < x \}, \\
R_u & := \{(x, y) \in \mathbb{R}^2 \mid y \geq 0, \text{ and } x < y \leq -y \}, \text{ and} \\
R_b & := \{(x, y) \in \mathbb{R}^2 \mid y \leq 0, \text{ and } -x \leq y \leq x \},
\end{align*}
\]

as well as four distinguished clients \( v_l \), \( v_r \), \( v_u \) and \( v_b \) which are chosen by Algorithm 1. These clients satisfy some simple properties:

**Lemma 2.18.** The relations \( v_l^t \leq v_b^t \leq v_x^t \leq v_u^t \), \( v_l^t \leq v_x^t \leq v_r^t \leq v_b^t \leq v_y^t \), \( v_y^t \leq v_r^t \leq v_b^t \) hold.

**Proof.** We prove the claim by induction on the number of iterations of the while-loop. Obviously, the relations are satisfied before the first iteration. W.l.o.g. assume in the \( k \)-th iteration \( v^r = w_k \) was set. Then \( v_b^t \leq v_y^t \leq v_l^t \) is satisfied by the choice of \( w_k \). By symmetry it suffices to show \( v_x^t \leq v_u^t \). If \( v^t = (0, \infty) \) then \( v_l^t = 0 \leq v_x^t \) as \( v^r \in R_\ell \). Otherwise, \( v^t = w_{k'} \) was set in an iteration \( k' < k \). This means \( ||w_{k'}||_\infty \leq ||w_k||_\infty \), implying \( v_x^t \leq ||w_{k'}||_\infty \leq ||w_k||_\infty = \max(\{|v^t_x|, |v^t_y|\}) = v_x^t \) as \( v^r \in R_r \). This completes the proof. \( \square \)

By the choice of \( v_l^t, v_r^t, v_u^t \) and \( v_b^t \) we have \( C = C_l \cup C_r \cup C_u \cup C_b \).

**Lemma 2.19.** The rectilinear plane is \((4,1)-\text{FL-optimal}.\)

**Proof.** First recall that by construction \( s_x \geq v_x^r \) holds for each \( s \in C_r \). So for \( s \in C_r \) we get \( \ell_1(s, m) = s_x + |s_y| \geq s_x - v_x^r + |v_x^r| + |s_y| \geq s_x - v_x^r + |s_y - v_y^r| = \ell_1(s, v^r) \). Analogous inequalities hold for \( C_l, C_b \) and \( C_i \), and summing up yields

\[
\sum_{s \in C} \ell_1(s, m) \geq \sum_{s \in C_l} \ell_1(s, v^l) + \sum_{s \in C_r} \ell_1(s, v^r) + \sum_{s \in C_u} \ell_1(s, v^u) + \sum_{s \in C_b} \ell_1(s, v^b).
\]

Note that if \( v^x \) is an artificial client for \( x \in \{l, r, b, t\} \), that is, a client in the set \( \{(-\infty, 0), (\infty, 0), (0, -\infty), (0, \infty)\} \), then \( C_x = \emptyset \). Thus using
Input: Set of clients $\mathcal{C}$ with median at $(0,0)$.
Output: Clients $v^l, v^r, v^b, v^t \in \mathcal{C} \cup \{(-\infty, 0), (\infty, 0), (0,\infty), (0,-\infty)\}$ and partitioning $\mathcal{C}_l \cup \mathcal{C}_r \cup \mathcal{C}_b$ of $\mathcal{C}$.

1. $v^l \leftarrow (-\infty, 0)$, $v^r \leftarrow (\infty, 0)$, $v^t \leftarrow (0,\infty)$ and $v^b \leftarrow (0,-\infty)$;
2. $k \leftarrow 0$;
3. while $\exists v \in ((v^l_x, v^r_x) \times (v^b_y, v^t_y)) \cap \mathcal{C}$ do
   4. $k \leftarrow k + 1$;
   5. $w_k \leftarrow \arg\min_{s \in ((v^l_x, v^r_x) \times (v^b_y, v^t_y)) \cap \mathcal{S}} ||s||_\infty$;
   6. $v^\times \leftarrow w_k$ for $x \in \{r, l, t, b\}$ such that $w_k \in R_x$;
end

$\mathcal{C}_l \leftarrow \{s \in \mathcal{C} \mid s_x \leq v^l_x\}$;
$\mathcal{C}_r \leftarrow \{s \in \mathcal{C} \setminus \mathcal{C}_l \mid s_x \geq v^r_x\}$;
$\mathcal{C}_t \leftarrow \{s \in \mathcal{C} \setminus (\mathcal{C}_l \cup \mathcal{C}_r) \mid s_y \geq v^t_y\}$;
$\mathcal{C}_b \leftarrow \{s \in \mathcal{C} \setminus (\mathcal{C}_l \cup \mathcal{C}_r \cup \mathcal{C}_t) \mid s_y \leq v^b_y\}$;

Algorithm 1: Selection of four distinguished clients and partitioning of $\mathcal{C}$.

Theorem 2.20. The rectilinear plane is $(3,1)$-FL-optimal.

Proof. We start with the assignment of clients to $v^l, v^r, v^b, v^t$ as in the proof of Lemma 2.19 and try to redirect clients from one of those facilities to another one. If we want to omit, say, $v^t$, we have to reassign all $s \in \mathcal{C}_t$ to $v^l$, 

Note that some clients may not be contained in any of these sets. Figure 2.2 illustrates the setup.
We can assign all clients \( s \in C \) with \( s_x \leq v_x^l \) to \( v^l \) and all clients \( s \in C \) with \( s_x \geq v_x^r \) to \( v^r \) and the connection cost of these clients to their new facilities is not higher than their connection cost to \( m \). Thus it remains to redistribute the clients in \( D_1 \cup D_2 \).

If there are clients \( s \in D_1 \cup D_2 \) and \( s' \in C_1 \), then we can assign \( s \) to \( v^l \), as \( \ell_1(s, v^l) + \ell_1(s', v^l) = \ell_1(s, m) + \ell_1(s', m) \) (observe that \( s_x' < v_x^l \) by Lemma 2.18). Similarly, two clients \( s \in D_1 \cup D_2 \) and \( s' \in A_2 \) can be assigned to \( v^r \) without exceeding the cost of connecting both clients to \( m \). So if \( |D_1| + |D_2| \leq |C_1| + |A_2| \) holds, all \( s \in D_1 \cup D_2 \) can be reassigned to \( v^r \) and \( v^l \). In the same way we can assign clients in \( A_1 \cup A_2 \) to \( v^l \) and \( v^b \) if \( |A_1| + |A_2| \leq |D_1| + |B_2| \), and so on.

Consequently, if this reassignment method fails for all 4 possible choices of the facility we want to omit, then

\[
|D_1| + |D_2| > |C_1| + |A_2|, \quad |A_1| + |A_2| > |D_1| + |B_2|,
\]

\[
|B_1| + |B_2| > |A_1| + |C_2|, \quad |C_1| + |C_2| > |B_1| + |D_2|
\]

must hold. Summing up these four inequalities results in a contradiction, so either \( v^l \), \( v^r \), \( v^b \) or \( v^t \) does not have to be opened.

This finishes the proof of Theorem 2.16.

For a lower bound consider the client set \( C = \{(1,0), (-1,0), (0,1), (0,-1)\} \). Setting the facility opening cost to 2, an optimal solution of the facility location problem in the rectilinear plane for \( C \) has cost 6, while an optimal solution for \( C \) with facility positions restricted to the client positions has cost 8.

**Corollary 2.21.** The facility location ratio of the rectilinear plane \( \rho(\mathbb{R}^2, \ell_1) \) is at least \( 4/3 \).
We think that this example is already the worst case that can occur.

**Conjecture 2.22.** The rectilinear plane $(\mathbb{R}^2, \ell_1)$ is $(2,1)$-FL-optimal. Using Theorem 2.17 and Corollary 2.21, this would imply $\rho(\mathbb{R}^2, \ell_1) = \frac{4}{3}$.

### 2.6 Conclusion and Future Work

<table>
<thead>
<tr>
<th>Metric</th>
<th>Upper bounds</th>
<th>Lower bounds</th>
<th>Conjectures</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(\mathbb{R}^2, \ell_1)$</td>
<td>$3$</td>
<td>$1.5$</td>
<td>$\frac{4}{3}$</td>
</tr>
<tr>
<td>$(\mathbb{R}^2, \ell_2)$</td>
<td>$4$</td>
<td>$1.3546$</td>
<td>$\frac{4}{7}$</td>
</tr>
<tr>
<td>$(\mathbb{R}^2, \ell_k)$</td>
<td>$4$</td>
<td>$2$</td>
<td>$2$</td>
</tr>
</tbody>
</table>

Table 2.1: Summary of the results, including proven upper bounds, lower bounds and the conjectured exact values

In this chapter we introduced the facility location ratio as an analog to the well-studied Steiner ratio related to the uncapacitated and capacitated metric facility location problem. We then presented tools to analyze this property of a metric space and provided upper and lower bounds on the facility location ratio for general metrics as well as several special cases in the two-dimensional plane. Moreover, we formulated conjectures for the exact values of these ratios. Table 2.6 summarizes these results. It lists bounds for the smallest numbers $\alpha$ and $\beta$ such that the metric spaces are $(\alpha, 1)$-FL-optimal and $(1, \beta)$-FL-optimal as well as the facility location ratio implied by these values.

Future work may include research on higher-dimensional metric spaces and closing the gaps between the bounds. It is also possible to study the relation between the facility location ratio and the related Steiner ratio. For example, both values are in general bounded by 2 (c.f. E. F. Moore [21] and Theorem 2.4).
Chapter 3

Steiner Tree Embedding

In this chapter we face the first problem motivated by repeater tree design. As we have seen in the Chapter 1, repeater trees can be modeled by rectilinear Steiner trees. Given a set of terminals and a distinguished vertex \( r \) we want to construct a rectilinear Steiner tree connecting them. To model the timing constraints we use an individual length restriction for each terminal and we require that the length of the unique root-terminal-path does not exceed those terminal’s length restriction. The problem of building such a tree is NP-complete as it contains the Steiner tree problem. Nevertheless, there are several heuristics to solve the problem in practice.

Our main focus in this chapter is to study how to improve a given solution for this problem by only moving the Steiner points. More precisely, we are given the terminals containing a distinguished root, their positions, the length restrictions and the topology of a Steiner tree on the terminals. Now the task is to compute positions for the Steiner points minimizing the total length of the tree such that the length restrictions are satisfied. In other words we are looking for a feasible embedding of the Steiner points of a given tree.

We show that the problem can be formulated as a linear program. Therefore the problem can be solved in polynomial time using methods like the ellipsoid method \([40, 60]\). In practice linear programs are solved using implementations of the simplex method \([12]\). Nevertheless, we are interested in combinatorial algorithms to solve the problem. On the one hand such problem-specific algorithms can be much faster than the simplex method. On the other hand they often lead to a deeper understanding of the problem and its solutions. In this chapter we analyze the structure of optimal Steiner tree embeddings considering length restrictions and present the first polynomial-time combinatorial algorithm computing those.

An extended abstract containing the main results of this chapter will be published in the Proceedings of the 21st International Computing and
3.1 Problem Formulation

The Rectilinear Steiner Tree Problem With Given Topology And Length Restrictions can be stated as follows. The input \((S, T, r, p, l)\) consists of a set of terminals \(T\) with positions \(p : T \rightarrow \mathbb{R}^2\), a tree \(S\) with \(T \subseteq V(S)\), a distinguished terminal \(r \in T\) - called the root of the tree - and length restrictions \(l : T \rightarrow \mathbb{R}_{\geq 0}\).

The task is to find an embedding \(\pi : V(S) \rightarrow \mathbb{R}^2\) of the vertices of the tree into the plane with \(\pi(t) = p(t)\) for all \(t \in T\), such that for all \(t \in T\) the length \(d_\pi(t)\) of the unique path from \(r\) to \(t\) in \(S\) with edge set \(E_S[r, t]\) has length at most \(l(t)\), that is,

\[
d_\pi(t) = \sum_{\{v, w\} \in E_S[r, t]} ||\pi(v) - \pi(w)||_1 \leq l(t) \tag{3.1}
\]

and the total length

\[
c(\pi) := \sum_{\{v, w\} \in E(S)} ||\pi(v) - \pi(w)||_1 \tag{3.2}
\]

of the tree is minimized. Throughout this chapter we assume w.l.o.g. that the root is placed at the origin, that is, \(p(r) = (0, 0)\). By adding Steiner points and edges of length zero we can assume that the terminals are leaves of \(S\) and that all Steiner points have degree 3. Moreover, we denote by \(\pi_x(v)\) and \(\pi_y(v)\) the \(x\)- and \(y\)-coordinate, respectively, of \(\pi(v)\) for an embedding \(\pi\) and a vertex \(v \in V(S)\).

A further generalization of the problem is to extend it to other metrics or to consider length restrictions between any pair of terminals. In this chapter we restrict ourselves to the \(\ell_1\) metric and length restrictions between one distinguished vertex and all other terminals, due to its application in repeater tree design.

If we consider the problem of embedding a rectilinear Steiner tree without length restrictions an optimal embedding can be computed in linear time using dynamic programming (see e.g. [39]). The problem of embedding a Steiner tree with a given topology satisfying length restrictions has not been studied yet. In this chapter we present the first combinatorial polynomial time algorithm that computes an optimal embedding.

Figure 3.1 (i) shows an instance with seven terminals drawn as black squares and 5 Steiner points drawn as white circles. Figure (ii) shows an
optimal solution if there are no length restrictions. In Figure (iii) an optimal solution is shown, if we have length restrictions \( l(t_1) = 5 \), \( l(t_2) = 6 \) and \( l(s) = \infty \) otherwise. If there are no length restrictions, then there always exists an optimal solution where the Steiner points are positioned at vertices of the Hanan grid on \( T \). With length restrictions, this is no longer true. In this case we prove that if the positions of the terminals and the length restrictions are integral, then there always exists a solution on half-integral positions.

![Figure 3.1: Instance (i), optimal embedding without length restrictions (ii) and optimal embedding with length restrictions \( l(t_1) = 5 \) and \( l(t_2) = 6 \) (iii). The regular dotted grid has a lattice spacing of 1.](image)

In Section 3.2 we give two linear program formulations for the studied problem. After introducing several definitions concerning the movement of components of the tree in Section 3.3, we present our main observations in Section 3.4. Among others, we prove that there always exists an optimal embedding where the Steiner points are on half-integral positions. Based on this observation, we introduce in Section 3.5 a dynamic programming algorithm which is the main ingredient to achieve a pseudo-polynomial time algorithm. Refining this algorithm we finally gain a polynomial time algorithm in Section 3.6.

### 3.2 Linear Programs

In this section we give two LP formulations for the problem proving that it can be solved in polynomial time using methods like the ellipsoid method or the inner point method. The first LP is a straightforward extension of the linear programs as discussed for example in Jiang and Wang [39] or Brenner and Vygen [7]. It contains for every edge \( e \) four variables \( x_{e1}, x_{e2}, y_{e1}, y_{e2} \) and for each vertex \( v \) two variables \( x_v, y_v \) and is stated as follows:
minimize $\sum_{e \in E(S)} x_e^2 - x_e^1 + y_e^2 - y_e^1$

subject to $x_e^1 \leq x_v \leq x_e^2$, $\forall v \in V(S), e \in \delta(v)$ (3.3)

$y_e^1 \leq y_v \leq y_e^2$, $\forall v \in V(S), e \in \delta(v)$ (3.4)

$\sum_{e \in E(S)} x_e^2 - x_e^1 + y_e^2 - y_e^1 \leq l(t)$, $\forall t \in T$ (3.5)

$(x_t, y_t) = p(t)$, $\forall t \in T$ (3.6)

Let $(\{x_e^1, x_e^2, y_e^1, y_e^2\}_{e \in E(S)}, \{x_v, y_v\}_{v \in V(S)})$ be an optimal solution of the linear program. We define an embedding $\pi : V(S) \rightarrow \mathbb{R}^2$ by setting $\pi(v) = (x_v, y_v)$ for all $v \in V(S)$. The equations (3.6) ensure that the terminals remain on their original positions. Note that for every edge $e = (v, w) \in E(S)$ we have $x_e^1 = \min\{x_v, x_w\}, x_e^2 = \max\{x_v, x_w\}, y_e^1 = \min\{y_v, y_w\}$ and $y_e^2 = \max\{y_v, y_w\}$ and therefore

$$||\pi(v) - \pi(w)||_1 = x_e^2 - x_e^1 + y_e^2 - y_e^1.$$ (3.7)

Thus the objective is to minimize the total length of the embedding. The constraints (3.5) ensure that the length restrictions are satisfied.

On the other hand, if we have a shortest, feasible embedding $\pi : V(S) \rightarrow \mathbb{R}^2$, then setting $x_v = \pi_x(v), y_v = \pi_y(v)$ for $v \in V(T), x_e^1 = \min\{x_v, x_w\}, x_e^2 = \max\{x_v, x_w\}, y_e^1 = \min\{y_v, y_w\}$ and $y_e^2 = \max\{y_v, y_w\}$ for $e = (v, w) \in E(S)$ gives a feasible solution of the linear program.

In the second LP we have just one variable $d_e$ for every edge $e \in E(S)$ where $d_e$ will be the length of edge $e$. The linear program contains more constraints than the first one, but it better reflects the structure of the problem:

minimize $\sum_{e \in E(S)} d_e$

subject to $\sum_{e \in E_S[s,t]} d_e \geq ||p(s) - p(t)||_1$, $\forall s, t \in T$ (3.8)

$\sum_{e \in E_S[r,t]} d_e \leq l(t)$, $\forall t \in T$ (3.9)

$d_e \geq 0$, $\forall e \in E(S)$. (3.10)

Now we have to show, that this LP models our problem.
Lemma 3.1. If \( \pi \) is a feasible embedding, then \( \{d_e\}_{e \in E(S)} \), defined by 
\[ d(v,w) = ||\pi(v) - \pi(w)||_1 \]
for all \( (v,w) \in E(S) \) is a feasible solution of the LP. If, on the other hand, \( \{d_e\}_{e \in E(S)} \) is a feasible solution of the LP, then there exists a feasible embedding \( \pi \) with 
\[ ||\pi(v) - \pi(w)||_1 \leq d(v,w) \]
for all edges \( (v,w) \in E(S) \) and this embedding can be computed efficiently.

Proof. The first part of the lemma is obvious: As \( \pi \) is a feasible embedding and in each embedding the length of an \( s\!-\!t \)-path for \( s, t \in T \) is at least 
\[ ||p(s) - p(t)||_1 \]
the constraints of the LP are satisfied.

To prove the second part, let \( \{d_e\}_{e \in E(S)} \) be an optimal solution of the LP. We prove the claim by induction on the number of Steiner points. If there is no Steiner point, then \( S \) contains only one edge \( e = (r,t) \) and thus 
\[ \pi(v) = p(v) \]
for all \( v \in V(S) \). By (3.9) and (3.8) we have 
\[ l(t) \geq d_e \geq ||p(r) - p(t)||_1 = ||\pi(r) - \pi(t)||_1 \]
Thus \( \pi \) is a feasible embedding.

Now assume \( S \) is a Steiner tree with \( n \) Steiner points. Let \( v \in V(S) \setminus T \) be a Steiner point with maximum distance to \( r \) in terms of the number of edges and denote by \( T_v \) the set of terminals \( t \in T \) that contain \( v \) on the unique \( r\!-\!t \)-path. By construction all terminals in \( T_v \) are neighbors of \( v \) and \( v \) has only one neighbor outside \( T_v \).

We want to compute a feasible position \( \pi(v) \) for \( v \). To this end observe that for each terminal \( t \) the distance between \( \pi(v) \) and \( p(t) \) has to be at most 
\[ z_t := \sum_{e \in E_S[t,v]} d_e \]
Thus \( \pi(v) \) must lie in \( B_t = \{ p \in \mathbb{R}^2 : ||p - p(t)||_1 \leq z_t \} \), which is the \( \ell_1 \)-ball with radius \( z_t \) around \( p(t) \). Geometrically, \( B_t \) is a square whose sides are parallel to the diagonals \( \{(x,x) : x \in \mathbb{R} \} \) and \( \{(x,-x) : x \in \mathbb{R} \} \). For any two sets \( B_t \) and \( B_s \), \( s \neq t \), we have 
\[ B_s \cap B_t \neq \emptyset \]
as
\[ \sum_{e \in E_S[s,t]} d_e \geq \sum_{e \in E_S[s,v]} d_e + \sum_{e \in E_S[v,t]} d_e = z_s + z_t \geq ||p(s) - p(t)||_1. \] (3.11)

Thus any pair \( B_s, B_t \) of squares has a non-empty intersection. But then by Helly’s Theorem [29] the intersection of all squares is non-empty, that is,
\[ \bigcap_{t \in T} B_t \neq \emptyset \] (3.12)
and therefore we can set the position \( \pi(v) \) of \( v \) to one of the points in this intersection. Note that by construction for all \( t \in T_v \), the length of the edge \( (v,t) \) is at most \( d(v,t) = z_t \).

In a straightforward way we construct a new instance \( I' = (S', T', r, p', l') \) for the linear program. Let \( T' = (T \cup T_v) \cup \{v\}, p'(t) = p(t) \) and \( l'(t) = l(t) \) for \( t \in T' \setminus T, p'(v) = \pi(v), l'(v) = \min_{t \in T_v} (l(t) - d(v,t)) \) and let \( S' \) be the Steiner tree we obtain from \( S \) by removing the vertices in \( T_v \) and their incident
edges. By construction \( \{d_e\}_{e \in E(S')} \) is a feasible solution of the LP for \( I' \). As \( S' \) contains one Steiner point less than \( S \) we can apply the induction hypothesis and get a feasible embedding \( \pi' \) with \( ||\pi'(\tilde{v}) - \pi'(\tilde{w})||_1 \leq d_{(\tilde{v}, \tilde{w})} \) for all edges \( (\tilde{v}, \tilde{w}) \in E(S') \). By setting \( \pi(\tilde{v}) = \pi'(\tilde{v}) \) for \( \tilde{v} \in V(S) \setminus \{v\} \) we get a feasible embedding \( \pi \) for \( S \). \( \Box \)

Note that this LP has \( O(n^2) \) constraints while the first one only has \( O(n) \). Nevertheless the second LP gives a better insight into the problem: We are dealing with paths between pairs of terminals. For each of these paths a lower bound on the length is given. Moreover, for some of them we have additional upper bounds on their length.

The problem can easily be extended to deal with length restrictions between any pair of terminals and not just between root and terminals. This extended version of the problem can still be solved in polynomial time as we can extend the presented linear programs by adding additional length constraints. Nevertheless, due to their practical relevance, we only consider length restrictions between the root and the other terminals throughout this chapter.

### 3.3 Moving Components

Before we come to the main observations of the chapter we examine how the movements of Steiner points of a given embedding influence the total length of the tree and the length of root-terminal-paths. First we start with several definitions that we need throughout this chapter.

If \( \pi \) is an embedding, then an \textit{x-component} \( C \) at position \( x(C) \) with respect to \( \pi \), \( x(C) \in \mathbb{R} \), is a connected subtree \( C \) of \( T \) such that all vertices in \( C \) have \( x \)-coordinate \( x(C) \). An \textit{x-component} \( C \) is called \textit{maximal} if there does not exist any \textit{x-component} \( C' \) with \( C \subsetneq C' \). A component always depends on the embedding \( \pi \). In the following, we omit \( \pi \) in the notation if it is clear from the context. In an analogous way we define a \textit{y-component} \( C \) at position \( y(C) \). In the remainder of the chapter we introduce several definitions and state lemmata concerning \textit{x-components}. By symmetry, these definitions and lemmata also hold for \textit{y-components}.

Let \( \Gamma(V(C)) \) be the neighbors of the vertices of \( C \). For an \textit{x-component} \( C \) we define

\[
\Gamma_{\leq}(C) := \{v \in \Gamma(V(C)) : \pi_x(v) < x(C)\} \quad \text{and} \quad \Gamma_{\geq}(C) := \{v \in \Gamma(V(C)) : \pi_x(v) > x(C)\}. \tag{3.13}
\]

In an analogous way we define \( \Gamma_{\leq}(C) \) and \( \Gamma_{\geq}(C) \) for a \textit{y-components} \( C \). If \( C \) is a component not containing \( r \), then the \textit{predecessor} of \( C \) is the unique
vertex \( v \in \Gamma_\pi^>(C) \cup \Gamma_\pi^<(C) \) such that \( v \) is on the root-\( w \)-path for all \( w \in V(C) \).

For simplicity of notation we define

\[
\text{sign}(C) = \begin{cases} 
1 & \text{if the predecessor of } C \text{ is in } \Gamma_\pi^<(C) \\
-1 & \text{otherwise.}
\end{cases}
\]  

(3.15)

If \( C \) is an \( x \)-component with respect to some embedding \( \pi \) then we say that we move \( C \) by \( \delta \) if we replace \( \pi \) by the embedding \( \pi' \) defined by

\[
\pi'(v) := \begin{cases} 
\pi(v) + (\delta, 0) & \text{for all } v \in V(C) \setminus T, \\
\pi(v) & \text{otherwise.}
\end{cases}
\]  

(3.16)

We say, that we move \( C \) towards its predecessor if \( \delta \cdot \text{sign}(C) < 0 \).

If \( C \) is a maximal component containing no terminals, then we define \( R(C) \) to be the set of terminals \( t \) such that the unique root-\( t \)-path \( P \) passes \( C \), that is, \( V(P) \cap V(C) \neq \emptyset \) and the path enters and leaves \( C \) at the same side, that is, we have either \( |V(P) \cap \Gamma_\pi^>(C)| = 2 \) or \( |V(P) \cap \Gamma_\pi^<(C)| = 2 \). If we choose \( \delta \in \mathbb{R} \) with \( |\delta| \) small enough and move \( C \) by \( \delta \), then the length of all root-\( t \)-paths with \( t \in R(C) \) change by \( 2 \text{sign}(C)\delta \). The length of any other root-terminal-path does not change.

Figure 3.2: An embedding \( \pi \) with a maximal \( y \)-component \( C \) with \( V(C) = \{s_1, s_3, s_4\} \), predecessor \( s_2, \Gamma_\pi^>(C) = \{t_2, s_5\}, \Gamma_\pi^<(C) = \{t_1, s_2, t_6\}, \text{sign}(C) = 1 \) and \( R(C) = \{t_1, t_6\} \) (i). Embedding obtained by moving \( C \) by \( \delta < 0 \). The new embedding preserves the local order of \( \pi \) (ii). The length of all root-\( t \) with \( t \in R(C) \) changed by \( 2 \text{sign}(C)\delta = 2\delta \) (< 0).

Figure 3.2 illustrates some of the definitions.

If \( \pi \) and \( \pi' \) are two embedding, then we say that \( \pi' \) preserves the local order of \( \pi \) if for every edge \( (v, w) \in E(S) \) we have

\[
(\pi_x(v) \leq \pi_x(w)) \implies (\pi'_x(v) \leq \pi'_x(w)) \quad \text{and} \quad (3.17)
\]

\[
(\pi_y(v) \leq \pi_y(w)) \implies (\pi'_y(v) \leq \pi'_y(w)). \quad (3.18)
\]
Note, that by (3.17) if \( \pi_x(v) = \pi_x(w) \) then \( \pi'_x(v) = \pi'_x(w) \) and analogously for \( y \). This implies that each component with respect to \( \pi \) is also a component with respect to \( \pi' \) (but not necessarily the other way round!). Moreover, if \( v \) is a vertex of an \( x \)-component that contains terminals, we have \( \pi_x(v) = \pi'_x(v) \).

Now we can analyze how the length of the embedding and of root-terminal-paths change if we move maximal components simultaneously and the local order is preserved.

**Lemma 3.2.** Let \( \pi \) be an embedding, \( \Delta \) be the set of all maximal \( x \)- and \( y \)-components, and \( \delta_C \in \mathbb{R} \) for \( C \in \Delta \). Denote by \( \pi' \) the embedding we obtain by moving each component \( C \in \Delta \) by \( \delta_C \). If \( \pi' \) preserves the local order of \( \pi \) then

\[
c(\pi') = c(\pi) + \sum_{C \in \Delta} \delta_C \left( |\Gamma_<(\pi)(C)| - |\Gamma_>(\pi)(C)| \right). \tag{3.19}
\]

Moreover we have for all \( t \in T \):

\[
d_{\pi'}(t) = d_{\pi}(t) + \sum_{C \in \Delta : t \in R(C)} 2 \text{sign}(C) \cdot \delta_C. \tag{3.20}
\]

**Proof.** Consider an \( x \)-component \( C \in \Delta_x \). If we move \( C \), then only the length of edges \( \{v, w\} \in E(S) \) with \( v \in V(C) \) and \( w \notin V(C) \) are changed. Let \( \{v, w\} \) be such an edge and assume \( w \in \Gamma_<(\pi)(C) \), that is \( \pi_x(w) < \pi_x(v) \). As the local order is preserved, we have \( \pi'_x(w) \leq \pi'_x(v) \). But then moving \( C \) by \( \delta \) increases the length of the edge \( \{v, w\} \) by \( \delta \). In an analogous way we see that the length of the edge decreases by \( \delta \) if \( w \in \Gamma_>(\pi)(C) \). Summing up the changes over all components we obtain (3.19).

Now consider a terminal \( t \in T \). Again, as the local order is preserved by \( \pi' \), the length of the root-\( t \)-path is only influenced by components \( C \) with \( t \in R(C) \). Consider such a component \( C \). If we move \( C \) by \( |\delta_C| \) towards the predecessor of \( C \), the length of the path is reduced by \( 2|\delta_C| \). On the other hand, if we move \( C \) in the other direction by \( |\delta_C| \), then the length is increased by \( 2|\delta_C| \). In total, the length changes by \( \text{sign}(C)2 \cdot |\delta_C| \). Summing up over all such components, we obtain (3.20).

The following observation is crucial in order to prove that there exist optimal solutions that are half-integral.

**Lemma 3.3.** If \( \Delta \) is a set of maximal \( x \)-components that do not contain terminals, then \( \{R(C)\}_{C \in \Delta} \) is a laminar family.

**Proof.** Let \( C_1, C_2 \in \Delta \). By definition \( V(C_1) \cap V(C_2) = \emptyset \). For \( i \in \{1, 2\} \) let \( v_i \) be the vertex of \( V(C'_i) \) that is adjacent to the predecessor of \( C_i \). Note that \( v_i \) is on the unique \( r \)-\( t \)-path for every \( t \in R(C_i) \) (see Figure 3.3). Now
assume that neither \( v_1 \) is on the \( r-v_2 \)-path nor \( v_2 \) is on the \( r-v_1 \)-path. Then 
\( R(C_1) \cap R(C_2) = \emptyset \). Otherwise, assume w.l.o.g. that \( v_1 \) is on the \( r-v_2 \)-path. In this case there exists a unique vertex \( v \) on the \( v_1-v_2 \)-path satisfying 
\( v \in \Gamma^>_\pi(C_1) \cup \Gamma^<_\pi(C_1) \). Now note that for all \( t \in R(C_2) \) the length of the 
\( r-t \)-path changes when moving \( C_1 \) if and only if the length of the \( r-v \)-path changes when moving \( C_1 \). Hence, 
\( R(C_2) \subseteq R(C_1) \) or 
\( R(C_2) \cap R(C_1) = \emptyset \). This implies the desired result.

Before we continue with the main result we make another simple observation:

**Proposition 3.4.** If \( \pi \) is an embedding and there exists a vertex \( t \in T \) such that 
\( d_\pi(t) > ||p(t) - p(r)||_1 \), then there exists a component \( C \) such that moving 
\( C \) towards its predecessor decreases the length of the root-\( t \)-path.

\[ \square \]

### 3.4 Half-Integrality of Solutions

In this section we prove that if all terminals are on integral coordinates and 
all length restrictions are integral, then there exists an optimal half-integral 
embedding. More precisely we prove that for any given feasible embedding \( \pi \) 
there exists a feasible half-integral embedding \( \sigma \) of at most the same cost such 
that the \( \ell_\infty \) distance between the positions of a vertex in both embeddings is 
at most 0.5. To this end we consider a sub problem that can be formulated 
as a linear program based on a totally unimodular matrix.

We start with some observations on half-integral embeddings.

**Proposition 3.5.** Every half-integral embedding has half-integral cost.

**Proof.** Obviously, all edges in such an embedding have half-integral lengths 
and thus the total length is also half-integral. \[ \square \]
Proposition 3.6. In every half-integral embedding $\pi$ the length of every root-terminal path has integral length.

Proof. Let $t \in T$ and denote by $P$ the unique root-$t$-path in $S$. If $P$ is a shortest path, then the length of $P$ is $||\pi(r) - \pi(t)||_1$, which is integral. If $P$ is not a shortest path, then by Proposition 3.4 there exists a component $C$ such that moving $C$ towards its predecessor decreases the length of $P$. As $\pi$ is half-integral, we can move $C$ by 0.5 towards its predecessor, reducing the length of $P$ by 1 and obtaining a new half-integral embedding $\pi'$. Then by induction the length of $P$ must be integral. $\square$

The main theorem of this section is the following.

Theorem 3.7. If $\pi$ is an embedding for an instance $(S,T,r,p,l)$ where all positions and length restrictions are integral, then there exists an half-integral embedding $\sigma$ with $\max_{v \in V} |\pi(v) - \sigma(v)|_\infty \leq 0.5$ and $c(\sigma) \leq c(\pi)$.

Proof. For $x \in \mathbb{R}$ we denote by $I(x)$ the smallest interval in $\mathbb{R}$ with half-integral boundaries such that $x$ is in the interior of the interval, that is,

$$I(x) := [\left\lfloor \frac{2x - 1}{2} \right\rfloor / 2, \left\lfloor \frac{2x + 1}{2} \right\rfloor / 2].$$

For a point $(x,y) \in \mathbb{R}^2$ we set $I((x,y)) := I(x) \times I(y)$. We show that there exists an half-integral embedding $\sigma$ with

$$\sigma(v) \in I(\pi(v)) \text{ for all } v \in V$$

such that $c(\sigma) \leq c(\pi)$.

Let $\sigma$ be a feasible embedding for $(S,T,r,p,l)$ of minimum cost satisfying (3.22). If there are several such embeddings we choose one with a minimal number of components that are not on half-integral coordinates. We denote this number by $N(\sigma)$ and prove that $N(\sigma) = 0$. Suppose that this is not the case. The idea is to move maximal components such that $N(\sigma)$ gets smaller without increasing $c(\sigma)$. As $\pi$ trivially satisfies (3.22), we have $c(\sigma) \leq c(\pi)$.

Let $\Delta_x$ and $\Delta_y$ be the sets of maximal $x$- and $y$-components, respectively, with respect to $\sigma$ that are not on half-integral coordinates and set $\Delta := \Delta_x \cup \Delta_y$. Then $N(\sigma) = |\Delta|$. For $C \in \Delta$ we set

$$z_C^* := \begin{cases} x(C) - \left\lfloor \frac{2x(C)}{2} \right\rfloor / 2 & C \in \Delta_x, \\ y(C) - \left\lfloor \frac{2y(C)}{2} \right\rfloor / 2 & C \in \Delta_y. \end{cases}$$

Consider a vector $z \in [0,0.5]^\Delta$. Starting with the embedding $\sigma$ and moving each component $C \in \Delta$ by $z_C - z_C^*$ in $x$- or $y$-direction, whether depending $i C \in \Delta_x$ or $C \in \Delta_y$, respectively, we obtain a new embedding
τ(0). Note that by the definition of $z_C^*$ this embedding is half-integral if and only if $z \in \{0, 0.5\}^\Delta$. Observe that $\tau(0)$ is half-integral, but it does not necessarily satisfy the length restrictions. Since by construction $\tau(0)$ preserves the local order of $\sigma$ we can apply Lemma 3.2 and conclude that for all $t \in T$ the length of the root-$t$-path with respect to $\tau(0)$ is

$$d_{\tau(0)}(t) = d_\sigma(t) + \sum_{C \in \Delta : t \in R(C)} 2\text{sign}(C) \cdot (-z_C^*).$$  (3.24)

As $\tau(0)$ is integral, this length is also integral by Proposition 3.6.

Using $z$ as a variable we can formulate a linear program reflecting the new cost of the embedding $\tau(z)$ and the length restrictions, under the assumption that $\tau(z)$ preserves the local order of $\sigma$:

$$\begin{align*}
\min &\ c(\sigma) + \sum_{C \in \Delta} (z_C - z_C^*) \cdot (|\Gamma^\sigma_<(C)| - |\Gamma^\sigma_>(C)|), \\
\text{s.t.} &\ d_\sigma(t) + \sum_{C \in \Delta : t \in R(C)} 2\text{sign}(C)(z_C - z_C^*) \leq l(t) \quad \forall t \in T \quad (3.25) \\
&\ 0 \leq 2z_C \leq 1 \quad \forall C \in \Delta. \quad (3.26)
\end{align*}$$

As $z = z^*$ is a feasible solution the linear program has an optimal solution (see also Figure 3.4). Substituting $2z_C$ by $z_C'$ for all $C \in \Delta$ and using (3.24) we obtain the modified linear program $(P')$:

$$\begin{align*}
\min &\ \sum_{C \in \Delta} z_C'/2 \cdot (|\Gamma^\sigma_<(C)| - |\Gamma^\sigma_>(C)|), \\
\text{s.t.} &\ \sum_{C \in \Delta : t \in R(C)} \text{sign}(C) z_C' \leq l(t) - d_{\tau(0)}(t) \quad \forall t \in T \quad (3.27) \\
&\ 0 \leq z_C' \leq 1 \quad \forall C \in \Delta. \quad (3.28)
\end{align*}$$

Figure 3.4: Detail of an embedding $\sigma$ with three maximal components not on half-integral positions. The embeddings $\tau_0$ and $\tau'$ preserve the local order of $\sigma$. 
We show that the matrix $A$ defined by the left side of the inequalities (3.27) is totally unimodular. Note that all entries of a column of $A$ are either non-negative or non-positive. Thus multiplying all rows with non-positive entries by $-1$ we obtain a non-negative matrix where each column corresponds to the characteristic vectors of $\{R(C)\}_{C \in \Delta} = \{R(C)\}_{C \in \Delta_x} \cup \{R(C)\}_{C \in \Delta_y}$. 

Recall, that by Lemma 3.3 the sets $\{R(C)\}_{C \in \Delta} = \{R(C)\}_{C \in \Delta_x} \cup \{R(C)\}_{C \in \Delta_y}$ are laminar families. We conclude that the rows of $A$ correspond to the characteristic vectors of the union of two laminar families. Edmonds [14] proved, that such matrices are totally unimodular.

Consequently, as the right hand side of (3.27) is integral, the constraints in (3.28) are integral and $A$ is totally unimodular, there exists an optimal solution for $(P')$ that is integral which further implies that the original LP has an half-integral optimal solution $\hat{z}$. But then $\tau(\hat{z})$ is also half integral and satisfies (3.22) and $c(\tau(\hat{z})) \leq c(\sigma)$.

If $\tau(\hat{z})$ preserves the local order of $\sigma$, then $\tau(\hat{z})$ is the embedding we are looking for and we are done. Otherwise choose $\lambda > 0$ minimal such that $\tau^\lambda$ defined by $\tau^\lambda(v) = \lambda \sigma(v) + (1 - \lambda)\tau(\hat{z})(v)$ for all $v \in V$ preserves the local order of $\sigma$. Note that $\lambda$ is well defined as the set of all embeddings preserving the local order of $\sigma$ is closed. As the cost and length functions are convex, $\tau^\lambda$ is a feasible embedding and $c(\tau^\lambda) \leq \lambda c(\sigma) + (1 - \lambda) c(\tau(\hat{z})) \leq c(\sigma)$. Moreover, every maximal $x$- or $y$-component of $\sigma$ is also an $x$- or $y$-component of $\tau^\lambda$, respectively, implying $N(\tau^\lambda) \leq N(\sigma)$. As $\tau^{\lambda-\delta}$ is not preserving the local order of $\sigma$ for all $\delta < 0$ but for $\delta \geq 0$ there must be an edge $(v, w)$ with $(\sigma_x(v) < \sigma_x(w)$ and $\tau_x^\lambda(v) = \tau_x^\lambda(w)$) or $(\sigma_y(v) < \sigma_y(w)$ and $\tau_y^\lambda(v) = \tau_y^\lambda(w)$). As the only components that are moved are not on half-integral position with respect to $\sigma$, we must have $N(\tau^\lambda) < N(\sigma)$ contradicting the choice of $\sigma$. This finishes the proof.

**Conclusion 3.8.** If all positions and length restrictions are integral, then there exists an optimal embedding that is half-integral.

### 3.5 Dynamic Programming

A consequence of the previous section is, that any non-optimal half-integral embedding can be improved by small half-integral movements of the Steiner points.

**Lemma 3.9.** If $\pi$ is an half-integral embedding that is not optimal, then there exists an half-integral embedding $\pi'$ with $\pi(v) - \pi'(v) \in \{-0.5, 0, 0.5\}$ for all $v \in V(S)$ and $c(\pi') \leq c(\pi) - 0.5$. 

Proof. Let \( \sigma \) be an optimal half-integral embedding. For \( \lambda \in (0,1) \) we define \( \pi_\lambda \) by \( \pi_\lambda(v) = \lambda \pi(v) + (1-\lambda)\sigma(v) \) for all \( v \in V(S) \). As \( \pi \) is not optimal and by the convexity of the length function, \( \pi_\lambda \) is a feasible embedding and we have \( c(\pi_\lambda) \leq \lambda c(\pi) + (1-\lambda)c(\sigma) \leq c(\pi) \). Choose \( \lambda \) small enough such that \( \max_{v \in V(S)} ||\pi(v) - \pi_\lambda(v)||_\infty < 0.5 \). Now Theorem 3.7 yields an half-integral embedding \( \pi' \) satisfying \( \max_{v \in V} ||\pi(v) - \pi'(v)||_\infty \leq \max_{v \in V} ||\pi(v) - \pi_\lambda(v)||_\infty + ||\pi_\lambda(v) - \pi'(v)||_\infty < 1 \) and \( c(\pi') \leq c(\pi_\lambda) < c(\pi) \). The claim follows by observing that \( \pi' \) and \( \pi \) are half-integral.

This lemma gives a direct idea for an algorithm based on dynamic programming to improve a non-optimal half-integral embedding. In the following, we interpret \( S \) as an arborescence rooted at \( r \) and denote by \( \Gamma^+(v) \) the children of a vertex \( v \in V(S) \). For simplicity of notation we set \( \pi_\delta(v) := \pi(v) + \delta \) for \( \delta \in \{-0.5,0,0.5\} \). Moreover, we expand the definition of length restrictions to Steiner points: Initially we set \( l^S(t) := l(t) \) for all \( t \in T \). For each vertex \( v \in V(S) \) whose children have a length restriction, we recursively set

\[
l^S(v) = \min_{w \in \Gamma^+(v)} l^S(w) - ||\pi(v) - \pi(w)||_1.
\]

Given a half-integral embedding \( \pi \) we want to compute an half-integral embedding \( \pi' \) with \( \pi(v) - \pi'(v) \in \{-0.5,0,0.5\}^2 \) and \( c(\pi') \) minimal. Note, that in this case the length of every root-terminal-path changes by at most \( 2n \).

As, additionally, \( \pi' \) is half-integral, \( l^S(v) \) is half-integral and \( ||l^S(v) - l^S(v')|| \leq 2n \) for all \( v \in V(S) \).

Thus it is sufficient to compute for every vertex \( v \in V(S) \), every translation \( \delta \in \{-0.5,0,0.5\} \) and every possible length restriction \( l \in \{l^S(v) - 2n,l^S(v) - 2n + 0.5,\ldots,l^S(v) - 2n + 2n - 0.5,l^S(v) + 2n\} \) the minimum length \( \gamma(v,\delta,l) \) of an embedding of the arborescence rooted at \( v \) such that \( v \) is positioned at \( \pi_\delta(v) \) and \( v \) satisfies the length restriction \( l \). For a terminal \( t \) we have \( \gamma(t,\delta,l) = 0 \) if \( \delta = (0,0) \) and \( l \leq l(t) \). Otherwise, we set \( \gamma(t,\delta,l) = \infty \). For all other vertices \( v \in V(T) \) we obviously have

\[
\sum_{\delta' \in \{-0.5,0,0.5\}^2} \min_{\delta' \in \{-0.5,0,0.5\}} \gamma(w,\delta',l - ||\pi_\delta(v) - \pi_\delta'(w)||_1 + ||\pi_\delta(v) - \pi_\delta'(w)||_1.
\]

It follows, that the length of an optimal embedding \( \pi' \) with \( \pi(v) - \pi'(v) \in \{-0.5,0,0.5\} \) is \( \gamma(r,(0,0),0) \). This number can be computed in \( O(n^2) \) time: There are \( O(n^2) \) different triples \( (v,\delta,l) \) for which \( \gamma(v,\delta,l) \) has to be computed and each of these computations can be done in constant time.

To compute a global optimal solution, we start with the trivial embedding, where all Steiner points are positioned at the root. This solution has cost \( C = \sum_{t \in T} ||p(t)||_1 \). Then we apply the dynamic programming approach as
long as the cost of the newly computed embedding decreases. As the cost is reduced by at least 0.5 in every round, we must obtain an optimal embedding after at most $2C$ iterations. Thus our algorithm has a pseudo polynomial running time of $O(Cn^2)$. In the next section we show how to refine this approach in order to achieve a polynomial running time.

### 3.6 An Optimal Polynomial Time Algorithm

We refine the ideas of the previous sections in order to obtain a polynomial time algorithm for our problem. In the first algorithm the Steiner points are moved by at most 0.5 in each direction in every call of the dynamic programming. The idea of the refined algorithm is to move the Steiner points by $2^k$ for a suitable $k \in \mathbb{Z}$ in the first rounds. As soon as no improvements can be obtained by moving Steiner points by $2^k$, we reduce the moving distance to $2^{k-1}$ and continue applying the dynamic programming (see Algorithm 2). Repeating this procedure we finally move the Steiner points by 0.5, obtaining an optimal embedding.

**Input**: An integral instance $(S, T, r, p, l)$ with $p(r) = (0, 0)$.

**Output**: An optimal embedding $\pi : V(G) \setminus T \to \mathbb{R}^2$.

```
1 Set $m \leftarrow \min \{m' \in \mathbb{N} : |p_x(v)| \leq 2^{m'} \text{ and } |p_y(v)| \leq 2^{m'} \forall v \in V(S)\}$;
2 Set $\pi(s) \leftarrow (0, 0)$ for all Steiner points $s \in V(S) \setminus T$;
3 $k \leftarrow m$;
4 while $k \geq -1$ do
5     Improve embedding $\pi$ by applying the dynamic programming with step width $2^k$ until no further length reduction is obtained;
6     $k \leftarrow k - 1$;
7 end
8 return $\pi$;
```

**Algorithm 2**: Optimal polynomial time algorithm

First we state a trivial lemma on the existence of feasible embeddings.

**Lemma 3.10.** There exists a feasible embedding (and thus an optimal one) for $(S, T, r, p, l)$ if and only if $||p(t)||_1 \leq l(t)$ for all $t \in T$.

**Proof.** If there exists a feasible embedding, then obviously $||p(t)||_1 \leq l(t)$ for all $t \in T$. If on the other hand $||p(t)||_1 \leq l(t)$, then placing all internal vertices on the position of the root is a feasible embedding satisfying the length restrictions. $\square$
For each \( k \in \mathbb{N} \) we define a new instance \( I_k := (S, T, r, p_k, l_k) \) on the same set of terminals and the same topology, but with new positions
\[
p_k(t) = (2^k \lfloor p_x(t)/2^k \rfloor, 2^k \lfloor p_y(t)/2^k \rfloor)
\]
for all \( t \in T \) and length restrictions
\[
l_k(t) = 2^k \lfloor l(t) - ||p(t) - p_k(t)||_1 / 2^k \rfloor
\]
for \( t \in T \). In other words we move each terminal towards the root onto the next \( 2^k \)-integral position and round each length restriction to the next lower multiple of \( 2^k \).

If there exists a feasible embedding for \((S, T, r, p, l)\), then there also exists a feasible embedding for \((S, T, r, p_k, l_k)\): To show this it is sufficient to prove that \( ||p_k(t)||_1 \leq l_k(t) \) for all \( t \in T \) by Lemma 3.10. By the choice of \( p_k \) and \( l_k \) we have
\[
||p_k(t)||_1 = ||p(t)||_1 - ||p(t) - p_k(t)||_1 \leq l(t) - ||p(t) - p_k(t)||_1 \leq l_k(t). \tag{3.29}
\]

Set \( m := \min\{m' \in \mathbb{N} : |p_x(t)| < 2^{m'} \text{ and } |p_y(t)| < 2^{m'} \forall t \in T\} \). Thus \( m \) is the smallest \( m \in \mathbb{N} \) such that \( p_m(t) = (0, 0) \) for all \( t \in T \).

**Remark 3.11.** The number \( m \) is polynomially bounded in the size of the instance.

In \((S, T, r, p_m, l_m)\) all terminals are placed at the position of the root. Thus placing all internal vertices to that position yields a trivial optimal solution of length 0. Now we compute by induction an optimal embedding for \((S, T, r, p_{k-1}, l_{k-1})\) given an optimal embedding for \((S, T, r, p_k, l_k)\). As \( m \) is polynomially bounded in the size of the input, each iteration can be computed in polynomial time and \((S, T, r, p, l) = (S, T, r, p_0, l_0)\), we obtain an optimal solution in polynomial time.

**Lemma 3.12.** Denote by \( \sigma_k \) an optimal solution for \( I_k \) for all \( k \in \mathbb{N} \). Then for \( k \in \mathbb{N} \) we have \( c(\sigma_{k+1}) \leq c(\sigma_k) + 6n2^k \).

**Proof.** Starting with \( \sigma_k \) we construct a feasible embedding for \( I_{k+1} \). By Lemma 3.7 we can assume w.l.o.g. that all internal vertices of \( S \) are on \( 2^{k-1} \)-integral positions in \( \sigma_k \). We define \( \pi \) by setting \( \pi(t) = p_{k+1}(t) \) for \( t \in T \) and \( \pi(v) = \sigma_k(v) \) for \( v \in V(S) \setminus T \). By this setting we have
\[
||\pi(t) - \sigma_k(t)||_1 = ||p_{k+1}(t) - p_k(t)||_1 \leq 2^{k+1}
\]
for all \( t \in T \). Thus
\[
c(\pi) \leq c(\sigma_k) + n2^{k+1} \tag{3.30}
\]
and the length of each root-terminal-path is increased by at most $2^{k+1}$. As $l_k(t) \leq l_{k+1}(t) + 2^{k+1}$ for all $t \in T$, we conclude that for each $t \in T$ the length restriction $l_{k+1}(t)$ is hurt in $\pi$ by at most $2^{k+2}$:

$$
\sum_{e \in E[r,t]} \pi(e) \leq \sum_{e \in E[r,t]} \sigma_k(e) + 2^{k+1} \leq l_k(t) + 2^{k+1} \leq l_{k+1}(t) + 2^{k+2}. \quad (3.31)
$$

Now we move components towards their predecessors, until all length restrictions are satisfied. To this end, denote by $\Delta$ the set of all maximal components, that do not contain the root $r$. Moving all components $C \in \Delta$ by $2^{k-1}$ towards their predecessors we obtain a new feasible embedding $\pi'$ with $c(\pi') \leq c(\pi) + n2^k$. Moreover, the length of every root-terminal-path that has not been a shortest one with respect to $\pi$ is reduced by at least $2^k$. Repeating this process with the new embedding at most 3 times yields a feasible embedding $\pi^*$ for $I_{k+1}$. We conclude $c(\pi^*) \leq c(\pi) + 4n2^k$. Together with (3.30) we get $c(\pi^*) \leq c(\sigma_k) + 6n2^k$. We finish the proof by observing that, as $\pi^*$ is feasible for $I_{k+1}$, the embedding $\sigma_{k+1}$ cannot be longer.

Combining the observations of the previous lemmas we obtain the main result of this chapter.

**Theorem 3.13.** The rectilinear Steiner tree embedding problem with length restrictions can be solved in polynomial time by a combinatorial algorithm.

*Proof.* Let $I = (S, T, r, p, l)$ be an instance of the problem. First we calculate $m$ as above. $m$ is polynomially bounded in the size of the input. Now we have a polynomial number of instances $I_k$, $k \in \{1, \ldots, m\}$. For $I_m$ we have the trivial embedding $\sigma_m$ with $\pi_m(v) = 0$ for all $v \in V$.

Let $\sigma_{k+1}$ be an optimal embedding for $I_{k+1}$. Then $\pi$ defined as $\pi(v) = p_k(v)$ for $v \in T$ and $\pi(v) = \sigma_{k+1}(v)$ otherwise is a feasible embedding for $I_k$. By Lemma 3.12, $c(\pi) \leq c(\sigma_{k+1}) + n2^k \leq O_k + 7n2^k$ where $O_k$ denotes the optimal length of an embedding for $I_k$. Moreover, $I_k$ is a $2^k$ integral instance. Thus applying the dynamic programming from the previous section at most $14n$ times we obtain an optimal solution $\sigma_k$ for $I_k$. We conclude that computing $\sigma_k$ from $\sigma_{k+1}$ requires at most time $O(n^3)$. By induction we get an optimal solution $\sigma$ for $I_1 = (S, T, r, c, l)$. The total running time is $O(mn^3)$ where $m = \lceil \max\{|p_c(t)|, |p_s(t)| : t \in T\} \rceil + 1$.

Obviously, every feasible solution for $I_k$ corresponds to a feasible solution for $I$. Moreover, all Steiner points of such embeddings are on $2^{k-1}$-integral positions. Due to this observation, the implementation of the algorithm can be modified in order to decrease the number of dynamic programming steps in practice. Instead of computing an optimal solution for $I_k$, we are looking for an embedding of minimal cost for the original instance $I$, but all
Steiner points have to be on $2^{k-1}$-integral coordinates. We use the dynamic programming steps as described in order to improve a given embedding, but now the cost of each solution is computed using the original positions of the terminals and considering the original length restrictions. Using this method, the number of dynamic programming steps performed in the algorithm gets very small. It turns out, that in practice the number of dynamic programming calls is constant for each $k$ in the most cases.

![Figure 3.5: Instance for the Steiner tree embedding problem (i) and an optimal embedding if there are no length restrictions (ii)](image)

Figures 3.5 and 3.6 show how the algorithm works on an example. Figure 3.5 (i) shows the instance and Figure 3.5 (ii) an optimal embedding of length 35 if there are no length restrictions. In Figure 3.6 the embeddings computed by our algorithm are shown. As input we used the instance from Figure 3.5 with length restrictions $l_a = 10$, $l_b = 11$ and $l_c = 20$. As $\max\{|p_x(t)|, |p_y(t)| : t \in T\} = 10$ we have $m = 4$. Thus the algorithm begins with an embedding where all Steiner points are $2^{m-1}$-integral (Figure 3.6 (i)). The last one is the final optimal embedding of length 37.5. For each $k$ the dynamic programming is called at most twice, the first time the length is reduced, the second time an embedding of the same cost is computed, proving that it is an optimal one.
Figure 3.6: Run of the algorithm on the instance shown in Figure 3.5 (i) with length restrictions $l_a = 10$, $l_b = 11$ and $l_c = 20$. Figure (vi) shows the final optimal solution.
Chapter 4

Steiner Arborescences with Depth Restrictions

Recall that the Steiner trees we studied in the previous chapter were originally motivated by building repeater trees with timing constraints. In Chapter 1 we introduced a delay model where the delay on a path from a root $r$ to a terminal $t$ within a tree $S$ is proportional to the length of the unique $r$-$t$-path in $S$ plus the number of Steiner points on the path. If we embed a given Steiner tree into the rectilinear plane, the number of Steiner points on root-terminal-paths does not change and thus can be neglected. We have seen, that in this case an optimal embedding can be computed efficiently in polynomial time.

Now the question arises which complexity the problem has, if we also need to compute the topology of the Steiner tree. If the timing constraints are very loose, we arrive at the task of computing a shortest rectilinear Steiner tree, proving that our problem is NP-hard. But what happens if the timing constraints are as tight as possible? In this case the timing constraints force us to compute a Steiner tree where all root-terminal-paths are shortest ones and the number of Steiner points on each of these paths is given in advance. In this case we obtain the rectilinear Steiner arborescence problem with depth restrictions. In this chapter we prove that even this problem with the tightest possible constraints is NP-hard.

4.1 Problem Formulation

Let $T = \{t_1, \ldots, t_n\}$ be a set of terminals with positions $p(t) \in \mathbb{R}^2$ in the plane for all $t \in T$, a distinguished terminal $r = t_1$, which we call the root, with $p(r) = (0, 0)$ and a function $d : T \setminus \{r\} \rightarrow \mathbb{N}$. A depth-restricted rectilinear
**Steiner arborescence** is an arborescence $A$ with root $r$, leaves $T \setminus \{r\}$ and an embedding $\pi : T \to \mathbb{R}^2$ in the plane such that

- each Steiner point has degree 3,
- each terminal $t \in T$ has degree 1,
- $\pi(t) = p(t)$ for all $t \in T$,
- the unique path $P$ in $A$ from $r$ to $t \in T$ is a shortest path with respect to rectilinear distances, that is,

\[
\sum_{(v,w) \in E(P)} ||\pi(v) - \pi(w)||_1 = ||p(t) - p(s)||_1
\]

and

- for each $t \in T$ the number of internal vertices on the unique path from $r$ to $t$ is $d(t)$.

The task is to compute such a tree of minimum rectilinear length. During this chapter the depth of a terminal $t$ is always the number of internal vertices on the unique $r$-$t$-path.

Without the depth restrictions, we get the problem of computing a shortest rectilinear Steiner arborescence which is NP-hard as shown by Shi and Su [58].

Note that vertices of a feasible tree might be placed on the same position. Moreover, in an optimal solution it is possible to have edges that cross or run parallel on top of each other, which is not possible in an optimal Steiner arborescence without depth-restrictions.

Figure 4.1 shows examples for minimum Steiner trees, Steiner arborescences and depth-restricted Steiner arborescences in the rectilinear plane.

## 4.2 Feasibility

It is easy to verify, whether there exists a feasible solution for a given instance $(T, p, d)$. To this end, let $(A, \pi)$ be a feasible solution, that is, $A$ is a binary tree where each terminal $t \in T$ is at depth $d(t)$.

By Kraft’s inequality [46], there exists a binary tree with leaves at depths exactly $d_1, \ldots, d_n$ (in any order) if and only if

\[
\sum_{i=1}^{n} 2^{-d_i} = 1. \quad (4.1)
\]
Figure 4.1: A shortest rectilinear Steiner tree (i), a shortest rectilinear Steiner arborescence (ii) and a shortest depth-restricted rectilinear Steiner arborescence (iii). The root of the instances is denoted by $r$ and the numbers in (iii) denote the given depths of the terminals.

Hence, for a given instance $(T, p, d)$ a feasible solution exists if and only if (4.1) is satisfied for $d_i = d(v_i)$, $i \in \{1, \ldots, n\}$. In this case, it is easy to construct a feasible - but not necessarily shortest - depth-restricted Steiner arborescence: Using Huffman coding [31] one can compute a binary tree satisfying (4.1). Placing all internal vertices of this tree at the position of the root results in a feasible tree. We conclude that deciding whether a feasible tree exists for a given instance can be done in polynomial time. However, we are interested in the complexity of computing a shortest depth restricted tree.

4.3 Main Idea

In the remainder of this chapter we only consider instances where the root is placed at the origin and all terminals are placed in the first quadrant. Note that this is a further restriction of the problem. It will simplify our analysis significantly. Thus in any feasible solution the parent $w$ of a vertex $v$ always satisfies $p_x(w) \leq p_x(v)$ and $p_y(w) \leq p_y(v)$.

Let $A$ be a feasible arborescence for an instance $(T, p, d)$. Then obviously the depth of an internal vertex $v$ is one smaller than the depth of its two children $w_1$ and $w_2$, that is $d(v) = d(w_1) - 1 (= d(w_2) - 1)$. We extend the definition of depth to edges by setting the depth of an edge $(v, w)$ to be $d(w)$. Two vertices can have a common parent if and only if they have to be at the same depth.

If the arborescence $A$ is given, the optimal positions of the internal vertices can be easily computed:

**Proposition 4.1.** If an internal vertex $v$ has two children at positions $(x_1, y_1)$
and \((x_2, y_2)\), respectively, then the optimal position for \(v\) is \((\min\{x_1, x_2\}, \min\{y_1, y_2\})\).

Proof. In a feasible solution we have \(p_x(v) \leq \min\{x_1, x_2\}\) and \(p_y(v) \leq \min\{y_1, y_2\}\). If one of the inequalities is not satisfied with equality, moving the vertex to the right or above yields another feasible embedding that is shorter.

An intermediate consequence of Proposition 4.1 is that in every optimal solution all vertices are placed on the vertices of the Hanan grid on \(T\).

### 4.4 Reduction Overview

We prove the NP-completeness by a reduction from Maximum 2-Satisfiability (in short Max-2-Sat). A Max-2-Sat instance consists of a set of variables \(V = \{x_1, \ldots, x_n\}\) and a set of clauses \(C = \{C_1, \ldots, C_m\}\) on \(V\) with \(|C_i| = 2\) for \(i \in \{1, \ldots, m\}\). The problem is to find a truth assignment \(\pi\) such that the number of clauses satisfied by \(\pi\) is maximized. Garey et al. [19] proved that Max-2-Sat is NP-hard.

We use the component design technique to transform a Max-2-Sat instance \((V, C)\) into an instance for our problem. First we give a high-level overview of this reduction. The construction consists of several types of gadgets that are placed on a uniform grid, where the root is located at the origin. For each variable and each clause, we have a variable gadget and clause gadget, respectively, that are placed on the diagonal of the grid (see Figure 4.2). The gadgets are connected by horizontal or vertical connections representing the literals: For every variable \(x_i\), one connection is leaving the variable gadget to the left (corresponding to the literal \(x_i\)) and one connection is leaving to the bottom (corresponding to \(\overline{x_i}\)). Each clause gadget receives one connection from above and one from the right, representing the corresponding literals of the clause. In order to split a connection and to switch from horizontal to vertical connections or vice versa we add splitter gadgets. Finally, we require connections ensuring the existence of a feasible solution for the instance (marked by dashed lines in Figure 4.2).

Each truth assignment for \((V, C)\) corresponds to a feasible Steiner arborescence where a truth assignment satisfying a maximal number of clauses corresponds to a shortest feasible Steiner arborescence. The length of the connections within the variable and splitter gadgets and between them differ only slightly for different truth assignments, but the length of a connection of clause gadgets increases by a (relatively large) constant \(C\) if the clause is not satisfied by the truth assignment. Therefore, there exists a truth assignment \(\pi\) satisfying \(k\) of \(m\) clauses if and only if there exists a feasible Steiner
arborescence of length at most \( c + (m - k)C \) where \( c \) is a constant that is independent of \( \pi \).

Figure 4.2: High-level overview of the transformed instance. Root, variable, clause and splitter gadgets are marked by r,v,c and s, respectively. The bold lines show the connections between the gadgets and the dashed lines the connections required to enable a feasible solution.

4.5 The Tile Design

The gadgets are realized by equal sized quadratic tiles. In this section we describe the design of the different types of tiles. Each tile has size \((4\alpha + 2) \times (4\alpha + 2)\) and contains several terminals, depending on the tile’s type. The tiles are placed on a uniform grid containing \((1 + m + 2n) \times (1 + m + 2n)\) tiles and having lattice spacing \(4\alpha + 2\). The integral constant \(\alpha\) will be set later (see Section 4.6).

Figure 4.3 shows a prototype of a tile. The black squares show possible positions of terminals and the dotted lines show the Hanan grid on the set of possible terminal positions.

On some type of tiles we use terminal cascades, a set of terminals with consecutive given depths placed at the same position. If an edge of depth
\(a + k\) starts at the position \(p\) of a terminal cascade containing \(k\) terminals with depths from \(a + 1\) to \(a + k\), then all terminals of the cascade can be connected to the tree by adding \(k\) Steiner points at position \(p\) and connecting the terminals and Steiner points appropriately. In this case, an edge of depth \(a\) ends at the cascade and we have no additional connection cost as all inserted edges have length 0. If no edge of depth \(a + k\) starts at the position of the cascade, the instance is constructed such that we have \(k\) edges of length at least 1, increasing the cost to connect the cascade to the tree by at least \(k\).

A special type of terminal cascades are the double terminals consisting of two terminals at the same position with consecutive depths. Double terminals are only placed on positions \(D_j, j \in \{1, 2, 3, 4\}\) (see Figure 4.3). A tile \(t\) contains a terminal at position \(o_j\) (for \(j \in \{1, 2, 3, 4\}\)) if and only if there is a double terminal at position \(D_j\). Moreover, if the double terminal at \(D_j, j \in \{1, 2\}\), has depths \(k\) and \(k - 1\), then \(o_j\) has depth \(k - 2\) and if the double terminal at \(D_j, j \in \{3, 4\}\) has depths \(k\) and \(k - 1\), then \(o_j\) has depth \(k + 1\).

Let \((A, \pi)\) be a feasible solution, \(t\) a tile and \(\pi|_t\) be the embedding we obtain by projecting all vertices that are outside of \(t\) with respect to \(\pi\) to the nearest point on the border of \(t\). Then the length of \((A, \pi)\) on \(t\) is the total length with respect to \(\pi|_t\) of all edges \((v, w)\) that contain at least one vertex in the inner of \(t\). Note that the total length of \((A, \pi)\) on all tiles is a lower bound for the length of \((A, \pi)\).

**Proposition 4.2.** If \(t\) is a tile with \(k\) double terminals, then every feasible solution has length at least \(2k\alpha\) on \(t\).

**Proof.** Recall that double terminals are only located at positions \(D_j, j \in \{1, 2, 3, 4\}\). Let \(H\) be the Hanan grid on all terminals and consider a feasible arborescence \(A\) so that all vertices are on positions of vertices of \(H\). Let \(D\) be a double terminal at position \(p\) and \(P\) be the set of vertices of \(A\) having distance at most 1 to \(p\). By construction, for every edge \(\{v, w\}\) with \(v \in P\) we have either \(w \in P\) or \(||p(v) - p(w)|| \geq \alpha\).

In a feasible arborescence at least one edge must leave \(P\). If no edge is entering \(P\), then it only contains the two terminals at positions \(p\). As they have different depths, the cannot have the same parent and thus two edges must leave \(P\). Thus \(|\delta(P)| \geq 2\) implies that connecting a double terminal increases the length of a feasible connection by at least \(2\alpha\).

An intermediate consequence is the following:

**Conclusion 4.3.** Any feasible arborescence for an instance containing \(k\) double terminals has length at least \(2k\alpha\).

We construct the tiles and the instance \(I\) such that if \(I\) contains \(k\) double terminals, then there exists a solution \(A\) of length less than \((2k + 1)\alpha\).
Lemma 4.4. Let \((A, \pi)\) be an optimal solution of length strictly less than \((2k+1)\alpha\) for an instance \(I\) with \(k\) double terminals. If \(t\) is a tile with terminal \(s\) at position \(o_i\) for some \(i \in \{1, 2, 3, 4\}\), then the Steiner point connected to \(s\) is either placed at position \(o_i\) or position \(\hat{o}_i\).

Proof. Denote by \(s'\) the Steiner point in \(A\) connected to \(s\). As \(A\) is optimal, \(s'\) is placed on the Hanan grid given by the terminals. If \(s'\) is not placed at \(o_i\) or \(\hat{o}_i\), then the distance between \(s\) and \(s'\) is at least \(\alpha\). But then the total length of \(A\) is at least \((2k+1)\alpha\), contradicting the assumption.

During the remainder of the chapter we make the following assumption on the constructed instances. By setting \(\alpha\) to an appropriate value, we can later guarantee that the assumption is indeed satisfied.

Assumption 4.5. If \(I\) is an instance with \(k\) double terminals, then there exists an optimal solution of cost less than \((2k+1)\alpha\).

By Lemma 4.4 a tile \(t\) can only be entered or left at Steiner points that are connected to a terminal at position \(o_i, i \in \{1, 2, 3, 4\}\). We call these
Steiner points *input* or *output* of tile \( t \) if it is connected to a terminal at position \( o_i \) with \( i \in \{3, 4\} \) or \( i \in \{1, 2\} \), respectively. Note that each input of a tile \( t \) is the output of the tile that shares the border of \( t \) containing the input. Thus it suffices to only consider inputs in the following. For every input \( s \), we define the depth and the *parity* of \( s \): Consider a tile with an input connected to a terminal \( v \) at position \( o_i \). The depth of \( s \) is \( d(v) - 1 \) (recall that \( s \) is a child of \( v \)). If the input is placed at the position of \( v \), then the parity of the input is 0. Otherwise, the parity is 1. Later we associate with each input a literal. Then the literal is set to \texttt{true} if and only if the parities of the associated inputs are 1.

Now we can decompose an optimal solutions at the inputs and outputs of the tiles: Consider a tile \( t \) with inputs \( I \) and parities \( \pi : I \to \{0, 1\} \) for the inputs. A *tile branching* for \((t, \pi)\) is a branching \( B \) (that is a forest where each tree is an arborescence) containing an arborescence for each output and the leaves of \( B \) are the terminals of \( t \) plus one leaf for each input \( i \in I \) at the position corresponding to the parity \( \pi(i) \). Moreover, the branching satisfies the depth-restrictions, that is, \( d(w) = d(v) - 1 \) for the parent \( w \) of a vertex \( v \). Hereby we use the depths of the inputs as the depths for the terminals placed at their positions. An implication is that the depths of the roots coincide with the depths of the corresponding outputs.

A *shortest connection* for tile \( t \) with parities \( \pi \) for the inputs is a shortest tile branching for \((t, \pi)\). Shortest connections for a given tile \( t \) with parities \( \pi \) can be computed efficiently by dynamic programming: We know that the Steiner points are only placed at vertices of the Hanan grid and that both children of a Steiner point must have the same depth.

### Variable Tiles

For every variable of a 3-SAT instance \((V, C)\) we build a *variable tile* containing 8 terminals and two outputs. Figure 4.5 shows the positions of the terminals as black squares and their depths. The dotted lines show possible positions of edges in an optimal solution.
Figure 4.5: A variable tile

Figure 4.6: The two possible shortest connections for a variable tile
Lemma 4.6. There are two shortest connections for a variable tile. The total edge length of a minimum connection is $4\alpha + 5$.

Proof. The two minimum connections are shown in Figure 4.6. Note that by the placement of the terminal at position $(2\alpha + 1, 2\alpha + 1)$ the parity of at least one of the outputs must be 0.

We associate with the output at position $o_1/\hat{o}_1$ the literal $x_i$ and with the output at position $o_2/\hat{o}_2$ literal $\overline{x}_i$. Thereby, the connection in Figure 4.6 (left) corresponds to $x_i = \text{true}$, while the connection in Figure 4.6 (right) corresponds to $x_i = \text{false}$.

Clause Tiles

For every clause, we construct a clause tile as shown in Figure 4.7. It contains two terminal cascades (drawn as black rhombs), each with $\beta$ terminals and both inputs have depth $k$. The integral value $\beta$ is constant and the same for all clause tiles and will be set later. We denote the upper right terminal cascade by $S_1$.

The length of a minimum connection of a clause tile depends on the parities of the inputs.

Lemma 4.7. A minimum connection for a clause tile has length $6\alpha + 9$ if both inputs have parity 1, length $6\alpha + 10$ if exactly one input has parity 1 and length $6\alpha + 11 + \beta$ if both inputs have parity 0.

Proof. Figure 4.8 shows minimum connections for the four different pairs of parities of the inputs. Note that in the case where both inputs have parity 0 there are $k - l$ edges of length 1 leaving the terminal cascade at position $(2\alpha + 1, 2\alpha + 1)$. The terminal cascade at position $(2\alpha, 2\alpha)$ enforces the parity of the output to be 0.

Connection Tiles

In order to guarantee, that in an optimal solution the proper inputs and outputs are connected, we add connection tiles. A horizontal or vertical connection tile enables the connection of a horizontal or vertical input with an horizontal or vertical output, respectively (see Figure 4.9, left).

Lemma 4.8. A minimum connection for a connection tile has length $4\alpha + 2$ if the parity of the input is 1 and $4\alpha + 8$ otherwise.
Proof. The minimum connections are shown in Figure 4.10. Note that in a minimum connection the parities of the input and the output of a connection tile are the same.

Furthermore, we use crossing tiles, each consisting of the union of a vertical and an horizontal connection tile (see Figure 4.9, right). The vertical and horizontal connection of a crossing do not influence each other, if the depths of the corresponding terminals are distinct.

Splitter Tiles

Note that a connection tile can only connect inputs and outputs that are both at horizontal or both on vertical borders of their tiles and the tiles that are connected have to be in the same row or column of the underlying grid. But there exist cases where we have to connect one input with two outputs or where the input is on a horizontal border and the output is on a vertical borer or vise versa. In these cases we “split” the path connecting inputs and outputs.

To this end, we introduce splitter tiles (see Figure 4.11). A splitter tile contains one input with depth $k$ and two outputs. It is designed in such a way that in an optimal solution the parities of all inputs and outputs are the same. There are two types of splitter tiles; one, where the input is at the upper border and one where the input is on the right border. As these
Figure 4.8: Shortest connections for a clause tile

types of tiles are the same up to symmetry, we restrict ourselves to define
and analyze splitter tiles where the input is on the right border of the tile.
The terminal cascade plays a crucial role here.

**Proposition 4.9.** Each feasible connection of a splitter tile with a terminal
cascade containing $\gamma$ terminals has length at least $6\alpha + \gamma$.

**Proof.** Consider an horizontal splitter tile $t$ and denote by $C$ the terminals of
the terminal cascade. As $t$ contains one input and two outputs, the induced
length is at least $6\alpha$ by Prop. 4.2. Let $S$ be the set of Steiner points that
are placed on the position of the terminal cascade. If $S$ is empty, then the
distance between each terminal of $C$ and its parent is at least 1, thus the
total length of the connection is at least $6\alpha + |C| = 6\alpha + \gamma$.

If on the other hand $S \neq \emptyset$, then let $s \in S$ be a vertex with highest
depths. Consider the subtree rooted at $s$. As all terminals of $C$ have distinct
depths and by the observation that the double terminals of the input cannot
be in the subtree, there must be a vertex in the subtree outside the tile. But
then the induced length of $t$ is at least $8\alpha > 4\alpha + \gamma$, where $6\alpha$ comes from
the double terminals and $2\alpha$ from the subtree rooted at $s$. □
Now we analyze the length of shortest connections for a splitter tile:

**Lemma 4.10.** Let $A$ be a minimum connection for a splitter with a terminal cascade containing $\gamma$ terminals and set $L = 6\alpha + \gamma + 3$. If the parities of the inputs and outputs are 1, then $A$ has length $L$. If the parities of the inputs and outputs are 0, then $A$ has length $L + 8$. Finally, if the parity of the input is 0 and the parities of the outputs are 1 then $A$ has length $L + 1 + 2\gamma$.

**Proof.** Figure 4.12 shows the three possible shortest connections. In the first two cases each terminal of the terminal cascade has to be connected to the tree by an edge of length 1, but in the last case, they are connected by edges of length 2.

In the third case of Lemma 4.10 the parity switches from 0 at the input to 1 at the outputs, that is, the corresponding literal switches from `false`
to true. We call this type of connection a forbidden connection. As this is not allowed we have to ensure that such connections are never a part of an optimal solution. So we assume that the following assumption is satisfied during the remainder of the chapter.

**Assumption 4.11.** If $I$ is an instance with $k$ double terminals and $l$ splitter tiles with terminal cascades containing $\gamma_1, \ldots, \gamma_l$ terminals, respectively, then an optimal solution has cost less than $2k\alpha + \sum_{i \in \{1, \ldots, l\}} \gamma_i + \min_{i \in \{1, \ldots, l\}} \gamma_i$.

If this assumption is satisfied, then the solution does not contain forbidden connections.

![Figure 4.11: Horizontal splitter tile: position of the terminals and the terminal cascade and their depths](image1)

![Figure 4.12: The three possible shortest connections for a splitter tile](image2)
Additional Connections

Using the variable, clause, splitter and connection tiles we are now able to build the main part of our instance. But there are still some “open” outputs that are not connected yet, for example the outputs of the clauses. Thus we have to add terminals with consecutive depths in order to connect these outputs to the root \( r \). For every additional input and output that we use we add a double terminal in order to coincide with the structure of the prototype tile. If two such connections meet, we add a terminal cascade such that these connections can merge into a single vertex. Finally, we add a terminal cascade at the root in order to guarantee the existence of a feasible solution. Note that the minimum total length of the edges required to connect these additional terminals is always the same.

4.6 NP-completeness

Let \( T \) be the set of all tiles. For each tile \( t \in T \) we denote by \( L(t) \) the minimum length of a shortest connection for \( t \). Then \( L := \sum_{t \in T} L(t) \) is a lower bound for the length of a feasible solution.

Let \( \pi : \{x_1, \ldots, x_n\} \to \{0, 1\} \) be a truth assignment. We construct a feasible solution by setting the parities of the variable tiles according to the truth assignment, inducing the parities of all inputs and outputs of the tiles and call the resulting arborescence a realization of \( \pi \).

Figure 4.13 shows an instance for the Max-2-Sat \((V, C)\) defined by \( V = \{x_1, x_2, x_3\} \), \( C = \{C_1, \ldots, C_5\} \), \( C_1 = \{x_1, x_2\} \), \( C_2 = \{x_1, \bar{x}_2\} \), \( C_3 = \{\bar{x}_1, x_2\} \), \( C_4 = \{x_1, x_3\} \) and \( C_5 = \{\bar{x}_2, \bar{x}_3\} \). Moreover, a shortest solution corresponding to the truth assignment \( x_1 = x_3 = \text{true} \) and \( x_2 = \text{false} \) is shown. The given depths have been omitted for clarity of presentation. The figure also illustrates, how the open outputs are connected to the root.

Next, we analyze the length of a realization. By Lemma 4.7 the induced connection of a clause tile has length \( 6\alpha + 9 \) or \( 6\alpha + 10 \) if the clause is satisfied under \( \pi \) and \( 6\alpha + 11 + \beta \) otherwise. For all other tiles, the length of the induced connection is at most 8 units longer than a minimum connection for that tile. Let \( u \) be the number of clauses that are not satisfied under \( \pi \). Then the length \( L(\pi) \) of the solution is at least \( L + 2u\beta \) and at most \( L + 2u\beta + 3N^2 < L + 3u\beta \).

For the length \( \ell \) of a realization satisfying \( n - u \) of the clauses, we have

\[
\ell \in [L + u\beta, L + u\beta + 10(nm)^2] =: B_u. \tag{4.2}
\]

Now the values for \( \alpha, \beta \) and \( \gamma \) can be specified. We have to set \( \beta \) such that the length of a shortest realization indicates, how many clauses are satisfied.
To this end, the sets $B_u, u \in \{1, \ldots, m\}$, have to be distinct and we set
\[
\beta = 20(nm)^2. \tag{4.3}
\]

In order to satisfy Assumption 4.11, we observe that every feasible realization has length at most
\[
2l\alpha + \sum_{i \in \{1, \ldots, l\}} \gamma_i + 10(nm)^2 + m\beta, \tag{4.4}
\]
where $\gamma_i$ is the number of terminals in the terminal cascade of the $i$'s splitter tile. Thus Assumption 4.11 is satisfied if
\[
4(nm)^3 > \gamma_i > (nm)^3 \tag{4.5}
\]
for all $i \in \{1, \ldots, l\}$. This can be achieved by setting the depths of the four terminals of each variable tile to $4(nm)^3$ and the depths of the connection and splitter tiles appropriately. This is possible as every root-terminal-path passes at most two splitter tiles.

Using (4.4) in (4.5), we conclude, that the length of a realization for an instance with $k$ double terminals is at most
\[
2k\alpha + 4l(nm)^3 + 10(nm)^2 + m20(nm)^2 \tag{4.6}
\]
which is at most $2k\alpha + (nm)^4$ if $nm$ is sufficiently large. By setting $\alpha := (nm)^4$ Assumption 4.5 is satisfied. Note that the values for $\alpha, \beta$ and $\gamma$ and the number of terminals are polynomially bounded in $n + m$. Thus we have a polynomial transformation.

All the previous observations together give us the main result of this chapter.

**Theorem 4.12.** The depth-restricted rectilinear Steiner arborescence problem is strongly NP-complete.

**Proof.** The problem is obviously in NP. Using the transformation described in this chapter we can transform a Max-2-Sat instance $(\mathcal{V}, \mathcal{C})$ into an instance $I$ of the depth-restricted rectilinear Steiner arborescence problem. As the number of terminals, the depths and the distances in $T$ are polynomially bounded in $|\mathcal{V}| + |\mathcal{C}|$, the transformation is a polynomial one. We conclude that as Max-2-Sat is strongly NP-complete, so is the depth-restricted rectilinear Steiner arborescence problem. \qed
Figure 4.13: An instance of the rectilinear Steiner arborescence with depth restriction problem and a shortest solution for it
Chapter 5

Binary Trees with Choosable Edge Lengths

In the previous two chapters we have considered problems concerning the construction and embedding of Steiner trees. In both cases the problems were motivated by the task to build repeater trees satisfying some timing constraints. Recall that we have modeled the delay of a signal on the path between the root and a terminal to be equivalent to the length of the path plus some constant times the number of Steiner points on the path. In the case of the rectilinear Steiner tree embedding problem we were able to ignore the additional delay of the Steiner points as we fixed a topology of the tree and thus the number of vertices on a path does not change. If we also have to construct the topology of the Steiner tree, the delay imposed by the vertices cannot be neglected anymore. In the previous chapter we have seen that in this case computing a shortest Steiner tree satisfying timing constraints is NP-hard, even if all root-terminal-paths have to be shortest ones. Note that in the model used in the previous chapter the delay imposed by the vertices was constant.

In general there is a certain degree of freedom to distribute the vertex delay to different branches of the tree. In this delay model we consider only binary Steiner trees where Steiner points can have the same position. By inserting a gate at a vertex of the tree it is possible to reduce the delay of one of the incident branches while increasing the delay on the other branch by about the same amount. As there are only a discrete number of gates with different sizes available, this effect can be modeled by so-called $L'(k)$-trees for some appropriate $k \in \mathbb{N}$ where an $L'(k)$-tree is a binary tree in which all edges have positive integral lengths and the sum of the lengths of the two edges leading from every non-leaf to its two children is $k$. Then the required arrival times at each sink correspond to a depth restriction for a leaf of the
tree and the delay on the edges correspond to the length of the edges. The task is to build an \( L'(k) \)-tree such that the signal reaches each sink in time.

In this chapter we consider the problem of deciding, whether for given arrival times/depths there exists a feasible tree (if we consider the problem as a decision problem) and to compute a tree with latest possible starting time (considering it as an optimization problem). By placing all Steiner points at the position of the root, we can assume that all root-terminal-paths are shortest ones. Thus the problem reduces to computing an optimal \( L'(k) \)-tree.

We give the first polynomial time algorithm that computes an optimal binary tree with choosable edge lengths if \( k \) is constant.

The results in this chapter have been published in [51].

5.1 Problem Formulation

For a fixed \( k \in \mathbb{N} \) we define \( L'(k) = \{\{i,k-i\} \mid 1 \leq i \leq k-1, i \in \mathbb{N}\} \). An \( L'(k) \)-tree is a rooted strict binary tree with the property that the two edges leading from every non-leaf to its two children are assigned lengths \( l_1 \) and \( l_2 \) with \( \{l_1,l_2\} \in L'(k) \). In this context we will from now on call a multi-set \( D = \{d_1,\ldots,d_n\} \) a leaf signature. In this chapter we show for fixed \( k \) how to decide in polynomial time for a given leaf signature \( \{d_1,\ldots,d_n\} \) if there exists an \( L'(k) \)-tree with \( n \) leaves at depths at most \( d_1,\ldots,d_n \) (in any order) and how to construct such a tree. See Figure 5.1 for an example of an \( L'(6) \)-tree for the leaf signature \( \{5,7,7,8,8,9\} \). As the numbers of a leaf signature give an upper bound on the depths of the leaves, the tree of the example in Figure 5.1 is also an \( L'(6) \)-tree for the leaf signatures \( \{7,7,7,9,9,9\} \) and \( \{9,9,9,9,9,9\} \).

If \( k = 2 \), then all edges of an \( L'(k) \)-tree have length 1. In this case we get classical binary trees where the depth of a leaf is equal to the number of edges of the unique path from the root to the leaf. By Kraft’s inequality [46] an \( L'(2) \)-tree for \( D \) exists if and only if

\[
\sum_{i \in \{1,\ldots,n\}} 2^{-d_i} \leq 1. \tag{5.1}
\]

Such a tree can be constructed using the Huffman Coding algorithm [31].

\( L'(k) \)-trees have first been considered by Maßberg and Rautenbach [52].

In [52] it has been shown how to construct \( L'(4) \)-trees in polynomial time. For \( k > 4 \) it was an open problem if there exists a polynomial algorithm that can decide the existence of \( L'(k) \)-trees for given instances. We show that we can decide the existence in time \( O(n^{k+3}) \), that is, in polynomial time for constant \( k \).
The problem is related to prefix-free codes with unequal letter costs (see e.g. [5, 20, 22, 23]). Nevertheless, there are significant differences between these problems. First, in our problem we ask for a tree satisfying given depth restrictions for the leaves while for prefix-free codes the task is to find a tree minimizing \( \sum_{i \in \{1, \ldots, n\}} \text{depth}(i)w_i \) where \( \{w_1, \ldots, w_n\} \) are given numbers and \( \text{depth}(i) \) denotes the depth of leaf \( i \), \( i \in \{1, \ldots, n\} \). Moreover, in our problem we have the freedom to choose the edge lengths from a discrete set of numbers as long as the sum of the lengths of the edges leaving a vertex equals \( k \).

### 5.2 Core Algorithm

Let \( D = \{d_1, \ldots, d_n\} \) be a leaf signature. We want to decide if there exists an \( \mathcal{L}'(k) \)-tree with \( n \) leaves at depths at most \( d_1, \ldots, d_n \). If such a tree exists we call the leaf signature \( D \) realizable.

In this section we present the core algorithm that can decide the existence of an \( \mathcal{L}'(k) \)-tree for a given leaf signature. The idea of the algorithm is the following: We can reduce a leaf signature of length \( n \) by combining two leaves of a potential tree into one slightly higher up in the tree. This leads to a set of leaf signatures of length \( n - 1 \). If one of them is realizable, the original one is realizable, too. By iterative application of this method we get a set of leaf signatures of length 1 where the existence of an \( \mathcal{L}'(k) \)-tree can be checked.
easily. In Section 5.3 we refine the algorithm to get a polynomial running time.

For two integers \( a_1, a_2 \in \mathbb{Z} \) we define

\[
\omega_k(a_1, a_2) := \min\{a_1, a_2\} - \max\left\{1, \left\lfloor \frac{k - |a_1 - a_2|}{2} \right\rfloor \right\}. \tag{5.2}
\]

The idea of the core algorithm relies on the following observation.

**Proposition 5.1.** A leaf signature \( D = \{d_1, \ldots, d_n\}, \ n \geq 2 \), is realizable if and only if there exist \( i, j \) with \( 1 \leq i < j \leq n \) such that \( D \setminus \{d_i, d_j\} \cup \{\omega_k(d_i, d_j)\} \) is realizable.

**Proof:** Assume that \( D \) is realizable and let \( T \) be an \( L'(k) \)-tree realizing \( D \). Then there must be two leaves \( v \) and \( w \) of \( T \) that have a common parent \( u \). Let \( d_i \) and \( d_j, i < j \), be the depth limits from \( D \) assigned to \( v \) and \( w \).

Let \( d : V(T) \to \mathbb{N} \) be the depth of the vertices of \( T \) with respect to the length of the edges. We show that \( d(u) \leq \omega(d_i, d_j) \). As \( T \) is an \( L'(k) \)-tree realizing \( D \) we have \( d(u) \leq d(v) - 1 \leq d_i - 1, d(w) \leq d(u) \leq d(v) - 1 \leq d_j - 1 \) and \( k = (d(v) - d(u)) + (d(w) - d(u)) \leq d_i + d_j - 2d(u) \). We conclude, using that \( d(u) \in \mathbb{N} \):

\[
d(u) \leq \min\{d_i, d_j\} - 1
\]

and

\[
d(u) \leq \left\lfloor \frac{d(v) + d(w) - k}{2} \right\rfloor = \min\{d_i, d_j\} - \left\lfloor \frac{k - |d_i - d_j|}{2} \right\rfloor.
\]

Therefore we have \( d(u) \leq \omega_k(d_i, d_j) \) and the tree \( (V(T) \setminus \{v, w\}, E(T) \setminus \{uv, uw\}) \) realizes \( D \setminus \{d_i, d_j\} \cup \{\omega_k(d_i, d_j)\} \).

On the other hand assume, that there are \( i, j \in \{1, \ldots, n\}, i < j \), such that \( D' = (D \setminus \{d_i, d_j\}) \cup \{\omega_k(d_i, d_j)\} \) is realizable. Let \( T \) be an \( L'(k) \)-tree realizing \( D' \) and let \( v \) be the leaf of \( T \) assigned to \( \omega_k(d_i, d_j) \). Add two edges of length \( a = \max\{k - 1, d_i - \omega_k(d_i, d_j)\} \) and \( b = k - a \) at \( v \). Then the two newly inserted leaves have depth at most \( d_i \) and \( d_j \) and thus the new tree realizes \( D \).

By repeatedly applying Proposition 5.1 we get the algorithm outlined at the beginning of the section. Unfortunately, the number of leaf signatures that are computed grows exponentially in the length of the initial leaf signature.
5.3 Refined Algorithm

In this section we refine the algorithm outlined in the previous section in order to get a polynomial running time. The refinement depends on the observations that the values of a leaf signature cannot be too big and that we can restrict ourselves to leaf signatures identical to the original signature except for the bottom $k$ layers.

First we introduce the notion of domination which will be useful in the proofs.

Definition 5.2. Let $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$ be two leaf signatures of the same length. If there exists a permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ such that $a_i \leq b_{\pi(i)}$ for all $i \in \{1, \ldots, n\}$ then $A$ is dominated by $B$.

Obviously, it is sufficient to consider only sets of leaf signatures, where no signature is dominated by another. Next we note that we can assume the values of a leaf signature not to be too big.

Proposition 5.3. The leaf signature $D = \{d_1, \ldots, d_n\}$ is realizable if and only if the leaf signature $D' = \{\min\{d_1, (k-1)(n-1)\}, \ldots, \min\{d_n, (k-1)(n-1)\}\}$ is realizable.

Proof: Assume there is an $L'(k)$-tree $T$ realizing $D$. As $T$ contains $n$ leaves, every root-leaf-path consists of at most $n - 1$ edges. Moreover, in an $L'(k)$-tree every edge has length at most $k - 1$. We conclude that the leaves of $T$ have depth at most $\min\{d_1, (k-1)(n-1)\}, \ldots, \min\{d_n, (k-1)(n-1)\}$ and therefore $D'$ is realizable.

The other direction of the proof follows from the fact, that $D'$ is dominated by $D$.

Now we show that we can assume each computed leaf signature to be identical to the original signature except for the bottom $k$ layers.

Proposition 5.4. The leaf signature $D = \{d_1, \ldots, d_n\}$ is realizable if and only if there exist $i, j$ with $1 \leq i < j \leq n$ such that the signature

$$(\min\{d_1, \omega\}, \ldots, \min\{d_n, \omega\}) \setminus \{\min(d_i, \omega), \min(d_j, \omega)\} \cup \{\omega_k(d_i, d_j)\} \quad (5.3)$$

with $\omega = \omega_k(d_i, d_j) + k - 1$ is realizable.

Proof: Let $T$ be an $L'(k)$-tree realizing $D$, let $u$ be an internal vertex of $T$ of maximum depth and denote by $v, w$ the two children of $u$. Obviously, $v$ and $w$ are leaves of $T$. Let $d_i$ and $d_j$, $1 \leq i < j \leq n$, be the values assigned to
and $w$. By the proof of Proposition 5.1 we have $d(u) \leq \omega_k(d_i, d_j)$. As $u$ is an internal vertex of maximum depth and all edges in $T$ have length at most $k - 1$, all leaves of $T$ have depth of at most $d(u) + k - 1 \leq \omega_k(d_i, d_j) + k - 1$. Deleting the leaves $v, w$ and their incident edges from $T$ we obtain a tree realizing the signature (5.3).

On the other hand assume that there are $i$ and $j$ with $1 \leq i < j \leq n$ such that $A = \{\min(d_1, \omega), \ldots, \min(d_n, \omega)\} \setminus \{\min(d_i, \omega), \min(d_j, \omega)\} \cup \{\omega_k(d_i, d_j)\}$ is realizable for $\omega = \omega_k(d_i, d_j) + k - 1$. Note, that $A$ is dominated by $A' = (\{d_1, \ldots, d_n\} \setminus \{d_i, d_j\}) \cup \{\omega_k(d_i, d_j)\}$ and thus $A'$ is realizable. Then by Proposition 5.1 the leaf signature $D$ is realizable. □

The above results lead us to Algorithm 3 which computes the sets $\mathcal{M}_z$ of leaf signatures of length $z$ iteratively, starting with $\mathcal{M}_n$ containing only the initial leaf signature of length $n$.

**Input:** An instance $D = \{d_1, \ldots, d_n\}$ and $k \in \mathbb{N}, k \geq 2$.

**Output:** Returns true iff there exists an $\mathcal{L}'(k)$-tree for $D$.

1 $\mathcal{M}_n \leftarrow \{D\}$;
2 for $z = n - 1$ to 1 do
3   $\mathcal{M}_z \leftarrow \emptyset$;
4   foreach leaf signature $A = \{a_1, \ldots, a_{z+1}\} \in \mathcal{M}_{z+1}$ do
5     $\mathcal{B}_A \leftarrow \emptyset$;
6     foreach $i, j \in \{1, \ldots, z+1\}, i < j$ do
7       Compute leaf signature $B$ out of $A$ by replacing $a_i$ and $a_j$
8       by $\omega_k(a_i, a_j)$ and truncating large values in $B$ according to
9       Proposition 5.3 and Proposition 5.4;
10      $\mathcal{B}_A \leftarrow \mathcal{B}_A \cup \{B\}$;
11   end
12   Remove all signatures from $\mathcal{B}_A$ that are dominated by other
13   signatures in $\mathcal{B}_A$;
14   $\mathcal{M}_z \leftarrow \mathcal{M}_z \cup \mathcal{B}_A$;
15 end
16 if $\{i\} \in \mathcal{M}_1$ for some $i \geq 0$ then return true;
17 return false;

**Algorithm 3:** $\mathcal{L}'(k)$-tree decision algorithm

Figure 5.2 shows the sets $\mathcal{M}_z, z \in \{1, \ldots, 6\}$, computed by the algorithm.
for the instance \(\{5, 7, 7, 8, 8, 9\}\) and \(k = 6\). Dominated signatures are removed. The arrows show where each new signature comes from. As the final set \(M_0\) contains the leaf signature \(\{0\}\), the initial signature is realizable.

Let \(D = \{d_1, \ldots, d_n\}\) be the input of the algorithm. In order to be able to reproduce the reduction steps we introduce a function \(p(\cdot)\) that assigns a leaf signature \(B\) to the signature it replaces, that is, \(p(B) = A\) for \(A\) and \(B\) as in Line 7 of the algorithm. For a leaf signature \(B\) we denote by \(l(B)\) the minimum value of an element that has been added to \(B\), that is, \(l(D) = \infty\) and \(l(B) = \min\{l(A), \omega(a_i, a_j)\}\) for \(A, B, i, j\) as in Line 7. This implies that we have only removed and changed elements \(b\) with \(b > l(B)\).

Thus if \(B = \{b_1, \ldots, b_z\}\) with \(b_1 \leq \ldots \leq b_z\) and \(d_1 \leq \ldots \leq d_z\) then

\[b_i = d_i \text{ for all } i \leq \max\{j| b_j < l(B)\} = \max\{j| d_j < l(B)\}.\] (5.4)

The algorithm computes all necessary leaf signatures according to Proposition 5.3 and Proposition 5.4. Consequently, a leaf signature in \(M_z\) is realizable if and only if a leaf signature in \(M_{z+1}\) is realizable for all \(z \in \{1, \ldots, n-1\}\) proving the correctness of the algorithm.

For simplicity of notation we set \(m(A) = \max_{a \in A} a\). We will prove now that the running time of Algorithm 3 is polynomially bounded in \(n\) for fixed \(k\). To this end we show that the sizes of the sets \(M_z\) are polynomially bounded in \(n\). First we show that for every computed leaf signature \(B\) the largest element in \(B\) is at most \(k - 1\) larger than the smallest element that has been inserted into \(B\) by the algorithm.
Proposition 5.5. If \( z \in \{1, \ldots, n - 1\} \), then all leaf signatures \( B \in \mathcal{M}_z \) satisfy
\[
m(B) \leq l(B) + k - 1. \tag{5.5}
\]

Proof: For contradiction we suppose that \( B \) is a set of maximum cardinality computed by the algorithm that contradicts (5.5).

Set \( A = p(B) \). Obviously, \( l(B) \leq l(A) \) and \( m(B) \leq m(A) \). If \( l(B) = l(A) \) then \( m(A) \geq m(B) > l(B) + k - 1 = l(A) + k - 1 \), i.e. \( A \) also does not satisfy (5.5), contradicting the maximality of \( B \).

Thus \( l(B) < l(A) \). But in this case \( l(B) = \omega(a_i, a_j) \) for the two elements \( a_i, a_j \in A \) that are replaced. As all values of \( B \) are truncated in Line 7 of the algorithm according to Proposition 5.4, we obtain
\[
m(B) \leq m(B) \leq \omega(a_i, a_j) + k - 1 = l(A) + k - 1, \tag{5.6}
\]
which is a contradiction and completes the proof. \( \square \)

Using the previous result we show that the size of \( \mathcal{M}_z \) is polynomially bounded in \( z \).

Lemma 5.6. If \( z \in \{1, \ldots, n - 1\} \), then
\[
|\mathcal{M}_z| \leq z^k. \tag{5.6}
\]

Proof: Define \( \mathcal{M}_z^i := \{ A \in \mathcal{M}_z : l(A) = i \} \) for \( i \in \mathbb{N} \). By Proposition 5.3 and the truncation of large values in Line 7 of the algorithm according to Proposition 5.3, we know \( \mathcal{M}_z^i = \emptyset \) for \( i > (z - 1)(k - 1) \).

Let \( i \in \{0, \ldots, (n - 1)(k - 1)\} \) and \( A = \{a_1, \ldots, a_z\} \in \mathcal{M}_z^i \) such that \( a_1 \leq \ldots \leq a_z \). Recall that \( D = \{d_1, \ldots, d_n\}, d_1 \leq \ldots \leq d_n \), is the input of the algorithm. Set \( t = \max\{j : d_j < l(A)\} \). By the definition of \( l(A) \) and (5.4), we conclude
\[
a_j = d_j \tag{5.7}
\]
for \( j \in \{1, \ldots, t\} \). On the other hand, by Proposition 5.5,
\[
a_j \in \{l(A), \ldots, l(A) + k - 1\} \tag{5.8}
\]
for \( j \in \{t + 1, \ldots, z\} \).

This implies that the sets \( A \in \mathcal{M}_z^i \) only differ in the largest \( (z - t) \) elements and each of these elements can only take one of \( k \) different values. Thus the number of different sets \( A \) (without removing dominated ones) is at most the number of integral partitions of \( z - t \) into the sum of \( k \) non-negative integers. This number equals \( \binom{z-t+k-1}{k-1} \) and is bounded by \( \frac{(z-t)^{k-1}}{k-1} \leq \frac{z^{k-1}}{k-1} \).

Altogether we have at most \( 1 + (z - 1)(k - 1) \) sets \( \mathcal{M}_z^i \) that are non-empty and each of these sets has at most \( z^{k-1} \) elements. Hence
\[ |\mathcal{M}_z| = \sum_{i \in \mathbb{N}} |\mathcal{M}_z^i| \leq (1 + (z - 1)(k - 1)) \frac{z^{k-1}}{k-1} \leq z^k. \quad (5.9) \]

This finishes the proof. \( \square \)

Before we can prove the running time of the algorithm we show that for each leaf signature \( A \) the size of the set \( B_A \) is not too big.

**Lemma 5.7.** For any leaf signature \( A = \{a_1, \ldots, a_{z+1}\} \) the set \( B_A \) contains at most \( k \cdot z \) elements. Moreover, the set \( B_A \) can be computed in time \( O(kn^2) \).

**Proof:** W.l.o.g. assume \( a_1 \leq \ldots \leq a_{z+1} \). For every pair \( i, j \in \{1, \ldots, z+1\}, i \leq j \), a new signature \( B \) is computed out of \( A \) in the foreach loop in Line 6. We denote this signature by \( B_{i,j} \).

First note that \( \omega_k(a_i, a_j) = a_i - \max\{1, \lceil (k - a_j + a_i)/2 \rceil \} \) and thus

\[ a_i - k + 1 \leq a_i - \lceil k/2 \rceil \leq \omega_k(a_i, a_j) \leq a_i - 1. \quad (5.10) \]

Now assume that there exist indices \( i, j, j', 1 \leq i \leq j \leq j' \leq z + 1 \), such that \( \omega = \omega_k(a_i, a_j) = \omega_k(a_i, a_{j'}) \). Then \( B_{i,j} = (B_{i,j} \setminus \{\min\{a_{j'}, (z - 1)(k - 1), \omega + k - 1\}\}) \cup \{\min\{a_j, (z - 1)(k - 1), \omega + k - 1\}\} \), that is, \( B_{i,j} \) is dominated by \( B_{i,j'} \). We conclude that the maximum number of signatures in \( B_A \) depends on the number of possible values for \( a_i \) and \( \omega \), respectively.

Instead of traversing all values for \( i \) and \( j \) in order to compute \( B_A \) it suffices to traverse all possible values of \( i \) and \( \omega \). In order to construct a new signature for given \( i \) and \( \omega \) we have to traverse \( z + 1 \) elements, implying the total running time of \( O(kz^2) \).

Joining the previous results together we are able to prove that the running time is polynomially bounded in the size of the input.

**Theorem 5.8.** For a given leaf signature \( D = \{d_1, \ldots, d_n\} \) it can be decided in time \( O(n^{k+3}) \) if there exists an \( \mathcal{L}'(k) \)-tree with \( n \) leaves at depth at most \( d_1, \ldots, d_n \). Moreover, such a tree can be constructed in the same running time.

**Proof:** To achieve this running time we combine Algorithm 3 and the idea of Lemma 5.7 in order to compute the sets \( B_A \). By Lemma 5.7 each set \( B_A \) can be computed in time \( O(kz^2) \) for \( z + 1 = |A| \). Applying Lemma 5.6 the total running time is bounded by

\[ O\left( \sum_{z \in \{1, \ldots, n-1\}} kz^2 |\mathcal{M}_z| \right) \leq O\left( \sum_{z \in \{1, \ldots, n-1\}} kz^{k+2} \right) \leq O\left( n^{k+3} \right). \quad (5.11) \]
As we have seen before, there exists an $L'(k)$-tree realizing the signature $D$ if and only if $\{i\} \in \mathcal{M}_1$ for an $i \geq 0$. This tree can be constructed using the predecessor function $p(\cdot)$.

In practice the running time can be improved by removing dominated signatures.

**Remark 5.9.** By removing dominated signatures after each iteration of the main loop (see Line 13 of Algorithm 3), the cardinalities of the sets $\mathcal{M}_z$ and the running time of the algorithm can be reduced significantly in practice. Nevertheless, it seems that the theoretical worst case running times do not decrease in general.

### 5.4 Conclusion and Future Work

In this chapter we have presented an algorithm building rooted binary trees with choosable edge lengths in polynomial time for fixed $k$. This algorithm can be seen as a generalization of Huffman coding: If $k = 2$ we are in the case of ordinary binary trees with all edges having length 1. In this case it is easy to show that for any leaf signature $A = \{a_1, \ldots, a_z\}$ the set $B_A$ only contains the signature we get by replacing the largest two elements $a_i, a_j$, $1 \leq i < j < z$, by $\omega_2(a_i, a_j) = \min\{a_i, a_j\} - 1$ (after removing dominated signatures). Thus $|\mathcal{M}_z| = 1$ for all $z \in \{1, \ldots, n\}$ and the algorithm is equivalent to Huffman coding.

It is still open if there is an algorithm for the $L'(k)$-tree with a significantly better running time, for example an algorithm with a running time that is polynomially bounded not only in $n$ but also in $k$. 
Chapter 6

The Art Gallery Problem

Another well known problem that combines geometrical and combinatorial aspects is the art gallery problem: Given the layout of an art gallery represented by a polygon, the task is to place a minimum number of guards such that each point in the gallery can be seen by a guard. There are many reasonable variants of this problem, starting with the restriction that the guards have to be placed on vertices, or the guards having only a limited field of vision, up to guards that are not placed at a fixed position but can move along a line.

In this chapter we consider the classical variant where the guards cannot move, have no limited field of vision and can be placed anywhere within the gallery. We are interested in computing lower bounds for the number of guards required to guard a gallery without holes that is given as a polygon. A fundamental result in this context is the following.

Theorem 6.1 (Chvátal [11]). At most \( \left\lfloor \frac{n}{3} \right\rfloor \) guards are necessary to guard a gallery containing \( n \) vertices and no holes.

This theorem has first been proved by Chvátal [11]. Later Fisk [17] gave a very short and nice proof which we outline here: First the gallery is triangulated (see Figure 6.1). Then the vertices are colored using 3 colors such that any two adjacent vertices have different colors. Placing guards at the positions of vertices colored with the least used color yields a solution with at most \( \left\lfloor \frac{n}{3} \right\rfloor \) guards: As the three vertices of each triangle have different colors, on one of them a guard is placed and thus the triangle is guarded.

The bound in Theorem 6.1 is tight as can be seen by considering the so-called comb example in Figure 6.2.

If we consider the case of rectilinear galleries (also called orthogonal galleries) where each wall is either parallel to the x- or to the y-axis we get a tighter result:
Figure 6.1: An art gallery (left) and a triangulation of it (right)

Figure 6.2: A gallery that requires at least \( \left\lfloor \frac{n}{3} \right\rfloor \) guards

**Theorem 6.2** (Kahn et al. [41]). *In a rectilinear art gallery without holes, \( \left\lfloor \frac{n}{4} \right\rfloor \) guards are sufficient to guard the gallery.*

Unfortunately, for this theorem no proof is known that is as simple as the proof of Fisk for the general case.

First Kahn et al. [41] extended the idea of Fisk to rectilinear galleries by looking for a convex quadrilateralization of the gallery. If such a quadrilateralization is given, then the vertices can be colored using 4 colors and such that each of the four vertices of each quadrilateral has different colors. As in the proof of Fisk positioning guards at vertices with the least used color, we obtain a feasible solution with at most \( \left\lfloor \frac{n}{4} \right\rfloor \) guards. Kahn et al. have shown that a convex quadrilateralization always exists, but it is not easy to compute. Alternative proofs for Theorem 6.2 were found by Győri [26], Lubiw [49] and O’Rourke [54].

In this chapter we introduce a new method based on perfect graphs that can be used to bound the number of guards needed to guard rectilinear art galleries in terms of vertices and to guard polyominoes in terms of area and perimeter, where a *polyomino* is the union of a finite number of lattice squares. Unless otherwise stated the polyominoes and art galleries considered in this chapter do not contain holes, that is, if \( P \) is the closure of all points
inside the gallery, then \( \mathbb{R}^2 \setminus P \) contains only one connected region.

The bounds in terms of vertices and area are already known. One new result proved in this chapter is the following theorem.

**Theorem 6.3.** The minimum number of guards necessary to guard a polyomino is at most \( \max \{1, \left\lfloor \frac{\ell}{6} \right\rfloor \} \) where \( \ell \) denotes the perimeter of the polyomino.

This variant of the art gallery problem was proposed by Jäger and Rautenbach [36], who originally conjectured the statement of Theorem 6.3.

Section 6.2 contains a new proof of the rectilinear art gallery Theorem 6.1 based on properties of perfect graphs, which are recalled in Section 6.1. Section 6.3 contains the proof of Theorem 6.3. Section 6.4 gives a new proof for the area bound of Bield et al. [4] for guarding polyominoes.

The main results in this chapter are published in [50].

### 6.1 Preliminaries

In this section we introduce the idea of our method based on maximal rectangles within an art gallery and their connection to guards and perfect graphs.

Let \( \mathcal{R} \) be the set of all maximal axis-parallel rectangles that are inside the gallery. A rectangle \( R \) is maximal if it is inside the gallery and there is no other rectangle \( R' \neq R \) inside the gallery with \( R \subset R' \).

We have to specify which points inside a gallery can be seen from a point \( v \in \mathbb{R}^2 \). In the original setting a point \( v \) covers (or sees) a point \( w \) if the line segment \( vw \) is part of the gallery. An alternative model of visibility is the so called \( r \)-visibility. Two points \( v \) and \( w \) are \( r \)-visible if the minimum axis-parallel rectangle containing \( v \) and \( w \) is completely inside the gallery. Note that \( r \)-visibility implies visibility.

An \( r \)-star is a polygon \( P \) such that there exists a point \( v \) with \( P = \{ w \mid v \text{ and } w \text{ are } r\text{-visible} \} \). Note that this implies that \( P \) is the union of all maximal rectangles \( R \in \mathcal{R} \) such that \( v \in R \). One method to find an upper bound on the number of guards is to find a set of \( r \)-stars covering the gallery. Our approach is slightly different. Instead of looking for a minimum number of guards that cover the gallery we want to find a minimum number of guards \( \tau(\mathcal{R}) \) such that each maximal rectangle \( R \in \mathcal{R} \) contains at least one of them. Such a set of guards or points is a transversal of \( \mathcal{R} \) and \( \tau(\mathcal{R}) \) is the transversal number of \( \mathcal{R} \). It is obvious that such a set of guards covers the gallery. There are galleries such that \( \tau(\mathcal{R}) \) is greater than the minimum number of \( r \)-stars necessary to cover the gallery (see Figure 6.3). Nevertheless, transversals yield upper bounds for the number of guards necessary to guard a gallery.
Figure 6.3: A gallery where the transversal number $\tau(\mathcal{R})$ is 3 (a), but where 2 $r$-stars are sufficient to cover the gallery (b).

To bound $\tau(\mathcal{R})$ from above we investigate the intersection graph of $\mathcal{R}$ with vertex set $\mathcal{R}$ and edges between pairs of rectangles with nonempty intersection. A clique cover of a graph $G$ is a set of cliques $C_1, \ldots, C_k$ in $G$ such that every vertex $v \in V(G)$ lies in at least one of these cliques. For a guard $p$ let $\mathcal{R}_p \subseteq \mathcal{R}$ be the set of maximal rectangles that contain $p$. Since all rectangles in $\mathcal{R}_p$ have $p$ in common, they form a clique in the intersection graph $G_{\mathcal{R}}$. On the other hand, consider a clique $C \subseteq \mathcal{R}$ in $G_{\mathcal{R}}$, that is, any pair of rectangles $R, R' \in C$ has a non-empty intersection. By Helly’s theorem [29] there exists a point $q$ that is contained in all rectangles in $C$. We conclude that every transversal of $\mathcal{R}$ corresponds to a clique cover of $G_{\mathcal{R}}$ of the same size and vice versa. Due to this fact we denote the minimum size of a clique cover of a graph $G$ by $\tau(G)$.

Next we prove that in the case of rectilinear galleries $\tau(G_{\mathcal{R}})$ equals the maximum size of an independent set of $G_{\mathcal{R}}$. In order to do this, we introduce some notation and known results from graph theory. For a graph $G$ let a vertex coloring of $G$ be a function $c : V(G) \to \mathbb{N}$ such that no two adjacent vertices receive the same color, that is, $c(v) \neq c(w)$ for all $\{v, w\} \in E(G)$. The chromatic number $\chi(G)$ is the smallest number of colors needed to color the vertices of $G$. The vertices of the same color form an independent set in $G$. Thus a coloring of $G$ with $k$ colors is equivalent to a partitioning of $V(G)$ into $k$ independent sets.

The clique number $\omega(G)$ of $G$ is the number of vertices in a maximum clique in $G$. Since every vertex in a clique receives a different color, we have $\chi(G) \geq \omega(G)$. A graph $G$ is perfect if for every induced subgraph $G'$ of $G$, the clique number equals the chromatic number, that is, $\omega(G') = \chi(G')$.

We denote the maximum size of an independent set in a graph $G$ by $\alpha(G)$. It is easy to see that every clique in $G$ corresponds to an independent set in the complement $\bar{G}$ of $G$ and every vertex coloring of a graph $G$ corresponds to a clique cover in $\bar{G}$. Thus

$$\alpha(G) = \omega(\bar{G}) \leq \chi(\bar{G}) = \tau(\bar{G}).$$

(6.1)
By the Weak Perfect Graph Theorem [48], the complement of a perfect graph is itself perfect. This immediately implies that for a perfect graph $G$ the equation
\[
\alpha(G) = \tau(G)
\] (6.2)
holds.

Now we consider again the set $\mathcal{R}$ of maximal rectangles and its intersection graph $G_\mathcal{R}$. Shearer [57] proved the following lemma.

**Lemma 6.4** (Shearer [57]). *The intersection graph of maximal rectangles in a rectilinear polygon without holes is perfect.*

By (6.2), we obtain $\alpha(G_\mathcal{R}) = \tau(G_\mathcal{R})$, that is, the minimum size of a clique cover equals the maximum size of an independent set in $G_\mathcal{R}$. A maximum independent set in $G_\mathcal{R}$ corresponds to a set of pairwise disjoint rectangles $\mathcal{P} \subseteq \mathcal{R}$ (also called a *packing* of $\mathcal{R}$) of maximum size.

**Proposition 6.5.** Let $\mathcal{R}$ be the set of maximal rectangles in a rectilinear gallery $M$. The maximum size of a packing of $\mathcal{R}$ is an upper bound for the number of guards necessary to cover $M$.

Note that the number of maximum rectangles inside a gallery $M$ is polynomially bounded in the number of vertices of $M$. Thus the intersection graph $G$ of maximal rectangles in $M$ has polynomial size. Grötschel et al. [24] described a polynomial time algorithm to compute a minimum size vertex coloring in perfect graphs. Applying this algorithm on $\bar{G}$ we get a clique covering $\mathcal{C}$ of $G$ of size $\alpha(G)$ which is equal to $\tau(G)$ by Proposition 6.2. Finally, we can place for every clique $C$ of $\mathcal{C}$ a guard in the non-empty intersection of the rectangles in $C$. In summary, there exists a polynomial time algorithm to compute the positions of $\tau(G)$ guards covering the gallery.

### 6.2 The Rectilinear Art Gallery Theorem

Our method yields a new proof of the rectilinear art gallery Theorem 6.2 using packings of maximal rectangles.

*Proof.* By Proposition 6.5 the minimum number of guards that are sufficient to guard the gallery is bounded by the maximum size a set $\mathcal{P}$ of pairwise disjoint maximal rectangles that are inside the gallery. To complete the proof we show that every maximal rectangle contains at least 4 vertices of the gallery.

Let $R \in \mathcal{P}$ and $(a, b) \in \mathbb{R}^2$ be the lower left corner of $R$. Let $\{(a, y) \mid b' \leq y \leq b''\}$ be the bottommost wall segment that is part of the left side of $R$
and let \( \{(x, b) \mid a' \leq x \leq a''\} \) be the leftmost wall segment that is part of the lower side of \( R \). Due to the maximality of \( R \) such segments with \( b' < b'' \) and \( a' < a'' \) exist. If \((a, b') = (a', b) = (a, b)\) then \((a, b)\) is a vertex of the gallery. Otherwise, at least one of \((a, b')\) and \((a', b)\) is distinct from \((a, b)\) and is thus a vertex of the gallery. By analogous arguments for the other three corners of \( R \) we obtain that each rectangle \( R \in \mathcal{P} \) contains at least 4 vertices of the gallery.

As the rectangles in \( \mathcal{P} \) are pairwise disjoint, no vertex is in more than one of them and hence \(|\mathcal{P}| \leq \lfloor \frac{n}{4} \rfloor \).

\[ \square \]

### 6.3 A Bound in Terms of the Perimeter

In this section we prove Theorem 6.3 using the concept of maximum rectangle packings. To this end consider a polyomino \( M \), which might contain holes.

In order to give a bound in terms of the perimeter \( \ell \) of \( M \), we split up the boundary of \( M \) into segments of length 1, where a segment is a set \( \{(x, b) \mid a < x < a + 1\} \) or \( \{(a, y) \mid b < y < b + 1\} \) for some \( a, b \in \mathbb{Z} \). As all vertices have integral coordinates, either all points or none of the points of a segment belongs to the boundary of the polygon. A wall segment is a segment that is part of the boundary. Let \( W \) be the set of all wall segments. Hence the perimeter \( \ell \) of \( M \) is equal to \(|W|\).

Now let \( \mathcal{P} \subseteq \mathcal{R} \) be a maximum size packing. If \( \ell = 4 \), then \(|\mathcal{P}| = 1 \). Our goal is to show \(|\mathcal{P}| \leq \frac{\ell}{6} \) for \( \ell \geq 6 \). Removing \( \bigcup_{R \in \mathcal{P}} R \) from \( M \) separates the polyomino into disjoint connected regions. Let \( \mathcal{Q} \) be the set of the closures of the maximal connected regions of \( M \setminus \bigcup_{R \in \mathcal{P}} R \).

Consider the bipartite graph \( G \) with vertex set \( \mathcal{P} \cup \mathcal{Q} \) and edge set \( E \) where the edges \( E \) are the pairs of regions \( \{R, Q\}, R \in \mathcal{P}, Q \in \mathcal{Q}, \) that intersect (see Figure 6.4). Note that such regions \( \mathcal{Q} \) and such a graph \( G \) can be constructed for any rectilinear gallery and not only for polyominoes. For \( v \in V(G), \) we denote by \( d_G(v) = |\{w \in V(G) \mid \{v, w\} \in E(G)\}| \) the degree of \( v \) in \( G \).

Using this bipartite graph we count the number of wall segments that are contained in each region in \( V(G) \). First we consider the maximal rectangles.

**Lemma 6.6.** Each maximal rectangle in a polyomino contains at least 4 wall segments.

**Proof.** A maximal rectangle contains a wall segment in each direction. Otherwise, we can enlarge the rectangle in a direction that does not contain such a segment which is a contradiction to its maximality. \( \square \)
If one guard is sufficient to cover the gallery, Theorem 6.3 holds. Thus we assume that we need at least 2 guards. By Proposition 6.5, we have $|\mathcal{P}| \geq 2$.

In addition, note that each wall segment belongs to exactly one region of $\mathcal{P} \cup \mathcal{Q}$.

**Lemma 6.7.** If $\mathcal{Q}$ and $G$ are defined as above, then each region $Q \in \mathcal{Q}$ contains at least $d_G(Q)$ wall segments.

**Proof.** Let $Q \in \mathcal{Q}$, $d = d_G(Q)$ and let $p$ be a point on the border of $Q$. We traverse the border of $Q$ counterclockwise, starting at $p$, until we reach $p$ again. During this traversal we pass the borders of the $d$ rectangles $R_1, \ldots, R_d$ (in this order) that are adjacent to $Q$, that is, $\{Q, R_j\} \in E(G)$ and hence $Q \cap R_j \neq \emptyset$. For simplicity let $R_{d+1} = R_1$. Since any two consecutive rectangles $R_j$ and $R_{j+1}$ are disjoint, there is at least one wall segment on the path between these rectangles and these wall segments are in $Q$ due to its maximality. We conclude that there are at least $d$ wall segments in $Q$. \(\square\)

**Lemma 6.8.** If $\mathcal{P}$ and $G$ are defined as above and if $R \in \mathcal{P}$ is a vertex of degree 1 in $G$, then there are at least 5 wall segments in $R$.

**Proof.** First observe that $R$ contains at least two squares as otherwise it is not maximal. Say the northern and the southern edge of $R$ have length at least 2, that is, they contain at least two segments. By Lemma 6.6, there
are 4 wall segments on the border of $R$, one in each direction. Note that $R$ has only one neighbor in $G$. Hence either all segments of the northern edge or all segments of the southern edge of $R$ are wall segments. Thus this edge contains at least 2 wall segments.

Now we give a lower bound for the number of wall segments in terms of $|P|$.

**Lemma 6.9.** A polyomino (with possible holes) with a maximum size packing $\mathcal{P}$ of maximal rectangles contains at least $6|\mathcal{P}|$ wall segments.

**Proof.** Let $G$ and $Q$ be defined as above and denote by $\sigma : \mathcal{P} \cup Q \to \mathbb{N}$ the number of wall segments contained in each region. As $G$ is bipartite, each edge is incident to one vertex of $Q$ and we obtain

$$\sum_{v \in Q} d_G(v) = |E(G)|. \quad (6.3)$$

Let $L$ be the set of all rectangles in $\mathcal{P}$ that have degree 1 in $G$. Then every rectangle in $L$ is incident to one edge of $G$ and every rectangle of $\mathcal{P} \setminus L$ is incident to at least two edges of $G$. Since all these edges are disjoint we conclude

$$|E(G)| \geq 2|\mathcal{P}| - |L|. \quad (6.4)$$

Therefore the number of wall segments satisfies

$$\sum_{v \in \mathcal{P} \cup Q} \sigma(v) = \sum_{v \in \mathcal{P}} \sigma(v) + \sum_{v \in Q} \sigma(v) \geq \sum_{v \in \mathcal{P}} \sigma(v) + \sum_{v \in Q} d_G(v) \quad (\text{Lemma 6.7})$$

$$= \sum_{v \in \mathcal{P}} \sigma(v) + |E(G)| \quad \text{(by (6.3))}$$

$$\geq 5|L| + 4(|\mathcal{P}| - |L|) + |E(G)| \quad (\text{Lemmas 6.6 and 6.8})$$

$$\geq 5|L| + 4(|\mathcal{P}| - |L|) + 2|\mathcal{P}| - |L| \quad (\text{by (6.4)})$$

$$= 6|\mathcal{P}|. \quad \square$$

Theorem 6.3 now immediately follows from Lemma 6.9 and Proposition 6.5. Figure 6.5 shows the well known “comb” example, proving that the bound of Theorem 6.3 is tight.

Note that Lemma 6.9 is true for packings of maximal rectangles in any polyomino, even if the polyomino contains holes. We were not able to prove that the packing number is also an upper bound for the number of guards necessary to guard a gallery with holes. However, we feel that this is still true in the general case.
Figure 6.5: A gallery where \( \ell \) guards are necessary

**Conjecture 6.10.** In any rectilinear gallery which might contain holes the maximum size of a packing of maximal rectangles is an upper bound on the number of guards required to guard the gallery.

If this conjecture is true, then by the above observations Theorem 6.3 is also true for polyominoes with holes. Note that in rectilinear polygons with holes the intersection graph of the maximal rectangles is not necessarily perfect. However, this is no contradiction to our conjecture. Figure 6.6 shows a gallery with one hole in which the intersection graph of the maximal rectangles is not perfect because it contains an induced cycle of length 5. Nevertheless, there is a packing of size 3, which is the minimum number of guards necessary to guard the gallery.

Figure 6.6: A gallery with one hole in which the intersection graph of the maximal rectangles is not perfect

### 6.4 Guarding \( m \)-polyominoes

A polyomino that is the union of \( m \) squares is called an \( m \)-polyomino. Note that an \( m \)-polyomino and can have holes. Recently Biedl et al. [4] proved the following bound in terms of the area of a polyomino:

**Theorem 6.11** (Biedl et al. [4]). At most \( \left\lfloor \frac{m+1}{4} \right\rfloor \) guards are necessary to guard an \( m \)-polyomino.
In this section we show that our bound on the number of guards in terms of the perimeter implies this bound in terms of the area.

First consider the case that the gallery does not contain a hole. Let \(P, Q\) and \(G\) be defined as above. Again, we assume that \(|P| \geq 2\). Let \(a(P)\) be the area of a polygon \(P \in P \cup Q\).

**Lemma 6.12.** For all \(R \in P\) and \(Q \in Q\) we have \(a(R) \geq 2\) and \(a(Q) \geq d_G(Q) - 1\).

**Proof.** First observe that every maximal rectangle contains at least two unit squares, that is, \(a(R) \geq 2\) for all \(R \in P\). By Lemma 6.7, each region \(Q \in Q\) contains at least \(d_G(Q)\) wall segments. Moreover, it contains at least \(d_G(Q)\) segments in its boundary that are not wall segments, one for each neighbor of \(Q\) in \(G\). Thus the perimeter of \(Q\) is at least \(2d_G(Q)\). The area of a connected polyomino with perimeter \(2k\) is at least \(k - 1\). As \(Q\) itself is a connected polyomino the claim follows directly. \(\square\)

The graph \(G\) is connected and thus

\[|E(G)| \geq |V(G)| - 1 = |Q| + |P| - 1. \tag{6.5}\]

We conclude using Lemma 6.12

\[m = \sum_{R \in P} a(R) + \sum_{Q \in Q} a(Q) \geq 2|P| + \sum_{Q \in Q} (d_G(Q) - 1) \quad \text{(Lemma 6.12)}\]

\[\geq 2|P| - |Q| + |E(G)| \quad \text{(by (6.3))}\]

\[\geq 3|P| - 1. \quad \text{(by (6.5))}\]

By Proposition 6.5, \(|P|\) is an upper bound for the number of guards necessary to guard the gallery. This proves Lemma 6.11 in the case that the gallery has no holes.

If there are holes we add additional walls to make the gallery hole-free. This can be done by connecting holes to other holes or to the exterior with new walls (see e.g. [55]) such that the interior of the gallery is a connected region with no hole. For an example see Figure 6.4 (a),(b). By this procedure we increase the number of walls and more guards might be necessary, but the area of the gallery does not change.

Denote by \(\hat{M}\) the new gallery and let \(\hat{M}\) be the interior of \(\hat{M}\), that is, all points of \(\hat{M}\) that are not on a wall. We “shrink” the gallery by setting

\[M' = \{(x, y) \in \mathbb{R}^2 \mid \max_{p \in \hat{M}} ||p - (x, y)||_1 \geq \epsilon\}\]

for a fixed \(\epsilon > 0\) (Figure 6.4 (c)). The new \(M'\) does not contain any holes and there exists an obvious one-to-one correspondence between the maximum
Figure 6.7: The transformation for the proof of Theorem 6.11 for polyominoes with holes. (a) Original polyomino with holes. (b) The new gallery $M$ after inserting additional walls. (c) The gallery $M'$. (d) $M'$ with packing $P'$ of maximal rectangles. (e) The corresponding maximal rectangles in the gallery $M$.

rectangles in $M$ and $M'$. Let $P'$ be a maximal packing of maximum rectangles in $M'$ and define $Q'$ and $G'$ as above (Figure 6.4 (d)). Let $P$ and $Q$ be the sets we obtain by replacing the regions in $P'$ and $Q'$ by their corresponding regions in $M$ and let $G$ be the corresponding graph with vertex set $P \cup Q$ (Figure 6.4 (e)). Some of the rectangles in $P$ might share the same walls and vertices of $M$, but the interiors of the regions $P \cup Q$ are pairwise disjoint. Hence, we apply Lemma 6.5 and the method for hole-free polyominoes on $P, Q$ and $G$. We conclude that at most $\lfloor \frac{m+1}{3} \rfloor$ guards are sufficient to guard $M$ and thus $\lfloor \frac{m+1}{3} \rfloor$ guards are also sufficient to guard the original polyomino with holes. This proves Lemma 6.11 in the general case.
Chapter 7

Bibliography


