Extensions of d-Dimensional Arrangement Problems

Paul Stahr

Born 1st April 1993 in Münster, Germany

18th January 2019

Master's Thesis Mathematics

Advisor: Dr. Ulrich Brenner

Second Advisor: Prof. Dr. Jens Vygen

Forschungsinstitut für Diskrete Mathematik

Mathematisch-Naturwissenschaftliche Fakultät der

Rheinischen Friedrich-Wilhelms-Universität Bonn

Contents

1. Introduction

1.1. Motivation

In the d-dimensional arrangement problem, d-DIMAP(see Definition 3.1.6), we search for an embedding of a given undirected graph G with minimal edge length into the d-dimensional lattice. This problem can be motivated by VLSI placement and chip design where we have to embed many circuits and minimize wires between them. It also appears in other fields like the research of breast cancer [27]. Variants of this problem have weighted edges, different domains, different metrics and blockages or even terminals. There are many variants of this problem, we could consider hyperedges, bounded domains or different distance functions.

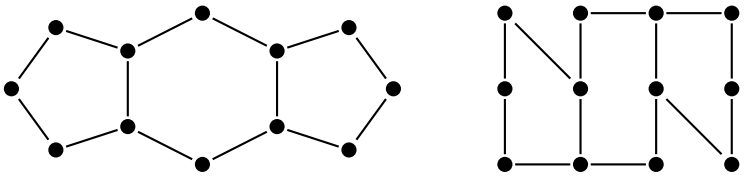

Figure 1.1.: Example of optimal embedding of a graph into a 4×3 grid.

1.2. Previous work

Even the one-dimensional arrangement problem is NP-complete which was shown by Even and Shiloah in 1975 [10]. In 1989 Hansen first showed an $\mathcal{O}(\log^2(|\,\mathrm{V}\,|))$ approximation to the unbounded d-DIMAP[13], by using the divide and conquer scheme with 1/10-9/10 seperators from Leighton and Rao [19]. In 2009, Arora, Rao and Vazirani improved these seperators which leads to a bound of $\mathcal{O}(\log^{3/2}|\,\mathrm{V}\,|)$ for Hansens algorithm [3]. Even, Naor, Rao and Schieber showed an $\mathcal{O}(\log|\,\mathrm{V}\,|\cdot\log\log|\,\mathrm{V}\,|)$-approximation to the d-linear arrangement problem d-LAP in 2000. In 2007 Charikar, Makarychev and Makarychev presented a Divide and Conquer algorithm for d-LAP with an approximation bound of $\mathcal{O}(\sqrt{\log|\,\mathrm{V}\,|})$ and claimed that this implies the same approximation bound for d-DIMAP with Hilbert space-filling curves [9]. As Rotter and Vygen pointed out in 2013, they lose a factor of $\mathcal{O}(\log|\,\mathrm{V}\,|)$ [26]. In the same paper, by combining the algorithm of [9] and [11], Vygen and Rotter attain an $\mathcal{O}(\log|\,\mathrm{V}\,|)$ approximation to the cube-bounded and unbounded d-DIMAP. In 2013, this result was generalized to hypergraphs by Ahrens [1].

He also gave a new approach for a derandomization of the algorithm by pessimistic estimators, but his cunstruction contains an error. In 2014, Gupta and Sidiropoulus considered the cube bounded and unbounded d-DIMAP with fixed points [12]. They got an approximation of $\mathcal{O}(\sqrt[d]{|\mathcal{T}|} \cdot \log|\mathrm{V}|)$ for the unbounded and $\mathcal{O}(|\mathcal{T}| \cdot \log|\mathcal{T}| \cdot \log|\mathrm{V}|)$ for the cube bounded case. Brenner, Rotter and Schorr discovered in 2014 that [26] can be applied with some adjustments on rectangles as well [5] attaining the same asymptotic bounds as Rotter and Vygen.

In 2012, Oswald, Reinelt and Wiesberg developed an exact branch-and-cut scheme for the bounded 2-DIMAP which is applicable to small graphs of about thirty vertices in reasonable runtime [23].

1.3. Our approach

In this thesis we are mainly interested in the cuboid-bounded case of the d-dimensional arrangement problem. This means, that we want to embed a hypergraph into a d-dimenional lattice which is bounded by a cuboid, minimizing the weighted edge lengths. We build on the algorithms of [26], [5], [11] and [1].

The algorithm presented and tested in this thesis works as follows: First a so-called spreading LP is solved. This LP is a lower bound to the embedding problem and provides a graph metric which is used to build a tree metric. This tree is then transformed into a linear embedding. In the last step, a Hilbert space-filling curve is used to rearrange the linear embedding into a cuboid of arbitrary dimension and size.

The algorithm results in a bound of $\mathcal{O}(\log|\mathrm{V}|)$ for the unbounded and the cuboid bounded case of d-DIMAP. Given blockages it results in a bound of $\mathcal{O}(\sqrt[d]{|\mathcal{B}|} \cdot \log|\mathrm{V}|)$ for the cube bounded case of d-DIMAP. Both bounds are an improvement to the currently known bounds. We have generalized the bound $\mathcal{O}(\log|\mathrm{V}|)$ from cubes [26] and rectangles [5] to cuboids and for cubes with blockages we changed the approximation bound from $\mathcal{O}(|\mathcal{B}| \cdot \log|\mathcal{B}| \cdot \log|\mathrm{V}|)$ of [12] to $\mathcal{O}(\sqrt[d]{|\mathcal{B}|} \cdot \log|\mathrm{V}|)$. Furthermore, by correcting the construction of the tree metric in [1] we show that this algorithm can be implemented without randomness.

Algorithm 1: d-Dimensional embedding Algorithm

input : Hypergraph $G = (V, H)$
Cuboid boundaries $b_1, \ldots, b_d \in \mathbb{N}$
Blockages $\mathcal{B} \subseteq \bigtimes_{i=1}^{d} [b_i]$

output: Injection $\pi : V \to \bigtimes_{i=1}^{d} [b_i] \setminus \mathcal{B}$

```
foreach Connected component do
    Precalculate LP-lower bounds
    while Violated LP inequality exists do
        Add one or more violated inequalities
        Resolve spreading-LP
        Post-optimize solution
    end
    Generate tree metric
    Optimize tree
    Create linear embedding
end
Use Hilbert space-filling curve to transform linear into cuboid embedding
```

In Chapter 2 we introduce basic notations and definitions which we are using in this thesis, followed by the main problems in Chapter 3.

In Chapter 4, we introduces the spreading LP, first mentioned in [9]. We will show how to adjust the spreading bound to the case of cuboids and mention some crucial optimizations to run the spreading LP at least for small instances in reasonable runtime.

Tree metrics - and how they are built - are the topic of Chapter 5. We combine the proofs of [11] and [1] to get a bound with the same constant as in [11] for hypergraphs. We then introduce two derandomizations. We generalize the one of [11] to hypergraphs and the one of [1]. For both we add details about the implementation.

In Chapter 6, we introduce the Hilbert space-filling curve. We show a possible generalization to cuboids. Then we prove a general bound for the infinite space and show a geometric bound for the case of cuboids.

In Chapter 7, we prove our two main theorems, the approximation bounds of our algorithm, first the one for the cube-bounded and unbounded MLA with blockages and then for the cuboid bounded MLA without blockages.

In Chapter 8, we give some test results of the program and compare the different algorithms. We show that there is no clear winner of the two presented LP-formulations

and a useful selection depends on the graph. Furthermore, we show that our optimizations indeed result in lower calculation times. Moreover, we compare the results of the different tree metric algorithms on different graphs.

In Appendix A.1, we describe the program which was written for this thesis. We will describe the usage and the most important parameters.

Our main result is the generalization of an approximation bound of $\mathcal{O}(\log|\mathrm{V}|)$ to the embedding of hypergraphs in cuboids. This generalizes the results of [26], [5] and [1] by one deterministic algorithm. We correct the ideas of [1] for a derandomization. Furthermore, we show, that in the presence of blockages this algorithm only gets worse by a factor of $\mathcal{O}(\sqrt[d]{|\mathcal{B}|})$ if we are embedding into a cube of arbitrary dimension.

1.4. Acknowledgement

First of all, I thank my advisor Dr. Ulrich Brenner for the discussions about problems presented in this thesis, and numerous useful remarks. I thank my second advisor Jens Vygen, for reading this thesis, and Dr. Marcus Oswald, for the test instances he send and explained to me.

I was delighted about the useful remarks by the careful readers Helene Glöckner, Florian Seiffarth and Benedikt Böing.

I am grateful for the love and support my parents, Barbara Stahr, and Rolf Stahr, and my girlfriend, Helene, always gave to me.

2. Definitions and notation

In this chapter we introduce the basic definitions and notations we use in this thesis.

2.1. Notation

We will use the following conventions:

- $\mathbb{N}$ are the positive natural numbers
- $\mathbb{R}_+$ for the non-negative and $\mathbb{R}_{>0}$ for the positive real numbers
- $\subset$ for strict subset, whereas $\subseteq$ is not necessarily strict

Definition 2.1.1 (Index set)**.** We will name the **index set** $\{1, \ldots, n\}$ by $[n]$ for every $n \in \mathbb{N}$.

Definition 2.1.2 (Symmetric group)**.** For an arbitrary finite set S we call an injective function $\pi : [|S|] \to S$ a **permutation** of S. We denote the set of all permutations of S by $\mathcal{S}(S)$, also called the **symmetric group**. Furthermore, given a number $k \in [|S|]$ we call an injection $\pi : [k] \to S$ a **partial permutation** of S.

Definition 2.1.3 (Binomialcoefficient)**.** Let S be an arbitrary finite set and $n \in \{0, \ldots, |S|\}$, then we identify

$$\binom{S}{n} := \{T \subseteq S : |T| = n\}$$

2.2. Definitions

Definition 2.2.1 (Hypergraph)**.** A **hypergraph** is a generalization of a graph in which edges can have more then two endpoints. In this thesis we will not consider parallel edges, hence we define a hypergraph as a tuple $\mathrm{G} = (\mathrm{V}, \mathrm{H})$, where V is an arbitrary finite set, and $\mathrm{H} \subseteq \mathcal{P}(\mathrm{V})$ subset of the powerset of V. V is called the set of **vertices** and H the set of **hyperedges**. If we have an conventional graph we will indicate the edges with E.

Definition 2.2.2 (Complete)**.** A graph $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ is called **complete** if $\mathrm{E} = \binom{\mathrm{V}}{2}$.

Definition 2.2.3 (Adjacent edges)**.** Let $\mathrm{G} = (\mathrm{V}, \mathrm{H})$ be a hypergraph. For a vertex $v \in \mathrm{V}$ we call the edges $h \in \mathrm{H}$ with $v \in h$ **adjacent** to v. We denote them by

$$\delta(v) := \{h \in \mathrm{H} \,|\, v \in h\}$$

Definition 2.2.4 (Cut)**.** A **cut** is defined as a set of vertices $\emptyset \subset S \subset \mathrm{V}$. The cut edges are defined as a function $\mathrm{C} : \mathcal{P}(\mathrm{V}) \to \mathcal{P}(\mathrm{H})$:

$$\mathrm{C}(S) := \{h \in \mathrm{H} : h \cap S \neq \emptyset \wedge h \setminus S \neq \emptyset\}$$

Definition 2.2.5 (Star)**.** Let $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ be a graph. G is called a **star** if there exists a vertex $v \in \mathrm{V}$ such that:

$$\mathrm{G} = (\mathrm{V}, \{v\} \times (\mathrm{V} \setminus \{v\}))$$

Definition 2.2.6 (Induced subgraph)**.** Let $\mathrm{G} = (\mathrm{V}, \mathrm{H})$ be a hypergraph and $S \subseteq \mathrm{V}$ a subset of the vertices. We define the **induced subgraph** $\mathrm{G}(S)$ by:

$$\mathrm{G}(S) := (S, \{h \in \mathrm{H} : h \subseteq S\})$$

Definition 2.2.7 (Distances)**.** Let $\mathrm{G} = (\mathrm{V}, \mathrm{H})$ be a hypergraph. Then we call a function $\mathrm{d} : \binom{\mathrm{V}}{2} \to \mathbb{R}_+$ a **distance function**.

Definition 2.2.8 (Graph metric)**.** Let $\mathrm{G} = (\mathrm{V}, \mathrm{H})$ be a hypergraph. Then $\mathrm{d} : \binom{\mathrm{V}}{2} \to \mathbb{R}_+$ is a **graph metric** if

$$\forall u, v, w \in V : \mathrm{d}(u, w) \leq \mathrm{d}(u, v) + \mathrm{d}(v, w)$$

We extend this definition to sets of vertices $S \subseteq \mathrm{V}$:

$$\mathrm{d}(S) := \max_{u,v \in S} \mathrm{d}(u, v) \tag{2.1}$$

We can consider Eq. (2.1) as the **length** of an edge $h \in \mathrm{H}$ or as the **diameter** of some vertex set.
In the algorithm we also need the distance between edges and nodes, thus we define the functions $\tilde{\mathrm{d}}_{\min}, \tilde{\mathrm{d}}_{\max} : (\mathrm{V} \cup \mathrm{H})^2 \to \mathbb{R}_+$ and $\tilde{\mathrm{d}} : (\mathrm{V} \cup \mathrm{H})^2 \to \mathcal{P}(\mathbb{R}_+)$:

$$\begin{aligned}
\tilde{\mathrm{d}}(x, y) &:= \left\{ \mathrm{d}(v, w) \;\middle|\; v \in \begin{cases} x, & \text{if } x \in \mathrm{H} \\ \{x\}, & \text{if } x \in \mathrm{V} \end{cases}, w \in \begin{cases} y, & \text{if } y \in \mathrm{H} \\ \{y\}, & \text{if } y \in \mathrm{V} \end{cases} \right\} \\
\tilde{\mathrm{d}}_{\max}(x, y) &:= \max \tilde{\mathrm{d}}(x, y) \\
\tilde{\mathrm{d}}_{\min}(x, y) &:= \min \tilde{\mathrm{d}}(x, y)
\end{aligned}$$

Definition 2.2.9 (Ball)**.** Let $\mathrm{G} = (\mathrm{V}, \mathrm{H})$ be a hypergraph, $v \in \mathrm{V}$ a vertex, $\mathrm{d} : \binom{\mathrm{V}}{2} \to \mathbb{R}_+$ a graph metric and $r \in \mathbb{R}_+$ a nonnegative number, called the radius. We define the **closed ball** $\mathrm{B} : \mathrm{V} \times \mathbb{R}_+ \to \mathcal{P}(\mathrm{V}) \times \mathcal{P}(\mathrm{H})$ as

$$\mathrm{B}(v, r) := \left(\left\{ u \in \mathrm{V} \text{ with } \mathrm{d}(v, u) \leq r \right\}, \left\{ h \in \mathrm{H} \text{ with } \tilde{\mathrm{d}}_{\max}(v, h) \leq r \right\} \right)$$

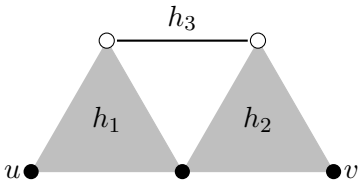

Figure 2.1.: Example of a path $\mathrm{P} = (u, v, \{h_1, h_2\})$. The filled vertices are included in the path. The triple $(u, v, \{h_1, h_2, h_3\})$ is not a path, because every pair of edges has a common vertex.

Definition 2.2.10 (Edge cost)**.** Let $\mathrm{G} = (\mathrm{V}, \mathrm{H})$. We define the **edge costs** as a function $\mathrm{c} : \mathrm{H} \to \mathbb{R}_{>0}$. Furthermore, we define the **edge volume** as a function $\mathrm{w} : \mathrm{H} \to \mathbb{R}_{+}$, $\mathrm{w}(h) := \mathrm{c}(h) \cdot \mathrm{d}(h)$ for $h \in \mathrm{H}$.

Definition 2.2.11 (Connected)**.** Let $\mathrm{G} = (\mathrm{V}, \mathrm{H})$ be a graph. G is **connected** if for every set $\emptyset \neq X \subset \mathrm{V}$ the cut $\mathrm{C}(X)$ is not empty.

$$\mathrm{C}(X) \neq \emptyset$$

Definition 2.2.12 (Path)**.** Let $\mathrm{G} = (\mathrm{V}, \mathrm{H})$ be a hypergraph. We call a triple $\mathrm{P} = (u, v, \mathrm{H_P})$, $u, v \in \mathrm{V}$ and $\mathrm{H_P} \subseteq \mathrm{H}$ a **path** from u to v if there exists an ordering $\mathrm{H_P} = \{h_1, \ldots, h_n\}$ with:

- $\forall i \in [n] : u \in h_i \Leftrightarrow i = 1$ and $v \in h_i \Leftrightarrow i = n$
- $\forall i, j \in [n] : h_i \cap h_j = \emptyset \Leftrightarrow |i - j| > 1$

We denote the edges of the path $\mathrm{P} = (u, v, \mathrm{H_P})$ by $\mathrm{H(P)} := \mathrm{H_P}$. Furthermore, for an edge $h \in \mathrm{H}$ we write $h \in \mathrm{P}$ if $h \in \mathrm{H_P}$ and for a vertex $w \in \mathrm{V}$ we write $w \in \mathrm{P}$ if $w \in \{u, v\}$ or $|\{h \in \mathrm{H_P} \,|w \in h\}| = 2$.
For any $S \subseteq \mathrm{V}$ we denote the set of all paths in G between two different endpoints in S by

$$\mathrm{paths}(S) := \left\{ \mathrm{P} = (u, v, \mathrm{H_P}) \;\middle|\; \{u, v\} \in \binom{S}{2} \text{ and } \mathrm{P} \text{ is a path from } u \text{ to } v \right\}$$

Given distances $\mathrm{d} : \binom{\mathrm{V}}{2} \to \mathbb{R}_{+}$, we define the length of a path $\mathrm{P} = (u, v, \mathrm{H_P})$ by:

$$\mathrm{d(P)} := \sum_{h \in \mathrm{H_P}} \mathrm{d}(h)$$

We call a path $\mathrm{P} = (u, v, \mathrm{H_P})$ a **shortest path** from u to v according to d if $\mathrm{d(P)}$ is minimal.

Remark 2.2.13. For a given path P the order of its hyperedge is unique.

Definition 2.2.14 (Metric closure)**.** Let $G = (V, H)$ be a hypergraph and $d : H \to \mathbb{R}_+$ edge distances. We define the **metric closure** $d : H \cup \binom{V}{2}$ by

$$d(S) := \max_{\{u,v\} \in \binom{S}{2}} \min_{P \in \text{paths}(u,v)} d(P)$$

If the distance function is not specified we always use d^1 which denotes the metric closure for all edges having length one.

3. Problem definition

We introduce now the main problem of this thesis. The task is to assign pairwise different positions in a domain, for example in $\mathbb{Z}^d$, to the vertices of a hypergraph and minimize the length of the hyperedges.

Definition 3.1.1 (Bounding box length)**.** Let $d \in \mathbb{N}$, $k \in \mathbb{N} \cup \{\infty\}$ and $S \subseteq \mathbb{Z}^d$, then the **bounding box length** for $k \neq \infty$ is defined as:

$$\mathrm{BBOX}_k(S) = \sqrt[k]{\sum_{i \in [d]} \left(\max_{s \in S} s_i - \min_{s \in S} s_i \right)^k}$$

For $k = \infty$ we use the limit

$$\mathrm{BBOX}_\infty(S) := \lim_{k \to \infty} \mathrm{BBOX}_k(S) = \max_{i \in [d]} \left(\max_{s \in S} s_i - \min_{s \in S} s_i \right)$$

Remark 3.1.2. For two elements $s, t \in S$ the bounding box length is the distance in the according Minkowski-Metrik with parameter k. Due to the fact that all metrics are equivalent for finite dimensions, our results can be applied to any k.

Remark 3.1.3. The bounding box length is monotone, i.e. $A \subseteq B \subset \mathbb{Z}^d \Rightarrow \mathrm{BBOX}(A) \leq \mathrm{BBOX}(B)$

Definition 3.1.4 (Minimal Linear Arrangement Problem)**.** The Minimal Linear Arrangement Problem, MLA, is defined as follows:

Given:	(Hyper)graph $\mathrm{G} = (\mathrm{V}, \mathrm{H})$
	Cost function $\mathrm{c} : \mathrm{H} \to \mathbb{R}_{>0}$
Task:	Find injection $\pi : \mathrm{V} \to [\,\lvert \mathrm{V} \rvert\,]$ which minimizes
	$\sum_{h \in \mathrm{H}} \mathrm{c}(h) \cdot \mathrm{BBOX}_1(\pi(h))$

Definition 3.1.5 (d-Linear Arrangement Problem)**.** The d-Linear Arrangement Problem, d-LAP, is defined as follows:

Given:	(Hyper)graph $\mathrm{G} = (\mathrm{V}, \mathrm{H})$
	Cost function $\mathrm{c} : \mathrm{H} \to \mathbb{R}_{>0}$
	Dimension $d \in \mathbb{N}$
Task:	Find injection $\pi : \mathrm{V} \to [\,\lvert \mathrm{V} \rvert\,]$ which minimizes
	$\sum_{h \in \mathrm{H}} \mathrm{c}(h) \cdot \sqrt[d]{\mathrm{BBOX}_1(\pi(h))}$

Definition 3.1.6 (d-Dimensional Arrangement Problem)**.** The d-Dimensional Arrangement Problem, d-DIMAP, is defined as follows:

Given: (Hyper)graph $\mathrm{G} = (\mathrm{V}, \mathrm{H})$
Domain $\mathrm{Q} \subseteq Z^d$
Cost function $\mathrm{c} : \mathrm{H} \to \mathbb{R}_{>0}$
Task: Find injection $\pi : \mathrm{V} \to \mathrm{Q}$ which minimizes
$\sum_{h \in \mathrm{H}} \mathrm{c}(h) \cdot \mathrm{BBOX}_1(\pi(h))$

Depending on the choice of the **domain** Q we specialize this problem:

- For $\mathrm{Q} = \mathbb{Z}^d$ we call this problem unbounded.
- Given a boundary $\mathrm{b} \in \mathbb{N}$ and $\mathrm{Q} = [\mathrm{b}]^d$ we call this problem **Cube-Bounded**.
- For two dimensional boundaries $\mathrm{b}_1, \mathrm{b}_2 \in \mathbb{N}$ and $\mathrm{Q} = \mathrm{b}_1 \times \mathrm{b}_2$ we call this problem **Rectangle-Bounded**.
- Given Cuboid boundaries $\mathrm{b}_1, \ldots, \mathrm{b}_d \in \mathbb{N}$ and $\mathrm{Q} = \bigtimes_{i=1}^{d} [\mathrm{b}_i]$ we call this problem **Cuboid-Bounded**.

Remark 3.1.7. The original definition of the problem deals only with the l_1 metric, but naturally we can define it for every Minkowski-Metric, because these metrics are all equivalent. Attaining an approximation bound in one metric directly results in the same bound in the others, except for a constant.

In the following, we will name the optimum LP-solution value of the arrangement problem by OPT.

Definition 3.1.8 (Blockage)**.** Given a dimension $d \in \mathbb{N}$ and a domain $\mathrm{Q} \subseteq \mathbb{Z}^d$ we define the set of **blockages** $\mathcal{B} \subseteq \mathrm{Q}$ as a subset of the domain.

Suppose we are given a bounded d-Dimensional Arrangement Problem with a hypergraph $\mathrm{G} = (\mathrm{V}, \mathrm{H})$ and some blockages $\mathcal{B} \subseteq \mathrm{Q}$. We require that there are no vertices placed on any blockage, i. e. $\pi(\mathrm{V}) \cap B = \emptyset$.

Definition 3.1.9 (Terminal)**.** Given a hypergraph $\mathrm{G} = (\mathrm{V}, \mathrm{H})$ and a domain $\mathrm{Q} \subseteq \mathbb{Z}^d$. We define the set of **terminals** $\mathcal{T} \subseteq \mathrm{V} \times \mathrm{Q}$ as a set of tuples of vertices and positions for with $v \neq w$ and $q_1 \neq q_2$ for all $\{(v, q_1), (w, q_2)\} \in \binom{\mathcal{T}}{2}$

Suppose we are given a bounded d-Dimensional Arrangement Problem, with a hypergraph $\mathrm{G} = (\mathrm{V}, \mathrm{H})$ and terminals $\mathcal{T} \subseteq \mathrm{V} \times \mathrm{Q}$. We require $\pi(v) = p$ for each $(v, p) \in \mathcal{T}$. Consequently, terminals are vertices which are fixed to a position.

4. Spreading LP

4.1. Linear program

Linear programs are a very powerful tool to express many continuous problems. They consist of a system of linear inequalities and a linear objective function. If we are restricted to integral solutions, it depends on the problem if we can solve it by a linear program. Even if this is not possible, one often can derrive useful bounds. For the d-Dimensional arrangement problem an LP will be used to calculate adequate edge lengths, that are in some sense as short as possible, but also fulfill some kind of spreading. For one vertex it is impossible to place all other vertices with distances as short as possible to this vertex. In this chapter we show how to express the d-Dimensional arrangement problem as an LP and what can be done to make this solvable in reasonable time.

Definition 4.1.1 (Linear inequality)**.** Let M be an arbitrary finite set. We define a **linear inequality** as a function $f : \mathbb{R}^{\mathrm{M}} \to \{0, 1\}$ of the form

$$f(x) = \mathbb{1}\left(\sum_{m \in \mathrm{M}} \lambda(m) \cdot x(m) \leq b\right)$$

Where $\lambda : \mathrm{M} \to \mathbb{R}$ and $b \in \mathbb{R}$.
Furthermore, let $x \in \mathbb{R}^{\mathrm{M}}$ be an arbitrary point. Then the inequality is

- valid if $\sum_{m \in \mathrm{M}} \lambda(m) \cdot x(m) \leq b$
- tight if $\sum_{m \in \mathrm{M}} \lambda(m) \cdot x(m) = b$
- slacked if $\sum_{m \in \mathrm{M}} \lambda(m) \cdot x(m) < b$
- violated if $\sum_{m \in \mathrm{M}} \lambda(m) \cdot x(m) > b$

Definition 4.1.2 (Linear program)**.** Let M be an arbitrary finite set. We define a linear program as a tuple (c, I) where $c : \mathrm{M} \to \mathbb{R}$ is the cost function, and I is a set of linear inequalities. We call a vector $x : \mathrm{M} \to \mathbb{R}$ a solution of the linear program if

1. $\forall f \in \mathrm{I} : f(x) = 1$
2. $\sum_{i \in \mathrm{M}} c(i) \cdot x(i)$ is minimal

Example 4.1.3. One can formulate the problem of embedding the complete graph $\mathrm{G} = (\mathrm{V}, \mathrm{H})$ with $\mathrm{V} = \{u, v, w\}$, into the line $\{1, 2, 3\}$. We only consider the distances. We can obtain that the sum of the edge lengths can be bounded from below $\mathrm{d}(u, v) + \mathrm{d}(v, w) + \mathrm{d}(w, u) \geq 4$. If we now try to minimize the sum of the edges we may find the optimal solution $\mathrm{d}(u, v) = \mathrm{d}(v, w) = \mathrm{d}(w, v) = \frac{4}{3}$. Of course, if we embed the graph, we have to select one distance whose value we set to two. In fact, our solution is a combination of the distances of the three possibilities to embed the vertices, precisely the average. There is no possibility of excluding this solution by additional inequalities without excluding one of the three embeddings. This shows that generalizing this problem to continuous solutions neither leads to valid embeddings nor do the distances necessarily correspond to a valid embedding.

Linear programs are solvable in polynomial time with the interior point method [16], but in practice in most cases the simplex algorithm is much faster. A good reference for linear programming is [29].

4.2. Definition

Let $d \in \mathbb{N}$ be the dimension, $\mathrm{G} = (\mathrm{V}, \mathrm{H})$ be a hypergraph, $\mathrm{b} \in \mathbb{N}^d$ be the cube boundary and $\pi : \mathrm{V} \to \bigtimes_{i=1}^{d} [\mathrm{b}_i]$ be an embedding of the vertices. If we consider an arbitrary subset of the vertices $S \subseteq \mathrm{V}$ and a vertex $v \in S$, we obtain lower bounds for the summed distances:

$$\sum_{s \in S} \mathrm{BBOX}_k(\pi(\{v, s\})) \tag{4.1}$$

In this chapter, we are mainly interested in the distances of the vertices, not in their positions. Our goal is to find a graph metric d which is a useful linear relaxation to this problem. We will state three different definitions for a lower bound $\mathrm{sb} : [|V|] \to \mathbb{R}_+$ of (4.1), the so called **spreading bound**. The first two bounds are only for the case $k = 1$. The original spreading bound was defined in 2000 by Even, Naor, Rao and Schieber [9].

Definition 4.2.1 (Spreading bound [9])**.** Let $d \in \mathbb{N}$ be the dimension. Then the unbounded spreading bound is defined by

$$\mathrm{sb}^u(k) := \frac{1}{4} \cdot (k-1)^{1+1/d}$$

An example of this bound can be seen in Fig. 4.1. With increasing number of points the summed distances of the points has to increase.

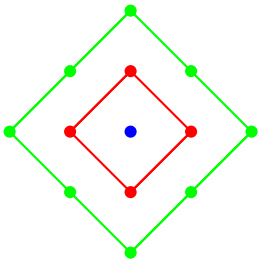

Figure 4.1.: Visualization of the Minkowski l_1-metric. The first four points can be placed in a distance of one. The next eight points have at least distance two, followed by distance three for the next points and so on.

Brenner, Rotter and Schorr showed that one can give an algorithm for the embedding into two-dimensional rectangles by defining a spreading bound which takes into account, that the distances decrease linear at some point, see Fig. 4.2.

Definition 4.2.2 (Spreading bound [5])**.** Let $\mathrm{b} \in \mathbb{N}^2$ be the side lengths of a rectangle. Then the rectangle spreading bound is defined as

$$\mathrm{sb}^r(k) := \frac{1}{4} \cdot k^{\max\left\{\frac{3}{2}, 2 - \frac{\ln \min\{\mathrm{b}_1, \mathrm{b}_2\}}{\ln k}\right\}}$$

For large bounds and small numbers of elements k this bound is equivalent to Definition 4.2.1. In the opposite extreme case we get a quadratic increase of the bound, which corresponds to a linear growth of the distances.

In this thesis, we want to solve the problem on arbitrary cuboids. Currently there is no useful closed formula for an approximation of the lower bound. This is no problem for the algorithm, because we can calculate the exact values of the bound. In the proofs we have more work to do, because we cannot use basic calculations on it. We assume that the point to which the distances of the other vertices are measured is positioned in the middle, because this is the point where this bound attains the smallest values (see Fig. 4.2).

Definition 4.2.3 (Spreading bound)**.** Let $\mathrm{b} \in \mathbb{N}^d$ be the side lengths of the cuboid. Then according to (4.1) we can define the following bound

$$\mathrm{sb}_k(n) := \min_{S \in \binom{[\mathrm{b}_1] \times \ldots \times [\mathrm{b}_d]}{n}} \sum_{s \in S} ||s - \lfloor \mathrm{b}/2 \rfloor||_k \tag{4.2}$$

Remark 4.2.4. The proofs in [9], [5], [26] and [1] based on the bounds of Definition 4.2.1 and Definition 4.2.2 will also work for Definition 4.2.3 with $k = 1$, because it is always at least as accurate as the other ones. Thus, this spreading bound is always greater than or equal to the rectangle or unbounded spreading bound but smaller than or equal to the real value.

Remark 4.2.5. Blockages can be excluded from the boundary $[b_1] \times \ldots \times [b_d]$. This would give an even more realistic bound and thus maybe results in better solutions, but we currently do not know if we can use this to get a better cube bound.

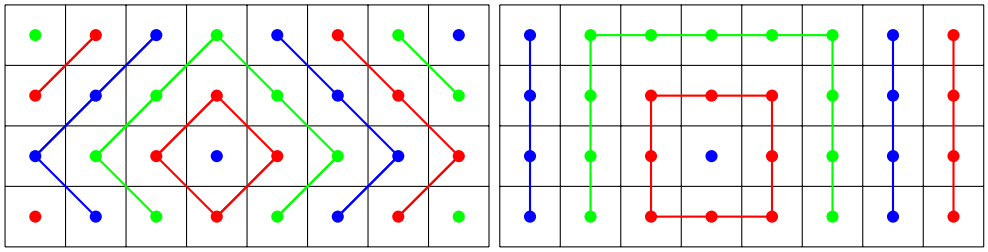

Figure 4.2.: Visualization of the Minkowski l_1- left and l_∞-metric right. The spreading bound is the sum of the distances of the points from the middle point. The left picture is adapted from [5].

Algorithm 2: Spreading bound Algorithm

input : Cuboid boundaries $b_1, \ldots, b_d \in \mathbb{N}$
output: Spreading bound $sb : \prod_{i \in [d]} b_i \to \mathbb{N}$

$j := 1$
$f : \left[\prod_{i \in [d]} b_i \to \mathbb{R} \right]$
foreach $q \in \bigtimes_{i=1}^{d} [b_i]$ **do**
  $f(j) = ||q - \lfloor b/2 \rfloor||_k$
  $j := j + 1$
end
sort(f) such that $f(i) \leq f(j)$ for all $i, j \in \prod_{i \in [d]} b_i$ with $i \leq j$
$sb_k(n) := \sum_{i \in [n]} f(i)$
return sb

Definition 4.2.6 (Optimal embedded set)**.** Let $k \in \mathbb{N} \cup \infty$ be an arbitrary number, $d \in \mathbb{N}$ be the dimension and $b \in \mathbb{N}^d$ be the cuboid boundary. Then we say that a set $S \in \binom{[b_1] \ldots [b_d]}{2}$ is an **optimal embedded set** of size $n \in \mathbb{N}$ if:

$$sb_k(|S|) = \sum_{s \in S} ||s - \lfloor b/2 \rfloor||_k$$

Remark 4.2.7. These are exactly the sets for which $sb_k(n)$ in (4.2) attains its minimum.

Lemma 4.2.8. *Let $d \in \mathbb{N}$ and $\mathrm{b} \in \mathbb{N}^d$. Let S be an optimal embedded set of size n in the maximum norm. Then*

$$\max_{o \in S} ||o - \lfloor \mathrm{b}/2 \rfloor||_\infty \leq 2 \cdot \frac{\mathrm{sb}_\infty(|S|)}{|S|} + 1$$

Proof. Let $m := \max_{o \in S} ||o - \lfloor \mathrm{b}/2 \rfloor||_\infty$ be the maximum and k be the dimension in which it is attained, i.e.:

$$\max_{o \in S} ||o - \lfloor \mathrm{b}/2 \rfloor||_\infty = \max_{o \in S} |o_k - \lfloor \mathrm{b}_k/2 \rfloor|$$

To bound the spreading bound, we split the sum of (4.2) in (4.3). Then we use that the remaining set is a cuboid (4.4). We consider only one dimension and build the average of the distances (4.5) see Fig. 4.3:

$$\begin{aligned}
\mathrm{sb}_\infty(|S|) &= \sum_{s \in S} ||s - \lfloor \mathrm{b}/2 \rfloor||_\infty \\
&= \sum_{s \in S} \begin{cases} ||s - \lfloor \mathrm{b}/2 \rfloor||_\infty, & ||s - \lfloor \mathrm{b}/2 \rfloor||_\infty < m \\ m, & ||s - \lfloor \mathrm{b}/2 \rfloor||_\infty = m \end{cases} && (4.3) \\
&\geq \sum_{s \in S} \begin{cases} |s_k - \lfloor \mathrm{b}_k/2 \rfloor|, & ||s - \lfloor \mathrm{b}/2 \rfloor||_\infty < m \\ m, & ||s - \lfloor \mathrm{b}/2 \rfloor||_\infty = m \end{cases} && (4.4) \\
&= \sum_{s \in S} \begin{cases} \sum_{i=1-m}^{m-1} \frac{i}{|\{1-m,\ldots,m-1\}|}, & ||s - \lfloor \mathrm{b}/2 \rfloor||_\infty < m \\ m, & ||s - \lfloor \mathrm{b}/2 \rfloor||_\infty = m \end{cases} && (4.5) \\
&= \sum_{s \in S} \begin{cases} \sum_{i \in [m-1]} \frac{i \cdot 2}{m \cdot 2 - 1}, & ||s - \lfloor \mathrm{b}/2 \rfloor||_\infty < m \\ m, & ||s - \lfloor \mathrm{b}/2 \rfloor||_\infty = m \end{cases} \\
&= \sum_{s \in S} \begin{cases} \frac{m \cdot (m-1)}{m \cdot 2 - 1}, & ||s - \lfloor \mathrm{b}/2 \rfloor||_\infty < m \\ m, & ||s - \lfloor \mathrm{b}/2 \rfloor||_\infty = m \end{cases} \\
&= \sum_{s \in S} \begin{cases} \frac{m-1}{2-1/m}, & ||s - \lfloor \mathrm{b}/2 \rfloor||_\infty < m \\ m, & ||s - \lfloor \mathrm{b}/2 \rfloor||_\infty = m \end{cases} \\
&\geq \sum_{s \in S} \frac{m-1}{2} \\
&= |S| \cdot \frac{m-1}{2} \\
&= |S| \cdot \frac{\max_{o \in S} ||o - \lfloor \mathrm{b}/2 \rfloor||_\infty - 1}{2}
\end{aligned}$$

□

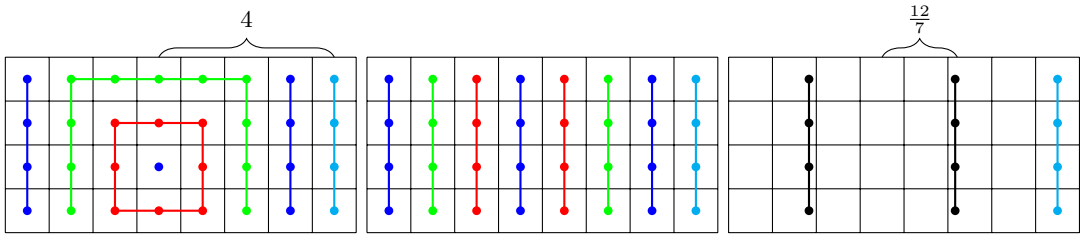

Figure 4.3.: Visualization of the proof with $b = (8, 4)$, $m = 4$. Left: The l_∞ metric (4.3). Middle: Only one dimension is considered (4.4). Right: Only the average is considered (4.5)

Definition 4.2.9 (Spreading LP)**.** Let $\mathrm{G} = (\mathrm{V}, \mathrm{H})$ be a graph, $\mathrm{c} : \mathrm{H} \to \mathbb{R}_{>0}$ be the edge cost. In the **spreading LP** we search for a graph metric $\mathrm{d} : \binom{\mathrm{V}}{2} \cup \mathrm{H} \to \mathbb{R}_+$ which minimizes (4.6) respecting the so called spreading inequalities (4.7), the triangle inequalities (4.8) and the hyperedge inequalities (4.9):

$$\min \sum_{h \in \mathrm{H}} \mathrm{c}(h)\,\mathrm{d}(h) \tag{4.6}$$

$$\mathrm{sb}(|U|) \leq \sum_{u \in U} \mathrm{d}(u, v) \qquad \forall U \subseteq \mathrm{V}, v \in U, |U| \geq 2 \tag{4.7}$$

$$\mathrm{d}(u, w) \leq \mathrm{d}(u, v) + \mathrm{d}(v, w) \qquad \forall \{u, v, w\} \in \binom{\mathrm{V}}{3} \tag{4.8}$$

$$\mathrm{d}(v, w) \leq \mathrm{d}(h) \qquad \forall h \in \mathrm{H}, \{v, w\} \in \binom{h}{2} \tag{4.9}$$

Lemma 4.2.10. *For every Solution of d-*DIMAP*, the bounding box length is a feasible solution of the spreading LP.*

Proof. Function (4.6) corresponds to the Arrangement Problem target function, see Definition 3.1.6. We show each inequality seperately for the bounding box length. For the other bounds see [9] and [5].

- (4.7): For the definition of spreading bounds in this paper the correctness is directly clear by Fig. 4.2.
- (4.8): This is directly satisfied, because the bounding box length for two vertices is a metric.
- (4.9): This follows from the fact, that the bounding box length is monotone, see Remark 3.1.3.

□

Remark 4.2.11. In the case of the spreading bound for cuboids in Definition 4.2.3 one could add the following inequality:

$$\min_{S \in \binom{[\mathrm{b}_1]\times\ldots\times[\mathrm{b}_d]}{|h|}} \max_{s,t\in S} ||s-t||_k \leq \mathrm{d}(h)$$

This gives a more accurate lower bound for hyperedges then the spreading bound (see Section 4.3.3).

4.3. Optimizations

A linear program can be solved in polynomial time by the ellipsoid-method which was shown 1979 by Leonid Khachiyan [17], but in practice the simplex method is much faster. In this chapter, we explain the most important optimizations made to solve this LP in a reasonable runtime. In the following $\mathrm{d} : \binom{\mathrm{V}}{2} \cup \mathrm{H} \to \mathbb{R}_+$ is the intermediate solution of the LP. This means that this is currently no graph metric and any of the three kinds of inequalities in the lp-formulation can be violated.

4.3.1. Sparse spreading LP

We can consider the following LP. Let $\mathrm{G} = (\mathrm{V}, \mathrm{H})$ be a graph, $\mathrm{c} : \mathrm{H} \to \mathbb{R}_{>0}$ be the edge cost and $\mathrm{d} : \mathrm{H} \to \mathbb{R}_+$ be the distances. Then we define the **sparse spreading lp** as:

Definition 4.3.1 (Sparse Spreading LP)**.**

$$\min \sum_{h\in\mathrm{H}} \mathrm{c}(h)\,\mathrm{d}(h)$$

$$\mathrm{sb}(|U|) \leq \sum_{i=2}^{n} \mathrm{d}(\mathrm{P}_i) \qquad \forall n \in \{2,\ldots,|\,\mathrm{V}\,|\}, \{u_1,\ldots,u_n\} \in \tbinom{\mathrm{V}}{n}, \; \forall\, \mathrm{P}_2 \in \mathrm{paths}(u_1,u_2),\ldots,\mathrm{P}_n \in \mathrm{paths}(u_1,u_n)$$

$$\mathrm{d}(v,w) \leq \mathrm{d}(h) \qquad \forall h \in \mathrm{H}\{v,w\} \in \binom{h}{2}$$

This definition leads to a solution with the same value as Definition 4.2.9. Furthermore, if we build the metric closure we get that if we have a solution of this LP we have a solution for the spreading lp and vice versa. In some papers, like [9], it is referred to this formulation. The advantage of this LP is that we only get $|\,\mathrm{H}\,|$ many variables and not $\mathcal{O}(|\,\mathrm{V}\,|^2 + |\,\mathrm{H}\,|)$. But we pay this with much more complex inequalities. Nevertheless, in the tests some sparse graphs are solved faster with this formulation, see Section 8.2. From the presented optimizations in this chapter only the one in Section 4.3.3 and Section 4.3.6 are useful for this LP.

4.3.2. Post-optimization of the LP-solver

Very similar to the pre-calculation of the lower bounds we can make a post-optimization to fix some inequalities by increasing the lengths. So for $S \in \binom{\mathrm{V}}{2}$:

$$\mathrm{d}'(S) := \max\left\{\mathrm{d}(S), \min_{\mathrm{P}\in\mathrm{paths}(S)} \mathrm{d}(\mathrm{P})\right\}$$

This does not change the value of the LP and does not violate any new inequalities. The runtime is again $\mathcal{O}(|\,\mathrm{V}\,|^3)$ with the Floyd-Warshall algorithm [18]. Moreover, this algorithm can easily be parallelized.

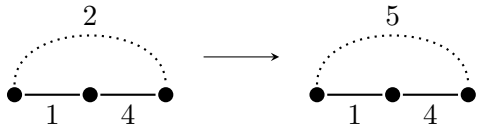

Figure 4.4.: Visualization of the post-optimization. The distance of the two nodes is set to the highest possible distance.

Lemma 4.3.2. *Let* $\mathrm{G} = (\mathrm{V}, \mathrm{H})$ *be a hypergraph,* $\mathrm{d} : \binom{\mathrm{V}}{2} \cup \mathrm{H} \to \mathbb{R}_+$ *be a distance function and* $\mathrm{c} : \mathrm{H} \to \mathbb{R}_{>0}$ *be a cost function. Define* $\mathrm{d}' : \binom{\mathrm{V}}{2} \cup \mathrm{H} \to \mathbb{R}_+$ *by*

$$\mathrm{d}'(S) := \max\left\{\mathrm{d}(S), \min_{\mathrm{P}\in\mathrm{paths}(S)} \mathrm{d}(\mathrm{P})\right\}$$

Then the following statements hold:

- *The solution values* (4.6) *of* d *and* d′ *are the same.*
- *Every valid inequality in* d *is also valid in* d′*.*

Proof. We will show each claim separately:

- The solution values (4.6) of d and d′ are the same:
  For every hyperedge $h \in \mathrm{H}$ we have:

$$\mathrm{d}'(h) \le \min_{\mathrm{P}\in\mathrm{paths}(h)} \mathrm{d}(\mathrm{P}) \le \mathrm{d}(h)$$

- Spreading constraints (4.7):
  Let $v \in \mathrm{V}$ then

$$\sum_{u\in U} \mathrm{d}'(u, v) \ge \sum_{u\in U} \mathrm{d}(u, v) \ge \mathrm{sb}(|U|)$$

- Triangle inequalities (4.8):
  Let $u, v, w \in \mathrm{V}$ and $\mathrm{d}(u,v) + \mathrm{d}(v,w) \geq \mathrm{d}(u,w)$. Then

$$\begin{aligned}
\mathrm{d}'(u,w) &= \max\left\{\mathrm{d}(v,w), \min_{\mathrm{P}\in\mathrm{paths}(u,w)} \mathrm{d}(\mathrm{P})\right\} \\
&\leq \max\left\{\mathrm{d}(v,w), \min_{\mathrm{P}\in\mathrm{paths}(u,v)} \mathrm{d}(\mathrm{P}) + \min_{\mathrm{P}\in\mathrm{paths}(v,w)} \mathrm{d}(\mathrm{P})\right\} \\
&\leq \mathrm{d}(u,v) + \mathrm{d}(v,w) \\
&\leq \mathrm{d}'(u,v) + \mathrm{d}'(v,w)
\end{aligned}$$

- Hyperedge inequalities (4.9)
  Let $h \in \mathrm{H}$, $u, v \in h$ and $\mathrm{d}(h) \geq \mathrm{d}(u,v)$ then

$$\mathrm{d}'(u,v) \leq \min_{\mathrm{P}\in\mathrm{paths}(u,w)} \mathrm{d}(\mathrm{P}) \leq \mathrm{d}(h) \leq \mathrm{d}'(h)$$

□

4.3.3. Pre-calculate lower bounds

For each hyperedge $h \in \mathrm{H}$ we can set

$$lb(h) := \max_{i\in\{2,\ldots,|h|\}} \frac{\mathrm{sb}(i)}{i-1} \tag{4.10}$$

An obvious lower bound for the distance between every pair of nodes is sb(2). If we want to get a better lower bound, we can set

$$lb(u,v) := \min_{\mathrm{P}\in\mathrm{paths}(u,v)} lb(\mathrm{P}) \tag{4.11}$$

This values can be calculated by the Floyd-Warshall algorithm in runtime $\mathcal{O}(|\,\mathrm{V}\,|^3)$. For sparse graphs calling Dijkstra's algorithm from every node with $\mathcal{O}(|\,\mathrm{H}\,|\cdot|\,\mathrm{V}\,|\log|\,\mathrm{V}\,|)$ can be faster [18].

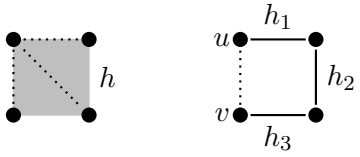

Figure 4.5.: Visualization of lower bounds. Left: A graph where the shortest path between two vertices u and v contains three edges. Thus, $3 \cdot \mathrm{sb}(2)$ is a lower bound for $\mathrm{d}(u,v)$. Right: A hyperedge h which contains four vertices. Thus, $\mathrm{sb}(4)/3$ is a lower bound.

Proposition 4.3.3. *The lower bounds* (4.11) *and* (4.10) *do not change the solution value of the LP.*

Proof. We prove both lower bounds separately

- Node-pair distances:
  Let $\{u, v\} \in \binom{\mathrm{V}}{2}$ and let d be an optimal solution. Performing the post-optimization yields to a solution

  $$\begin{aligned} \mathrm{d}'(u,v) &= \max\{\mathrm{d}(u,v), \min_{\mathrm{P}\in\mathrm{paths}(u,v)} \mathrm{d}(\mathrm{P})\} \\ &\geq \min_{\mathrm{P}\in\mathrm{paths}(u,v)} \mathrm{d}(\mathrm{P}) \end{aligned}$$

  By the correctness of the post-optimization we can restrict the LP to solutions with this lower bound without increasing the solution value (4.6).

- Hyperedge lengths:
  Let $i \in [|h|]$, $U \in \binom{h}{i}$ and $v \in U$. Then

  $$\begin{aligned} (i-1)\cdot \mathrm{d}(h) &\geq \sum_{u\in U\setminus\{v\}} \mathrm{d}(u,v) \\ &= \sum_{u\in U} \mathrm{d}(u,v) \\ &\geq \mathrm{sb}(|U|) \\ &= \mathrm{sb}(i) \end{aligned}$$

□

4.3.4. Triangle inequalities

The number of triangle inequalities (4.8) is polynomial in the input. Nevertheless the best way is to start with an empty LP and check its inequalities by the oracle. Otherwise, each optimization step of the LP-solver would be extremly slow. Moreover, we would have to store $\frac{|\mathrm{V}|^2\cdot(1+|\mathrm{V}|)}{2}$ inequalities, which will make this LP unsolvable if we have many vertices. For some tests see Section 8.2.

4.3.5. Weight violated inequalities

A natural choice of the violated inequality is to take the one with the highest Euclidean distance in the geometric representation of the LP. Let $\lambda : S \in \binom{\mathrm{V}}{2} \cup \mathrm{H} \to \{-1, 1\}$ be the coefficients and $b \in \mathbb{R}$ the bound of an inequality in the LP:

$$\sum_{S\in\binom{\mathrm{V}}{2}\cup\mathrm{H}} \mathrm{d}(S)\cdot\lambda(S) \leq b$$

If this inequality is violated, then the distance of the hyperplane representing the inequality and the point representing the current solution is

$$\frac{\sum_{S \in \binom{\mathrm{V}}{2} \cup \mathrm{H}} \mathrm{d}(S) \cdot \lambda(S) - b}{\sqrt{\sum_{S \in \binom{\mathrm{V}}{2} \cup \mathrm{H}} \lambda(S)^2}}$$

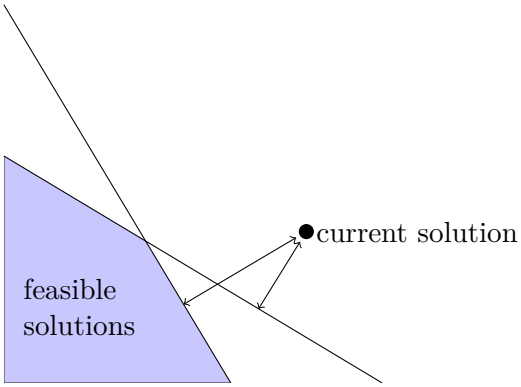

Figure 4.6.: Image of geometric LP representation. The current solution is violating two inequalities. The Euclidean distances are visualised by the two arrows.

4.3.6. Pre-calculate upper bound

Let $\mathrm{G} = (\mathrm{V}, \mathrm{H})$ be a graph and $S \in \mathrm{H} \cup \binom{\mathrm{V}}{2}$ be a set. We can give the following upper bound for S:

$$ub(S) := \max_{i \in [|\,\mathrm{V}\,|]} \{\mathrm{sb}(i) - \mathrm{sb}(i-1)\} \tag{4.12}$$

Lemma 4.3.4. *The upper bound* (4.12) *does not change the solution value nor makes the LP infeasible.*

Proof. We show this for each inequality separately

- Optimization function (4.6):
  This does not decrease with higher distances. Therefore, there is no reason to set an edge higher than needed.

- Spreading inequality (4.7)
  Let $v \neq w$ be two vertices with $\mathrm{d}(v, w) > lb(v, w)$. Without loss of generality let

$U \subseteq \mathrm{V}$ with $\{u, v\} \subseteq U$ be a set for which $\mathrm{sb}(|U|) \leq \sum_{u \in U} \mathrm{d}(u, v)$ is tight. Then we get the following contradiction:

$$\begin{aligned}
\mathrm{sb}(|U|) &= \sum_{u \in U} \mathrm{d}(u, v) \\
&> \sum_{u \in U \setminus \{w\}} \mathrm{d}(u, v) + lb(v, w) \\
&\underset{(4.12)}{\geq} \sum_{u \in U \setminus \{w\}} \mathrm{d}(u, v) + \mathrm{sb}(|U|) - \mathrm{sb}(|U| - 1) \\
&\underset{(4.7)}{\geq} \mathrm{sb}(|U|)
\end{aligned}$$

- Triangle inequality:
  Let $S \subset \binom{\mathrm{V}}{2}$ be all pairs for which $\mathrm{d}(u, v)$ is maximal. Then the triangle inequality (4.8) cannot become violated if we decrease these distances by an arbitrary small value.
- Hyperedge inequality:
  Let $h \in \mathrm{H}$ be a hyperedge with $\mathrm{d}(h) > lb(h)$ then there must exist a conventional edge $e \in \mathrm{H}$ with $\mathrm{d}(e) > lb(e)$. Otherwise, we can decrease $\mathrm{d}(h)$.

□

Remark 4.3.5. If we apply both upper and lower bounds, one needs to make sure that they do not contradict each other. This can be solved by adjusting the upper bound to the maximum of the upper and lower bound.

5. Tree metrics

In this chapter we will show how to construct a tree metric with the graph metric, given by the chapter before. The calculated tree is then transformed into a linear embedding.

5.0.1. Trees

Definition 5.0.1 (Tree)**.** Let $\mathrm{T} = (\mathrm{V_T}, \mathrm{E_T})$ be a graph and $\mathrm{r_T} \in \mathrm{V_T}$ a vertex called the **root**. Then we call $\mathrm{T} = (\mathrm{V_T}, \mathrm{E_T}, \mathrm{r_T})$ a **tree** if

- T is connected
- $|\,\mathrm{E_T}\,| = |\,\mathrm{V_T}\,| - 1$

Remark 5.0.2. Let $\mathrm{T} = (\mathrm{V_T}, \mathrm{E_T}, \mathrm{r_T})$ be a tree and $u, v \in \mathrm{V_T}$ be two vertices. Then there is exactly one path from u to v in T.

Definition 5.0.3 (Ancestor)**.** Let $\mathrm{T} = (\mathrm{V_T}, \mathrm{E_T}, \mathrm{r_T})$ be a tree and $u, v \in \mathrm{V_T}$ be two vertices v is an **ancestor** of u if v lies on the unique path from $\mathrm{r_T}$ to u, i.e. for $\mathrm{P} \in \mathrm{paths}(\mathrm{r_T}, u)$ we have $v \in \mathrm{P}$. Moreover, v is a **child** of u if in addition $\delta(u) \cap \delta(v) \neq \emptyset$.

Definition 5.0.4 (Leaves)**.** Let $\mathrm{T} = (\mathrm{V_T}, \mathrm{E_T}, \mathrm{r_T})$ be a tree. The set of **leaves** is defined as $\mathrm{Leaves_T} := \{v \in \mathrm{V} : (|\delta(v)| = 1 \text{ and } v \neq \mathrm{r_T}) \text{ or } \delta(v) = \emptyset\}$. Furthermore, we define $\mathrm{Leaves_T}$ as a function $\mathrm{Leaves_T} : \mathrm{V_T} \to \mathcal{P}(\mathrm{Leaves_T})$. For $u \in \mathrm{V}$ let $\mathrm{P}_u \in \mathrm{paths}(\mathrm{r_T}, u)$ be the unique path from $\mathrm{r_T}$ to u, then for any vertex $v \in \mathrm{V_T}$:

$$\mathrm{Leaves_T}(v) := \{u \in \mathrm{Leaves_T} \,|\, v \in \mathrm{P}_u\}$$

A tree is **full** if there exists a $k \in \mathbb{N}$ such that

$$\forall v \in \mathrm{Leaves_T} : \mathrm{d}^1_{\mathrm{T}}\, \mathrm{r_T}, v\mathrm{T} = k$$

Definition 5.0.5 (Fan-out)**.** Let $\mathrm{T} = (\mathrm{V_T}, \mathrm{E_T}, \mathrm{r_T})$ be a tree. We define the **fan-out** or the number of children as a function $\mathrm{F_T} : \mathrm{V} \to \mathbb{N}$ by

$$\mathrm{F_T}(v) := |\{w \in \mathrm{V_T} \,|\, w \text{ is child of } v\}|$$

Definition 5.0.6. (Cluster function) Let $\mathrm{T} = (\mathrm{V_T}, \mathrm{E_T}, \mathrm{r_T})$ be a tree, S an arbitrary set and $\Lambda : \mathrm{V_T} \to \mathcal{P}(S)$. We call Λ a **cluster function** if

- $\Lambda(\mathrm{r_T}) = S$
- $\forall v \in \mathrm{V_T} : \Lambda(v) = \bigcup_{l \in \mathrm{Leaves_T}(v)} \Lambda(l)$
- $\forall u, v \in \mathrm{Leaves_T} : \Lambda(u) \cap \Lambda(v) = \emptyset$

5.1. Separated trees

Definition 5.1.1 (2-hierarchically well-separated)**.** Let $\mathrm{T} = (\mathrm{V_T}, \mathrm{E_T}, \mathrm{r_T})$ be a tree and $\mathrm{d_T} : \mathrm{E_T} \to \mathbb{R}_+$ edge lengths. We call $(\mathrm{T}, \mathrm{d_T})$ **2-hierarchically well-separated** if there exists a $\rho > 0$ such that for all $e \in \mathrm{E_T}$, $\mathrm{d_T}(e) = \rho \cdot 2^{-k}$ holds, where $k = \min_{v \in e} \mathrm{d}^1_{\mathrm{T}}(\mathrm{r_T}, v)$ is the number of edges in the path from r to e.

Definition 5.1.2 (2-hierarchically closed-well-separated)**.** Let $\mathrm{T} = (\mathrm{V_T}, \mathrm{E_T}, \mathrm{r_T})$ be a tree and $\mathrm{d_T} : \mathrm{E_T} \to \mathbb{R}_+$ edge lengths. We call $(\mathrm{T}, \mathrm{d_T})$ **2-hierarchically closed-well-separated** if there exists an $\rho > 0$ such that for all $e \in \mathrm{E_T}$ the following holds

$$\mathrm{d_T}(e) = \rho \cdot 2^{-k} \cdot \begin{cases} 1, & \text{if } \mathrm{Leaves_T} \cap e = \emptyset \\ 2, & \text{if } \mathrm{Leaves_T} \cap e \neq \emptyset \end{cases}$$

where $k = \tilde{\mathrm{d}}^1_{\min}(\mathrm{r_T}, e)$ is the number of edges in the path from $\mathrm{r_T}$ to e.

In Section 5.1 we can see the comparison of this metrics. One can think of the closed version as the limit, if we increase the level of the unclosed version to infinity. In the 2-hierarchically closed-well-separated trees the length of a path from a vertex to one of its leaves is independent of the choice of the leave. This will be a very useful property.

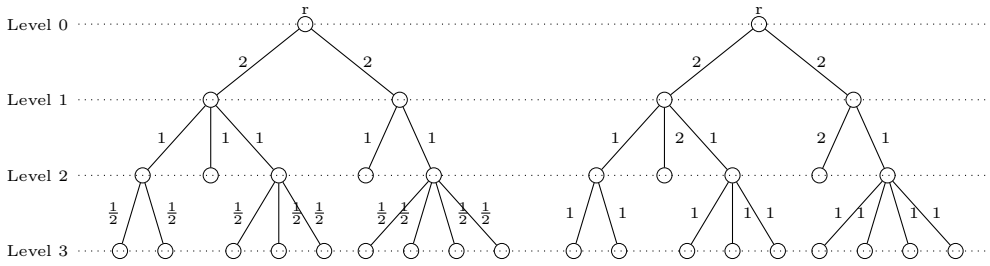

Figure 5.1.: 2-hierarchically well-separated tree, on the left-hand side the one stated in the original definition and on the right-hand side the closed one.

Lemma 5.1.3. *Let* $(\mathrm{T}, \mathrm{d_T})$ *be 2-hierarchically closed-well-separated. Then for every vertex* $v \in \mathrm{V_T}$*, and every child* $w \in \mathrm{Leaves_T}(v)$ *we have that* $\mathrm{d_T}(v, w) = \rho \cdot 2^{1-k}$ *with* $k = \mathrm{d}^1_{\mathrm{T}}(\mathrm{r_T}, v)$

Proof. Let $l := \mathrm{d}_{\mathrm{T}}^1(\mathrm{r}_{\mathrm{T}}, w)$ be the distance from the root to vertex w. We sum the edges on the path P from v to w which is unique according to Remark 5.0.2:

$$\mathrm{d}_{\mathrm{T}}(v,w) = \min_{\mathrm{P}\in\mathrm{paths}(v,w)} \mathrm{d}_{\mathrm{T}}(\mathrm{P}) = \rho \cdot \sum_{i=k}^{l} 2^{-i} \cdot \begin{cases} 1, & \text{if } i < l \\ 2, & \text{if } i = l \end{cases} = \rho \cdot 2^{1-k}$$

□

Lemma 5.1.4. *Let* $(\mathrm{T}, \mathrm{d}_{\mathrm{T}})$ *be 2-hierarchically closed-well-separated. Let* $u, v \in \mathrm{Leaves}_{\mathrm{T}}$ *and* $w \in \mathrm{V}_{\mathrm{T}}$ *be the lowest common ancestor. Then* $\mathrm{d}_{\mathrm{T}}(u,v) = \rho \cdot 2^{2-k}$ *with* $k = \mathrm{d}_{\mathrm{T}}^1(\mathrm{r}_{\mathrm{T}}, w)$.

Proof. By Lemma 5.1.3 we have the following equalities:

$$\mathrm{d}_{\mathrm{T}}(v,w) = \mathrm{d}_{\mathrm{T}}(r,v) + \mathrm{d}_{\mathrm{T}}(r,w) = \rho \cdot 2^{1-k} + \rho \cdot 2^{1-k} = \rho \cdot 2^{2-k}$$

□

Definition 5.1.5. (Induced tree metric) Let $\mathrm{G} = (\mathrm{V}, \mathrm{H})$ be a hypergraph, $\mathrm{T} = (\mathrm{V}_{\mathrm{T}}, \mathrm{E}_{\mathrm{T}}, \mathrm{r}_{\mathrm{T}})$ a tree, $\mathrm{d}_{\mathrm{T}} : \binom{\mathrm{V}_{\mathrm{T}}}{2} \to \mathbb{R}_+$ a graph metric on T and $\Lambda : \mathrm{V}_{\mathrm{T}} \to \mathcal{P}(\mathrm{V})$ a cluster function. Then we call the metric $\mathrm{d}^{\mathrm{T}} : \binom{V}{2} \to \mathbb{R}_+$ the **induced tree metric** if for all $u, v \in \mathrm{V}$, $x, y \in \mathrm{Leaves}_{\mathrm{T}}$ with $u \in \Lambda(x)$ and $v \in \Lambda(y)$:

$$\mathrm{d}^{\mathrm{T}}(u,v) = \mathrm{d}_{\mathrm{T}}(x,y)$$

Theorem 5.1.6 (Tree metric)**.** *Let* $\mathrm{G} = (\mathrm{V}, \mathrm{H})$ *be a hypergraph with* $|\,\mathrm{V}\,| \geq 2$ *and* $\mathrm{d} : \binom{\mathrm{V}}{2} \to \mathbb{R}_+$ *a metric. One can compute a 2-hierarchically well-separated tree* $(\mathrm{T}, \mathrm{d}_{\mathrm{T}})$ *and a cluster function* $\Lambda : \mathrm{V}_{\mathrm{T}} \to \mathcal{P}(\mathrm{V})$ *in polynomial time such that the induced tree metric* $\mathrm{d}^{\mathrm{T}} : \binom{\mathrm{V}}{2} \to \mathbb{R}_+$ *satisfies*

- $\mathrm{d}^{\mathrm{T}}(v,w) \geq \mathrm{d}(v,w),\ \forall v, w \in \mathrm{V}$ *and*
- $\sum_{h\in\mathrm{H}} \mathrm{d}^{\mathrm{T}}(h) \leq \mathcal{O}(\log|\,\mathrm{V}\,|) \cdot \sum_{h\in\mathrm{H}} \mathrm{d}(h)$.

We will present three algorithms to achieve this. First a randomized one and then two derandomizations.

Algorithm 3: TREE METRIC APPROXIMATION

input : Hypergraph $G = (V, H)$
Graph metric $d : \binom{V}{2} \to \mathbb{R}$
Start radius $\rho \in \mathbb{R}_{>0}$
Injection $\tau : V \to [|V|]$
output: Tree $T = (V_T, E_T, r_T)$
Cluster function $\Lambda : V_T \to \mathcal{P}(V)$

Create root r, set $(V_T, E_T, r_T) := (\{r\}, \emptyset, r)$ and $\Lambda(r) := V$
$i = 0$
while $\exists t \in V_T$ *with* $d_T^1(r_T, t) = i$ *and* $|\Lambda(t)| > 1$ **do**
  foreach $t \in V_T$ *with* $d_T^1(r_T, t) = i$ *and* $|\Lambda(t)| > 1$ **do**
    $S := \Lambda(t)$
    for $j = 1$ **to** $|V|$ **do**
      $U := \{w \in S : d(\tau(j), w) \leq \rho \cdot 2^{-i}\}$
      if $U \neq \emptyset$ **then**
        Create new tree node t'
        $\Lambda(t') := U$
        $(V_T, E_T) := (V_T \cup \{t'\}, E_T \cup \{(t, t')\})$
        $S := S \setminus U$
      end
    end
  end
  $i := i + 1$
end
return (T, Λ)

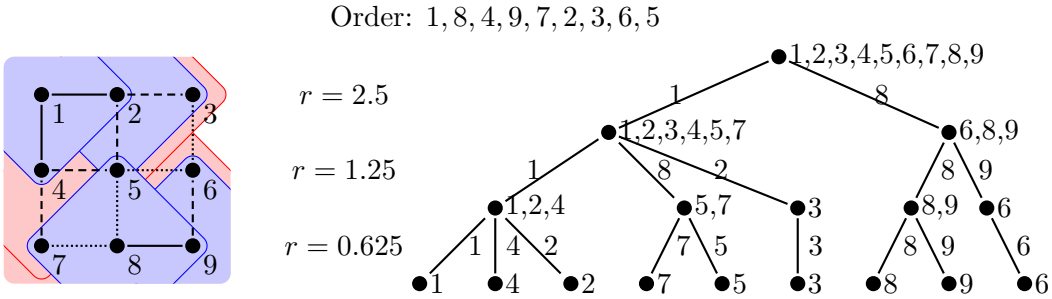

Figure 5.2.: Example of tree metric approximation with $\rho = 3$. The edges of the tree are labeled with the vertex that is chosen from the permutation. The edges are marked dashed for being cut in the first level, densely dotted for the second level and solid for the third level.

Lemma 5.1.7. *For any tree* T *obtained by Algorithm 3 we have* $d(u, v) \leq d^T(u, v)$ *for all pairs* $u, v \in V$. *Furthermore, let* $h \in H$ *be a hyperedge,* $w \in V_T$ *the least common ancestor and* $i := d_T^1(r_T, w)$ *the distance. Then* $d^T(h) = \rho \cdot 2^{2-i}$.

Proof. This directly follows from Lemma 5.1.4. □

5.2. Randomized

Let $G = (V, H)$ be a hypergraph. Let $d : \binom{V}{2} \to \mathbb{R}$ be a graph metric.

Algorithm 4: Tree metric probabilistic approximation

input : Hypergraph $G = (V, H)$
Graph metric $d : \binom{V}{2} \to \mathbb{R}$
output: Tree $T = (V_T, E_T, r_T)$
Cluster function $\Lambda : V_T \to \mathcal{P}(V)$

```
1 Pick ρ ∈ [d(V)/2, d(V)) with density ln(2)/(x · d(V)) randomly
2 Pick random injection τ : [| V |] → V
3 return TreeMetricApproximation(G, d, ρ, τ)
```

Remark 5.2.1. The algorithm is independent of the edge costs.

Remark 5.2.2. Some authors, like [1], choose ρ with uniform distribution, but in fact the choice of the original paper, $\ln(2)/(x \cdot d(V))$, is better. Mathematically this shows up in equation (5.1), where the radius and the density vanish. Lucidly considering also the divided cut-radii, this results in a continuous probability for every cut-radius.

Theorem 5.2.3 (Randomized tree metric)**.** *Let* $G = (V, H)$ *be a hypergraph and* $d : \binom{V}{2} \to \mathbb{R}_+$ *be a graph metric then Algorithm 4 is a randomized polynomial-time algorithm that produces an induced tree metric* $d^T : \binom{V}{2} \to \mathbb{R}_+$*, which satisfies:*

- $\forall h \in H : d(h) \leq d^T(h)$
- $\forall h \in H : \mathbb{E}[d^T(h)] \leq d(h) \cdot 4 \cdot \ln(2) \cdot (1 + \ln |V|) \in d(h) \cdot \mathcal{O}(\log |V|)$.

Proof. Theorem 5.2.3 was first proved by [11] but only for graphs. In [1], this was generalized to hypergraphs but with a worse constant of $16 \cdot (1 + \ln |V|)$ after bounding the harmonic sum. We will reproof this theorem with the technique of [11] to attain the original constant of $4 \cdot \ln(2) \cdot (1 + \ln |V|)$ for hypergraphs.

Let $h \in H$ be an arbitrary edge. We say that a vertex $v \in V$ settles an edge $h \in H$ in iteration i if the vertex of h that is first assigned in iteration i is assigned to v. We say a vertex v cuts an edge h on level i if h is in $C(S)$, where $S \subseteq V$ are the vertices which could be assigned to v in level i. Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a suitable probability space. Define the following events:

$$X_{iv}(h) := \{v \text{ cuts } h \text{ in iteration i}\} \in \mathcal{F}$$
$$S_{iv}(h) := \{v \text{ settles } h \text{ in iteration i}\} \in \mathcal{F}$$

Note that an edge being cut implies that it was settled, i. e. $\mathrm{X} \subseteq \mathrm{S}$.
Let $\{v_1, \ldots, v_n\} = \mathrm{V}$ with $\tilde{\mathrm{d}}_{\max}(h, v_i) \leq \tilde{\mathrm{d}}_{\max}(h, v_{i+1})$ for all $i \in [|\,\mathrm{V}\,| - 1]$.

$$
\begin{aligned}
\mathbb{E}\left[\mathrm{d}^{\mathrm{T}}(h)\right] &= \mathbb{E}\left[\max_{i \in \mathbb{N}} \mathbb{1}(\exists v \in \mathrm{V} : \mathrm{X}_{iv} \wedge \mathrm{S}_{iv}) \cdot \frac{\rho}{2^{i-2}}\right] \\
&\leq \mathbb{E}\left[\sum_{v \in \mathrm{V}} \sum_{i=0}^{\infty} \mathbb{1}(\mathrm{X}_{iv} \wedge \mathrm{S}_{iv}) \cdot \frac{\rho}{2^{i-2}}\right] \\
&= \mathbb{E}\left[\sum_{v \in \mathrm{V}} \sum_{i=0}^{\infty} \mathbb{P}[\mathrm{S}_{iv} \,|\, \mathrm{X}_{iv}, \rho] \cdot \mathbb{P}[\mathrm{X}_{iv} \,|\rho] \cdot \frac{\rho}{2^{i-2}}\right] \\
&\leq \mathbb{E}\left[\sum_{j=1}^{|\,\mathrm{V}\,|} \frac{1}{j} \sum_{i=0}^{\infty} \mathbb{P}[\mathrm{X}_{iv} \,|\rho] \cdot \frac{\rho}{2^{i-2}}\right] \\
&= \sum_{j=1}^{|\,\mathrm{V}\,|} \frac{1}{j} \sum_{i=0}^{\infty} \int_{\frac{\mathrm{d}(\mathrm{V})}{2}}^{\mathrm{d}(\mathrm{V})} \mathbb{1}\left(\frac{\rho}{2^i} \in \left[\tilde{\mathrm{d}}_{\min}(h, v_j), \tilde{\mathrm{d}}_{\max}(h, v_j)\right)\right) \cdot \frac{\rho}{2^{i-2}} \cdot \frac{\ln(2)}{\rho \cdot \mathrm{d}(\mathrm{V})}\, \mathrm{d}\rho \quad (5.1) \\
&= \sum_{j=1}^{|\,\mathrm{V}\,|} \frac{1}{j} \sum_{i=0}^{\infty} \int_{\frac{\mathrm{d}(\mathrm{V})}{2^{i+1}}}^{\frac{\mathrm{d}(\mathrm{V})}{2^i}} \mathbb{1}\left(x \in \left[\tilde{\mathrm{d}}_{\min}(h, v_j), \tilde{\mathrm{d}}_{\max}(h, v_j)\right)\right) \cdot 2^2 \cdot \frac{\ln(2)}{\mathrm{d}(\mathrm{V})}\, \mathrm{d}x \\
&= \sum_{j=1}^{|\,\mathrm{V}\,|} \frac{1}{j} \int_{0}^{\mathrm{d}(\mathrm{V})} \mathbb{1}\left(x \in \left[\tilde{\mathrm{d}}_{\min}(h, v_j), \tilde{\mathrm{d}}_{\max}(h, v_j)\right)\right) \cdot 2^2 \cdot \frac{\ln(2)}{\mathrm{d}(\mathrm{V})}\, \mathrm{d}x \\
&\leq \sum_{j=1}^{|\,\mathrm{V}\,|} \frac{4 \cdot \ln(2) \cdot \mathrm{d}(h)}{j} \\
&\leq 4 \cdot \ln(2) \cdot \mathrm{d}(h) \cdot (1 + \ln |\,\mathrm{V}\,|) \in \mathcal{O}(\log |\,\mathrm{V}\,|) \cdot \mathrm{d}(h)
\end{aligned}
$$

□

Remark 5.2.4. We can also pick $\rho \in [\mathrm{d}(\mathrm{V})/4, \mathrm{d}(\mathrm{V})/2)$ because estimated distances of the tree only decrease and the claim $\mathrm{d}(h) \leq \mathrm{d}^{\mathrm{T}}(h)$ is still valid.

We have now the guarantee that the stretch of the expected length is logarithmic in $|\,\mathrm{V}\,|$, but we want to bound the sum of all edges:

$$\sum_{h \in \mathrm{H}} \mathrm{c}(h) \cdot \mathrm{d}^{\mathrm{T}}(h) \leq \mathcal{O}(\log |\,\mathrm{V}\,|) \cdot \sum_{h \in \mathrm{H}} \mathrm{c}(h) \cdot \mathrm{d}(h)$$

In the next chapters we present two variants of derandomization of this algorithm, the original one in [11] and the one it [1]. Testing all permutations would give us a factor of $|\,\mathrm{V}\,|!$ to the running time. Therefore, we have to be more careful.

Corollary 5.2.5. *For a given metric* (V, d) *on a hypergraph* $\mathrm{G} = (\mathrm{V}, \mathrm{H})$ *and a cost function* $\mathrm{c} : \mathrm{H} \to \mathbb{R}_{>0}$*, Algorithm 4 is a randomized polynomial-time algorithm that produces a 2-hierarchically closed-well-separated tree metric* $(\mathrm{T}, \mathrm{d}_\mathrm{T})$*, for which the induced tree metric* $\mathrm{d}^\mathrm{T} : \binom{\mathrm{V}}{2} \to \mathbb{R}_+$ *satisfies:*

- $\forall h \in \mathrm{H} : \mathrm{d}(h) \leq \mathrm{d}^\mathrm{T}(h)$ *and*
- $\forall h \in \mathrm{H} : \mathrm{c}(h) \cdot \mathbb{E}\left[\mathrm{d}^\mathrm{T}(h)\right] \leq \mathrm{c}(h) \cdot \mathrm{d}(h) \cdot 4 \cdot \ln(2) \cdot (1 + \ln|\mathrm{V}|) \in \mathcal{O}(\log|\mathrm{V}|) \cdot \mathrm{w}(h)$.

5.3. Pessimistic estimators

The idea of this method was described by Raghavan in 1988, see [25]. Every time the algorithm usually makes a random decision, we calculate the expected solution values of every possible choice. There must be at least one which does not increase the expected solution value. If possible, we take the smallest one.

Applied to Algorithm 4 this means that we take one vertex after another. Each time we base our choice on the conditional expectation, given a partial permutation of the vertices. Ahrens stated in [1] that we can calculate the probability of an edge beeing cut in a certain level only with the information of how many vertices are near enough to cut or settle it. Unfortunately, this claim is wrong because he did not consider the possibility that an edge can be cut by the same vertex in different levels. We will state a counterexample and a correction for the calculation of the probabilities.
Let $\mathrm{G} = (\mathrm{V}, \mathrm{H})$ be a hypergraph and $\mathrm{d} : \binom{\mathrm{V}}{2} \cup \mathrm{H} \to \mathbb{R}_+$ be a graph metric. Let $S_l : \mathrm{H} \to \mathcal{P}(\mathrm{V})$ be all vertices that can settle an edge at iteration l and $C_l : \mathrm{H} \to \mathcal{P}(\mathrm{V})$ be all vertices that can cut an edge at iteration l.

$$C_l(h) := \left\{v \in \mathrm{V} : \tilde{\mathrm{d}}_{\min}(h, v) \leq \rho \cdot 2^{-l} < \tilde{\mathrm{d}}_{\max}(h, v)\right\}$$
$$S_l(h) := \left\{v \in \mathrm{V} : \tilde{\mathrm{d}}_{\min}(h, v) \leq \rho \cdot 2^{-l}\right\}$$

Note that we use different notation compared to [1]. The cutting vertices $C_l(h)$ are also included in the settling vertices S analogously to the definition of an edge being settled or cut in Section 5.2.

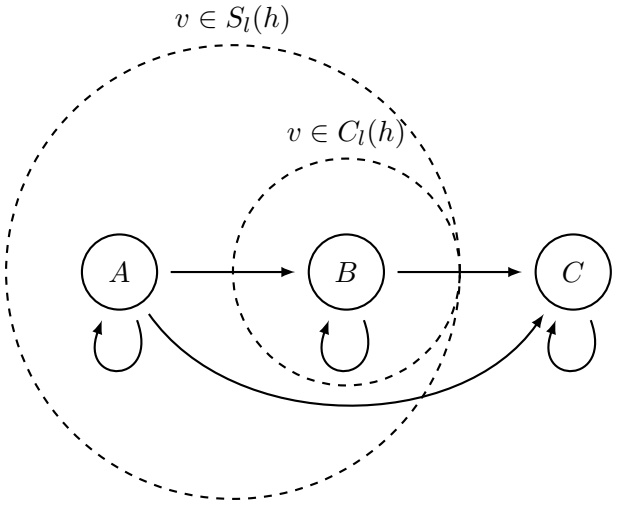

Figure 5.3.: Possible combinations of a vertex v beeing included in the settled or cut set for a hyperedge h. The transition is only possible in one direction for increasing levels.

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a suitable probability space. We define the random variable $\mathrm{L}(h) : \Omega \to \mathbb{N}$ as the level on which a hyperedge $h \in \mathrm{H}$ is cut. Ahrens claimed that the probability of an edge h beeing cut in iteration $l \in \mathbb{N}$ then is:

$$\mathbb{P}[L(h) = l] = \prod_{i \in [l-1]} \frac{|S_i(h)| - |C_l(h)|}{|S_i(h)|} \cdot \frac{|C_l(h)|}{|S_l(h)|}$$

Unfortunately, this is not true as we show in Example 5.3.1.

Example 5.3.1. Consider Fig. 5.4. Let the beginning radius be $\rho = 8/3$. The probability that h is cut on level 1 is $1/4$, on level 2 is $1/2$ and on level 3 is $1/4$. Let us consider the formula of thesis [1] for the second level:

$$\begin{aligned} \prod_{i \in [1]} \frac{|S_i(h)| - |C_i(h)|}{|S_i(h)|} \cdot \frac{|C_2(h)|}{|S_1(h)|} &= \frac{|S_1(h)| - |C_1(h)|}{|S_1(h)|} \cdot \frac{|C_2(h)|}{|S_2(h)|} \\ &= \frac{4-1}{4} \cdot \frac{3}{4} = \frac{9}{16} \neq \frac{1}{2} \end{aligned}$$

The problem in this example is that shifting the edge h to level 2 already implies that it cannot be cut by the leftmost vertex, even if it has the right distance.

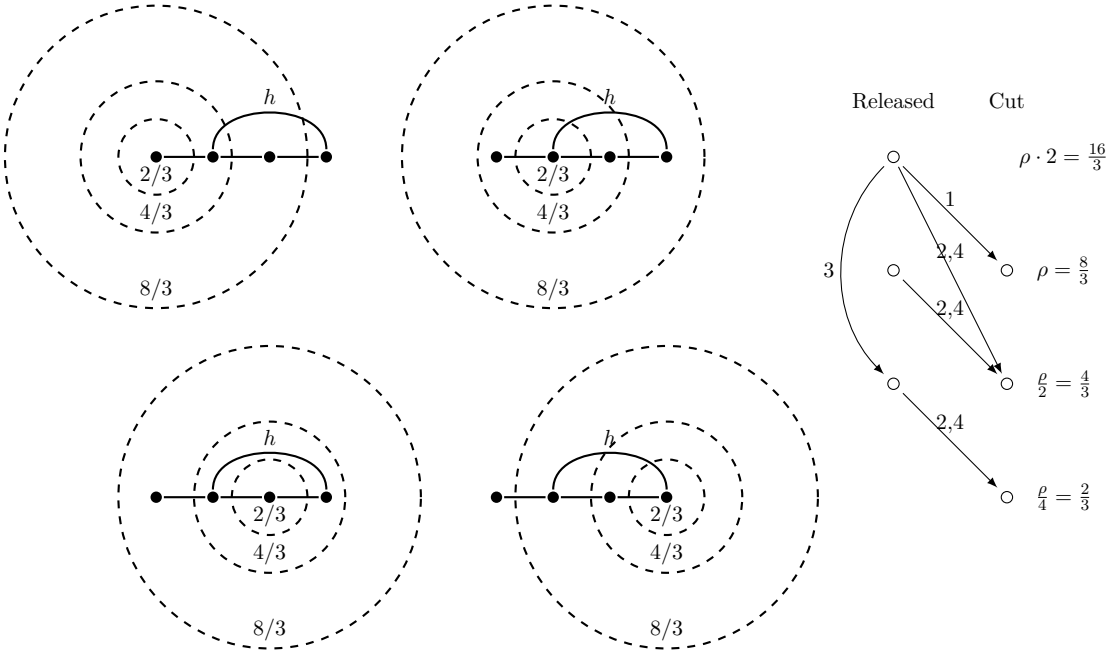

Figure 5.4.: Counterexample to the derandomization of the thesis [1]. All edges have length one, except for h which has length two. At the top left, h is cut in level one, at the top and bottom right, in level two and on the bottom left it is cut in level three.

We can fix this by more detailed information on the state an edge has for a given start radius and partial permutation. We observe that there are only few possible combinations and transitions of the sets in which a vertex is included, see Fig. 5.3. With this information we can make the following definition:

Definition 5.3.2 (Current level)**.** Given a hypergraph $\mathrm{G} = (\mathrm{V}, \mathrm{H})$, a graph metric $\mathrm{d} : \binom{\mathrm{V}}{2} \to \mathbb{R}_{>0}$, a start radius $\rho \in \mathbb{R}_{>0}$ and a partial permutation $\tau : [k] \to \mathrm{V}$ with $k \in [\mathrm{V}]$, we define the current level $\mathcal{L} : \mathrm{H} \to \mathbb{N}$ as

$$\mathcal{L}(h) = \max\{l \in \mathbb{N} \text{ with } \tau([k]) \cap S_l(h) \neq \emptyset\}$$

and the cut status $\mathcal{F} : \mathrm{H} \to \{\mathrm{True}, \mathrm{False}\}$ by

$$\mathcal{F}(h) = \mathbb{1}(\exists i \in [k] : \{l \in \mathbb{N} \text{ with } \tau([i-1]) \cap S_l(h) \neq \emptyset\} \not\supseteq \{l \in \mathbb{N} \text{ with } \tau(i) \in C_l(h)\}).$$

We observe that an edge can only be settled by different vertices if the vertex which first settles the edge does not cut it. In Fig. 5.5 this is shown by the diagonal edge. This leads to the following definition:

Definition 5.3.3 (Released)**.** Given a hypergraph $\mathrm{G} = (\mathrm{V}, \mathrm{H})$ with a graph metric $\mathrm{d} : \binom{\mathrm{V}}{2} \to \mathbb{R}_{+}$, a start radius $r \in \mathbb{R}_{>0}$ and a partial permutation $\tau : [k] \to \mathrm{V}$ with $k \in [\mathrm{V}]$, we say that an edge $h \in \mathrm{H}$ is **released** in level $l \in \mathbb{N}$ if it is not cut by the given points i. e. $\mathcal{F}(h) = \mathrm{False}$ and it is currently settled on level $\mathcal{L}(h) = l$.

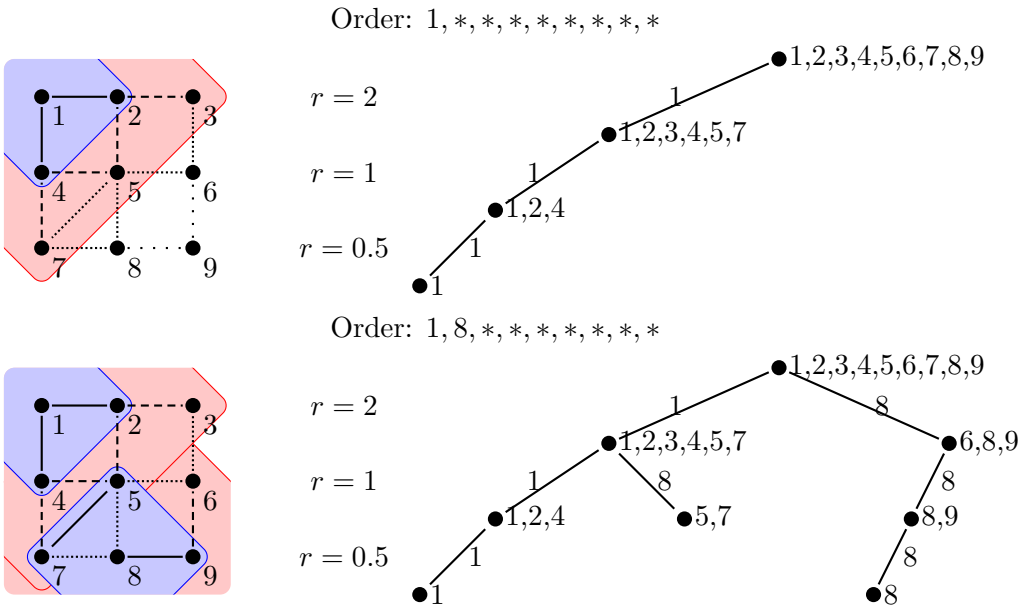

Figure 5.5.: Example of a partial tree metric approximation with $\rho = 2$. The edges of the tree are labeled with the vertex that is chosen from the permutation. The edges are marked loosely dotted if the current level is zero, densely dotted if it is one, dashed if it is two and solid if it is three. The diagonal edge is released in level one in the first figure and in level two in the second figure. All other edges are directly cut, when they are settled.

A crucial observation is that we can calculate the expected length of an edge using only the information on which level the edge is cut or released. We do not need the knowledge about the exact partial permutation. Let $\mathrm{d}^{\mathrm{T}} : \binom{\mathrm{V}}{2} \to \mathbb{R}_+$ be the induced tree metric.

$$\mathbb{E}\left[\mathrm{d}^{\mathrm{T}}(h) | L(h) = l\right] = \rho \cdot 2^{2-l}$$

$$\mathbb{E}\left[\mathrm{d}^{\mathrm{T}}(h) | h \text{ is released in level l}\right] = \frac{\sum_{i=l+1}^{\infty} \rho \cdot 2^{2-i} \cdot \left| C_i(h) \setminus \bigcup_{j=l+1}^{i-1} C_j(h) \right|}{|S_{l+1}(h)|} \tag{5.2}$$

$$+ \frac{\sum_{i=l+1}^{\infty} \mathbb{E}\left[\mathrm{d}^{\mathrm{T}}(h) | h \text{ is released in level i}\right] \cdot |S_i(h) \setminus (C_i(h) \cup S_{i+1}(h))|}{|S_{l+1}(h)|} \tag{5.3}$$

In (5.2) we calculate the costs of a cut in a certain level. We have to make sure that we do not count a cut twice, thus we have to exclude the cuts before. In (5.3) we calculate the probability of an edge being released again at a higher level. In the sense of Fig. 5.3 this is a transition from A directly to C.

With the possibility of calculating the expected length of an edge, we are also able to calculate the expected volume. Let the current level be defined by $\mathcal{L} : \mathrm{H} \to \mathbb{N}$ and the

cut status be defined by $\mathcal{F} : \mathrm{H} \to \{\mathrm{True}, \mathrm{False}\}$ then the estimated cost is given by the following formula:

$$\mathbb{E}\left[\mathrm{d}^{\mathrm{T}} \,|\, \mathcal{F}, \mathcal{L}\right] := \sum_{h \in \mathrm{H}} \mathrm{c}(h) \cdot \begin{cases} \mathbb{E}[\mathrm{d}^{\mathrm{T}}(h) |\, \mathrm{L}(h) = \mathcal{L}(h)], & \text{if } \mathcal{F}(h) \\ \mathbb{E}[\mathrm{d}^{\mathrm{T}}(h) | h \text{ is released in level } \mathcal{L}(h)], & \text{if not } \mathcal{F}(h) \end{cases}$$

Algorithm 5: Tree metric estimators approximation

input : Hypergraph $\mathrm{G} = (\mathrm{V}, \mathrm{H})$,
Graph metric $\mathrm{d} : \binom{\mathrm{V}}{2} \to \mathbb{R}$,
Cost function $\mathrm{c} : \mathrm{H} \to \mathbb{R}_+$
output: Tree $\mathrm{T} = (\mathrm{V_T}, \mathrm{E_T}, \mathrm{r_T})$
Cluster function $\Lambda : \mathrm{V_T} \to \mathcal{P}(\mathrm{V})$

$\mathrm{P} := \left\{ \mathrm{d}(u,v) \cdot 2^{\left\lceil \log_2 \frac{\mathrm{d}(\mathrm{V})}{\mathrm{d}(u,v)} \right\rceil - 1} \text{ for } \{u,v\} \in \binom{\mathrm{V}}{2} \right\} \subset \left[\frac{1}{2}\,\mathrm{d}(\mathrm{V}), \mathrm{d}(\mathrm{V}) \right)$

Init: $\mathcal{F}_h := \mathrm{False} \quad \forall h \in \mathrm{H}$

$\rho := \operatorname{argmin}_{r \in \mathrm{P}} \mathbb{E}[\mathrm{d}^{\mathrm{T}} \,|\, \mathcal{F}, \mathcal{L}]$ with $\mathcal{L}_h = 0$

Init: $\mathcal{L}_h := 0 \quad \forall h \in \mathrm{H}$

for $i = 1$ **to** $|\,\mathrm{V}\,|$ **do**

  $w := \operatorname{argmin}_{v \in V \setminus \tau([i-1])} \mathbb{E}[\mathrm{d}^{\mathrm{T}} \,|\texttt{PlaceNode}(\mathrm{G}, \mathcal{F}, \mathcal{L}, v, r_0)]$

  $\tau(i) := w$

  $(\mathcal{F}, \mathcal{L}) := \texttt{PlaceNode}(\mathrm{G}, \mathcal{F}, \mathcal{L}, w, r_0)$

end

return `TreeMetricApproximation`$(\mathrm{G}, \mathrm{d}, r_0, \tau)$

Procedure `PlaceNode()`

  input : Hypergraph $\mathrm{G} = (\mathrm{V}, \mathrm{H})$
  Edge cut flag $\mathcal{F} : \mathrm{H} \to \{\mathrm{True}, \mathrm{False}\}$
  Edge level $\mathcal{L} : \mathrm{H} \to \mathbb{N}$
  Vertex $v \in \mathrm{V}$
  Start radius $\rho \in \mathbb{R}_+$
  output: Edge cut flag $\mathcal{F} : \mathrm{H} \to \{\mathrm{True}, \mathrm{False}\}$
  Edge level $\mathcal{L} : \mathrm{H} \to \mathbb{N}$

  foreach $h \in \mathrm{H}$ **do**

    $l_{near} := \max\{k \in \mathbb{N} : \tilde{\mathrm{d}}_{\min}(v,h) \le \rho \cdot 2^{-k}\}$

    $l_{far} := \max\{k \in \mathbb{N} : \tilde{\mathrm{d}}_{\max}(v,h) \le \rho \cdot 2^{-k}\}$

    if not $\mathcal{F}_h$ **and** $l_{near} > \mathcal{L}_h$ **then**

      $\mathcal{L}_h := \max\{\mathcal{L}_h, l_{far}\}$

      if $l_{near} \neq l_{far}$ **then**

        $\mathcal{F}_h := \mathrm{True}$

      end

    end

  end

  return $(\mathcal{F}, \mathcal{L})$

Theorem 5.3.4 (Pessimistic estimators tree metric)**.** *Algorithm 5 returns a 2-hierarchically well-separated closed tree metric* $\mathrm{d_T} : \binom{\mathrm{V_T}}{2} \to \mathbb{R}_+$ *by only using deterministic operations, such that the induced tree metric* $\mathrm{d^T} : \binom{\mathrm{V}}{2} \to \mathbb{R}_+$ *fulfills the following inequalities:*

- $\forall h \in \mathrm{H} : \mathrm{d^T}(h) \geq \mathrm{d}(h)$
- $\sum_{h \in \mathrm{H}} \mathrm{c}(h) \cdot \mathrm{d^T}(h) \leq \sum_{h \in \mathrm{H}} \mathrm{c}(h) \cdot \mathrm{d}(h) \cdot 4 \cdot \ln(2) \cdot (1 + \ln |\,\mathrm{V}\,|) \in \mathcal{O}(\log |\,\mathrm{V}\,|) \cdot \sum_{h \in \mathrm{H}} \mathrm{w}(h)$

Proof. The bound is directly clear comparing the result to Corollary 5.2.5. This algorithm is always at least as good as the expectation and thus attains the same bound as Algorithm 3. □

Proposition 5.3.5. *Let the root-radius-set be defined as follows:*

$$\mathrm{P} := \left\{ \mathrm{d}(u,v) \cdot 2^{\lceil \log_2 \frac{\mathrm{d(V)}}{\mathrm{d}(u,v)} \rceil - 1} \text{ with } \{u,v\} \in \binom{\mathrm{V}}{2} \right\} \subset \left[\frac{1}{2}\,\mathrm{d(V)}, \mathrm{d(V)} \right)$$

Then this set contains a radius for which the minimum is attained.

Proof. Let $\rho \in \left[\frac{1}{2}\,\mathrm{d(V)}, \mathrm{d(V)} \right)$ be a start radius, then $\tilde{\rho} = \max\{x : x \in \mathrm{P} \text{ with } x \leq \rho\}$ generates exactly the same cuts, but the root radius is not greater, hence the edge lengths in the tree do not increase. □

Proposition 5.3.6. *Let* $l_{\max} = \log_2 \frac{\rho}{\min_{\{u,v\} \in \binom{\mathrm{V}}{2}} \mathrm{d}(u,v)}$ *be the maximum level which can be reached by Algorithm 5, then we can implement this algorithm in running time* $\mathcal{O}(|\,\mathrm{V}\,|^2 \cdot (|\,\mathrm{V}\,| + l_{\max}) \cdot |\,\mathrm{H}\,|)$.

Proof. First we run Dijkstra's algorithm from every vertex, which has running time $\mathcal{O}(|\,\mathrm{V}\,| \cdot |\,\mathrm{H}\,| \log |\,\mathrm{H}\,|)$. In the worst case, there are $|\,\mathrm{V}\,|^2$ possible values for ρ. Each calculation of the estimated cost has running time $\mathcal{O}((|\,\mathrm{V}\,| + l_{\max}) \cdot |\,\mathrm{H}\,|)$. So choosing the best ρ has already the desired running time. The rest of the algorithm is implementable in running time $\mathcal{O}(|\,\mathrm{V}\,|^2 \cdot |\,\mathrm{H}\,|)$. With pre-calculated expectations the procedure `PlaceNode` has running time $\mathcal{O}(|\,\mathrm{H}\,|)$. We call this procedure $\mathcal{O}(|\,\mathrm{V}\,|)$ times. □

Remark 5.3.7. We can also run the whole algorithm for each element in P, which only can improve the tree metric, but has running time $\mathcal{O}(|\,\mathrm{V}\,|^2 \cdot (|\,\mathrm{V}\,|^2 + l_{\max}) \cdot |\,\mathrm{H}\,|)$

Remark 5.3.8. In Remark 5.2.4 we stated that it is possible to half the start radius in Algorithm 4. This is also possible in Algorithm 5. It could be useful, because otherwise the algorithm will always try to choose the first vertex such that it settles the whole graph in the first iteration, see 8.3.

5.4. Region growing

In [11] the authors first describe the randomized algorithm and then a derandomization, which grows regions around each chosen point. Technically, the stated algorithm is not really a derandomization, because the arguments for the quality guarantee arise from an analytic view. They first proved their approximation bound for graphs with the restriction that the smallest edge length is one and the longest length is $|\,\mathrm{V}\,|$. Furthermore, they distribute additional weights over all edges to get a lower bound for the volume included in the ball of radius one. By using the original idea of the proof, we show that one can get rid of these assumptions without changing anything on the graph and apply the algorithm directly. By this we can expect a much better constant than in the original paper, even though they only state the asymptotic bound. Furthermore, we generalize the proof to hyperedges.

Definition 5.4.1 (Volume)**.** Let $\mathrm{G} = (\mathrm{V}, \mathrm{H})$ be a hypergraph, $\mathrm{d} : \binom{\mathrm{V}}{2} \cup \mathrm{H} \to \mathbb{R}_+$ be a graph metric and $\mathrm{c} : \mathrm{H} \to \mathbb{R}_{>0}$ be a cost function. The volume of the whole graph is defined by

$$\mathrm{W} := \sum_{h \in \mathrm{H}} \mathrm{w}(h) = \sum_{h \in \mathrm{H}} \mathrm{c}(h) \cdot \mathrm{d}(h)$$

Furthermore, for a vertex $t \in \mathrm{V}$ and a radius $r \in \mathbb{R}_+$ we define the volume of the neighborhood:

$$\mathrm{W}(t, r) := \sum_{h \in \mathrm{H}} \mathrm{c}(h) \cdot \begin{cases} 0, & r \in [0, \tilde{\mathrm{d}}_{\min}(t, h)] \\ \frac{\mathrm{d}(h) \cdot (r - \tilde{\mathrm{d}}_{\min}(t,h))}{\tilde{\mathrm{d}}_{\max}(t,h) - \tilde{\mathrm{d}}_{\min}(t,h)}, & r \in (\tilde{\mathrm{d}}_{\min}(t, h), \tilde{\mathrm{d}}_{\max}(t, h)) \\ \mathrm{d}(h), & r \in [\tilde{\mathrm{d}}_{\max}(t, h), \infty) \end{cases}$$

In the following we will use a shorter notation for the cost of a cut:

$$\mathrm{c}(t, r) := \mathrm{c}(\mathrm{C}(\mathrm{B}(t, r))) = \sum_{h \in \mathrm{H}} \mathrm{c}(h) \cdot \mathbb{1}\left(r \in \left[\tilde{\mathrm{d}}_{\min}(t, h), \tilde{\mathrm{d}}_{\max}(t, h)\right)\right)$$

Algorithm 6: Tree metric growing

input : Hypergraph $\mathrm{G} = (\mathrm{V}, \mathrm{H})$
Graph metric $\mathrm{d} : \binom{\mathrm{V}}{2} \to \mathbb{R}$
Cost function $\mathrm{c} : \mathrm{H} \to \mathbb{R}_{>0}$
Start radius $\rho \in [\mathrm{d}(\mathrm{V})/2, \mathrm{d}(\mathrm{V}))$

output: Tree $\mathrm{T} = (\mathrm{V_T}, \mathrm{E_T}, \mathrm{r_T})$
Cluster function $\Lambda : \mathrm{V_T} \to \mathcal{P}(\mathrm{V})$

1 Create root r, set $(\mathrm{V_T}, \mathrm{E_T}, \mathrm{r_T}) := (\{\mathrm{r}\}, \emptyset, r)$ and $\Lambda(\mathrm{r}) := \mathrm{V}$
2 $i = 0$
3 **while** $\exists t \in \mathrm{V_T}$ *with* $\mathrm{d}^1_{\mathrm{T}}(\mathrm{r_T}, t) = i$ *and* $|\Lambda(t)| \geq 1$ **do**
4 **foreach** $t \in \mathrm{V_T}$ *with* $\mathrm{d}^1_{\mathrm{T}}(\mathrm{r_T}, t) = i$ *and* $|\Lambda(t)| \geq 1$ **do**
5 $S := \Lambda(t)$
6 **while** $S \neq \emptyset$ **do**
7 $t := \operatorname{argmax}_{t \in S} \mathrm{W}_{\mathrm{G}(S)}(t, \rho \cdot 2^{-i-1})$
8 $r_i(t) := \operatorname{argmin}_{r \in [\rho \cdot 2^{-i-1}, \rho \cdot 2^{-i}]} \frac{c_{\mathrm{G}(S)}(t,r)}{\mathrm{W}_{\mathrm{G}(S)}(t,r)}$
9 $U := \{v \in S : \mathrm{d}(t, v) < r_i(t)\}$
10 Create new tree node t'
11 $\Lambda(t') := U$
12 $(\mathrm{V_T}, \mathrm{E_T}) := (\mathrm{V_T} \cup \{t'\}, \mathrm{E_T} \cup \{(t, t')\})$
13 $S := S \setminus U$
14 **end**
15 **end**
16 $i = i + 1$
17 **end**
18 **return** (T, Λ)

Lemma 5.4.2. *The minimum in line 8 can be calculated by considering only the following distances:*

$$X := \left[\rho \cdot 2^{-i-1}, \rho \cdot 2^{-i}\right] \cap \left(\tilde{\mathrm{d}}(t, S) \cup \left\{\rho \cdot 2^{-i}\right\}\right)$$

Proof. Suppose the minimum is attained at the radius $r \in [\rho \cdot 2^{-i-1}, \rho \cdot 2^{-i}]$. Then the radius $r' := \min\{y \in X : y > r\}$ does not increase the fraction in line 8, it holds that $\mathrm{W}_{\mathrm{G}(S)}(t, r) \leq \mathrm{W}_{\mathrm{G}(S)}(t, r')$ and $c_{\mathrm{G}(S)}(t, r) = c_{\mathrm{G}(S)}(t, r')$. □

Before we can prove the approximation bound of this algorithm, we have to show some helpful lemmata.

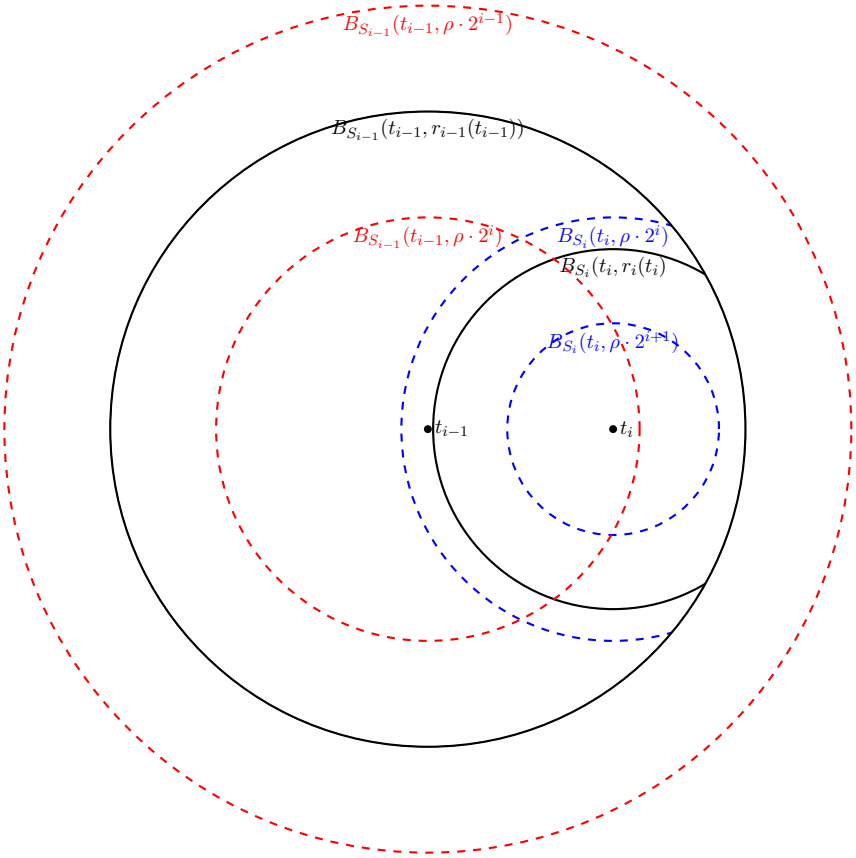

Figure 5.6.: Visualization of two cuts by Algorithm 6

Lemma 5.4.3. *For every weak differentiable function $f : [a, b] \to \mathbb{R}$ and every null set $N \in [a, b]$ there exist a number $r \in [a, b] \setminus N$ such that*

$$\frac{\partial f(r)}{\partial r} \cdot \frac{1}{f(r)} \leq \frac{1}{b-a} \cdot \ln \frac{f(b)}{f(a)}.$$

Proof.

$$\begin{aligned}\int_a^b \ln\frac{f(b)}{f(a)}dr &= \ln\frac{f(b)}{f(a)}\cdot(b-a)\\ &= \Big|_a^b \ln f(r)\cdot(b-a)\\ &= \int_a^b \frac{\partial \ln f(r)}{\partial r}\cdot(b-a)dr\\ &= \int_a^b \frac{1}{f(r)}\cdot\frac{\partial f(r)}{\partial r}\cdot(b-a)dr\end{aligned}$$

Comparing the functions in the integrals gives us the claim. □

Lemma 5.4.4. *For every $t \in \mathrm{V}$ and $i \in \mathbb{N}$ that were chosen in Algorithm 6 there exists an $r \in [\rho\cdot 2^{-i-1}, \rho\cdot 2^{-i})$ such that*

$$\frac{\mathrm{c}(t,r)}{\mathrm{W}(t,r)} \le \frac{1}{\rho}\cdot 2^{i+1}\ln\frac{\mathrm{W}(t,\rho\cdot 2^{-i})}{\mathrm{W}(t,\rho\cdot 2^{-i-1})}.$$

Proof. The function W is not continuous, thus we have to get rid of the jumps in the values.

$$\begin{aligned}f(r) &:= \int_{\rho\cdot 2^{-i-1}}^{r} \begin{cases}\frac{\partial \mathrm{W}(t,r)}{\partial r}, & \mathrm{W}(t,r) \text{ is differentiable}\\ 0, & \text{otherwise}\end{cases} dr\\ &= \lim_{y\nearrow x}\mathrm{W}(t,y) - \sum_{x\in(\rho\cdot 2^{-i-1},r)}\left(\lim_{y\nearrow r}\mathrm{W}(t,y) - \lim_{y\searrow x}\mathrm{W}(t,y)\right)\end{aligned}$$

Note that the following three statements hold:

1) $c(t,r) \le \partial f(r)$
2) $\mathrm{W}(t,r) \le f(r)$
3) $\mathrm{W}(\rho\cdot 2^{-i}) = f(\rho\cdot 2^{-i})$

Now f is continuous and thus there exist an $r \in [\rho\cdot 2^{-i-1}, \rho\cdot 2^{-i})$ such that

$$\frac{\mathrm{c}(t,r)}{\mathrm{W}(t,r)} \underset{1),2)}{\le} \frac{\partial f(r)}{\partial r}\cdot\frac{1}{f(r)} \underset{5.4.3}{\le} \frac{2^{1+i}}{\rho}\ln\frac{f(\rho\cdot 2^{-i})}{f(\rho\cdot 2^{-i-1})} \underset{2),3)}{\le} \frac{2^{1+i}}{\rho}\cdot\ln\frac{\mathrm{W}(t,\rho\cdot 2^{-i})}{\mathrm{W}(t,\rho\cdot 2^{-i-1})}$$

□

We are now ready to prove the following theorem:

Theorem 5.4.5 (Growing tree metric)**.** *Algorithm 6 is a deterministic algorithm, which returns a 2-hierarchically well-separated closed tree metric* $\mathrm{d_T} : \binom{\mathrm{V_T}}{2} \to \mathbb{R}_+$ *such that the induced tree metric* $\mathrm{d^T} : \binom{\mathrm{V}}{2} \to \mathbb{R}_+$ *fulfills the following statements:*

- $\forall h \in \mathrm{H}, \mathrm{d^T}(h) \geq \mathrm{d}(h)$ *and*
- $\sum_{h\in\mathrm{H}} \mathrm{c}(h) \cdot \mathrm{d^T}(h) \leq 8 \cdot \ln(|\,\mathrm{H}\,| \cdot 4) \cdot \sum_{h\in\mathrm{H}} \mathrm{c}(h) \cdot \mathrm{d}(h) \in \mathcal{O}(\log|\,\mathrm{H}\,|) \cdot \sum_{h\in\mathrm{H}} \mathrm{w}(h)$.

Proof. Now we lay the blame of each cut on its enclosing volume, more precise to the edges which are inside or in a cut of a vertex in V. For each edge $h \in \mathrm{H}$ we sum all cuts that are blamed to h. At the end we will sum the blamed value over all edges. Let $k \in \mathbb{N}$ be the depth where h is cut. We get a chain of vertices $t_k, \ldots, t_0 \in \mathrm{V}$, radii $r_k < \ldots < r_0 < \rho$ with $\rho \cdot 2^{-i-1} < r_i \leq \rho \cdot 2^{-i}$ and sets $S_k \subseteq \ldots \subseteq S_0 \subseteq \mathrm{V}$ to which the edge is settled. Because h was not cut in level $k+1$ we can bound the length by $\mathrm{d}(h) \leq \rho \cdot 2^{1-k}$.

$$\begin{aligned}
\sum_{i=k}^{0} \rho \cdot 2^{2-i} \cdot \mathrm{c}_{S_i}(t_i, r_i) \cdot \frac{\mathrm{c}(h)\cdot\mathrm{d}(h)}{\mathrm{W}_{S_i}(t_i, r_i)} &\underset{5.4.4}{\leq} \mathrm{c}(h)\cdot\mathrm{d}(h)\cdot 8 \cdot \sum_{i=k}^{0} \ln \frac{\mathrm{W}_{S_i}(t_i, \rho\cdot 2^{-i})}{\mathrm{W}_{S_i}(t_i, \rho\cdot 2^{-i-1})} \\
&\leq \mathrm{w}(h)\cdot 8 \cdot \sum_{i=k}^{0} \ln \frac{\mathrm{W}_{S_i}(t_i, \rho\cdot 2^{-i})}{\mathrm{W}_{S_{i+1}}(t_{i+1}, \rho\cdot 2^{-i-1})} \\
&= \mathrm{w}(h)\cdot 8 \cdot \ln \frac{\mathrm{W}_{S_0}(t_0, 2^0)}{\mathrm{W}_{S_k}(t_k, \rho\cdot 2^{-k-1})} \\
&\leq \mathrm{w}(h)\cdot 8 \cdot \ln \frac{\mathrm{W}}{\mathrm{c}(h)\cdot\rho\cdot 2^{-k-1}} \qquad (5.4) \\
&\leq \mathrm{w}(h)\cdot 8 \cdot \ln \frac{\mathrm{W}\cdot 4}{\mathrm{c}(h)\cdot\mathrm{d}(h)} \\
&= \mathrm{w}(h)\cdot 8 \cdot \ln \frac{\mathrm{W}\cdot 4}{\mathrm{w}(h)}
\end{aligned}$$

In inequality (5.4), we used that t_k is the node which maximizes $\mathrm{W}_{S_k}(t_k, \rho\cdot 2^{-k-1})$. We could have taken $t_k \in h$ which would have given us $\mathrm{c}(h)\cdot\rho\cdot 2^{-k-1}$.

We can deduce the following bound for weight that is blamed to the whole:

$$\begin{aligned}
\sum_{h\in\mathrm{H}} 8\cdot\mathrm{c}(h)\cdot\mathrm{d^T}(h) &\leq \sum_{h\in\mathrm{H}} 8\cdot\mathrm{w}(h)\cdot\ln\frac{\mathrm{W}\cdot 4}{\mathrm{w}(h)} \\
&= |\,\mathrm{H}\,|\cdot 8\cdot\sum_{h\in\mathrm{H}}\frac{1}{|\,\mathrm{H}\,|}\cdot\mathrm{w}(h)\cdot\ln\frac{\mathrm{W}\cdot 4}{\mathrm{w}(h)} \\
&\underset{5.4.6}{\leq} |\,\mathrm{H}\,|\cdot 8\cdot\frac{\mathrm{W}}{|\,\mathrm{H}\,|}\ln\frac{\mathrm{W}\cdot 4}{\mathrm{W}/|\,\mathrm{H}\,|} \qquad (5.5) \\
&= \mathrm{W}\cdot 8\cdot\ln(|\,\mathrm{H}\,|\cdot 4)
\end{aligned}$$

□

In inequality (5.5) we used the Jensen inequality and the fact that $x \cdot \ln(\mathrm{W} \cdot 4/x)$ is a concave function.

Lemma 5.4.6. *The function $g(x) := x \cdot \ln(\mathrm{W} \cdot 4/x)$ is concave.*

Proof.

$$\frac{\partial g}{\partial x}(x) = \ln \frac{\mathrm{W} \cdot 4}{x} - x \cdot \frac{x}{\mathrm{W} \cdot 4} \cdot \frac{\mathrm{W}}{x^2} = \ln \frac{\mathrm{W} \cdot 4}{x} - 1$$
$$\frac{\partial^2 g}{\partial^2 x}(x) = -\frac{x}{\mathrm{W} \cdot 4} \cdot \frac{\mathrm{W} \cdot 4}{x^2} = -\frac{1}{x} \leq 0$$

□

Remark 5.4.7 (Vertices with same distance). Given a graph $\mathrm{G} = (\mathrm{V}, \mathrm{H})$, a graph metric $\mathrm{d} : \binom{\mathrm{V}}{2} \to \mathbb{R}$ a vertex $t \in \mathrm{V}$ and a set of vertices $S \subseteq \mathrm{V} \setminus \{t\}$ with $\mathrm{d}(t, v) = \mathrm{d}(t, w)$ for all $v, w \in S$, there is no reason why we have to decide that all vertices in S or none of them are mapped to t. We allow the algorithm to take a subset of the vertices to minimize the cut in line 8 of Algorithm 6. The order in which the algorithm parses is fixed, one could think of also optimizing over this order.

Proposition 5.4.8. *Algorithm 6 can be implemented in running time $\mathcal{O}(|\,\mathrm{H}\,| \cdot |\,\mathrm{V}\,| \cdot (\log |\,\mathrm{V}\,| + \log l_{\max}))$ with $l_{\max} = \ln_2 \frac{\rho}{\min_{\{u,v\} \in \binom{\mathrm{V}}{2}} \mathrm{d}(u,v)}$.*

Proof. Depending on the input we first have to calculate the distances between all pairs of vertices. Furthermore we have to order them by their distance to each vertex. This can be done with Dijkstra in a running time of $\mathcal{O}(|\,\mathrm{H}\,| \cdot |\,\mathrm{V}\,| \cdot \log |\,\mathrm{V}\,|)$. Then in each iteration in line 7 we have to calculate the volume of the ball around every vertex to find the maximums. Given that we have ordered the vertices by their distance this needs $\mathcal{O}(|\,\mathrm{H}\,|)$ in the worst case. We have to do this for every vertex in every iteration which would give us $\mathcal{O}(|\,\mathrm{H}\,| \cdot |\,\mathrm{V}\,| \cdot \log l_{\max})$. □

5.5. Order of tree embedding

We want to construct a linear embedding $\tilde{\pi} : \mathrm{V} \to [\mathrm{V}]$ with the calculated tree. Given a Hypergraph $\mathrm{G} = (\mathrm{V}, \mathrm{H})$, a tree $\mathrm{T} = (\mathrm{V_T}, \mathrm{E_T}, \mathrm{r_T})$ and vertex references $\Lambda : \mathrm{V_T} \to \mathcal{P}\,\mathrm{V}$ we recursively iterate over the vertices of the tree, beginning by the root $\mathrm{r_T}$. Every time we pass a leaf $v_\mathrm{T} \in \mathrm{Leaves_T}$ of the tree we assign the next position in the linear embedding $\tilde{\pi}$ to the referenced vertex $\Lambda(v_\mathrm{T})$ to the linear embedding $\tilde{\pi} : \mathrm{V} \to [\mathrm{V}]$.

We have not specified in which order we recursively iterate through the tree. This is not necessary to achieve the optimality guarantee, but in practice we can get much better results by optimizing this. The optimization works as follows. We start from the root and go down step by step. The children of each node are contracted. Thus, we have to solve a linear arrangement problem with terminals.

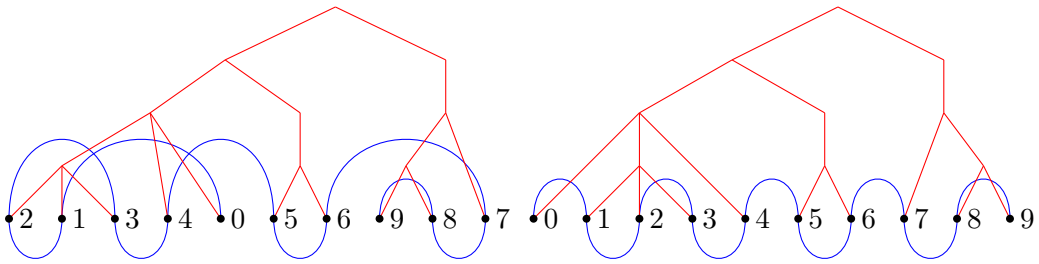

Figure 5.7.: Two linear embeddings of a tree. On the left-hand side with an arbitrary, on the right-hand side with an optimized order.

Lemma 5.5.1. *There exists no upper bound for the maximal fan-out of the trees calculated by one of the three introduced tree metric algorithms.*

Proof. Let $n \in \mathbb{N}$. Consider a graph $\mathrm{G} := \left(\mathrm{V}, \binom{\mathrm{V}}{2}\right)$. Then an optimal solution of the LP is given by the following formula:

$$\mathrm{d}(u,v) = \frac{\mathrm{sb}_k(|\,\mathrm{V}\,|)}{|\,\mathrm{V}\,|-1} \quad \forall \{u,v\} \in \binom{\mathrm{V}}{2}$$

All nodes are separated at the same level and the resulting tree is a star with a fan-out of $|\,\mathrm{V}\,|-1$. □

Remark 5.5.2. This solution is neither unique nor a vertex of the LP. We leave it as an open problem if this lemma also holds for vertices in normal graphs. For hypergraphs we can consider $\mathrm{G} = (\mathrm{V}, \{\mathrm{V}\})$ for which this solution is unique.

So we cannot expect that a full enumeration has polynomial runtime. Thus, we implemented a greedy optimization. Notice that even if we can find optimal solutions for the linear arrangements, this does not necessarily result in the best embedding of this tree, see Fig. 5.8.

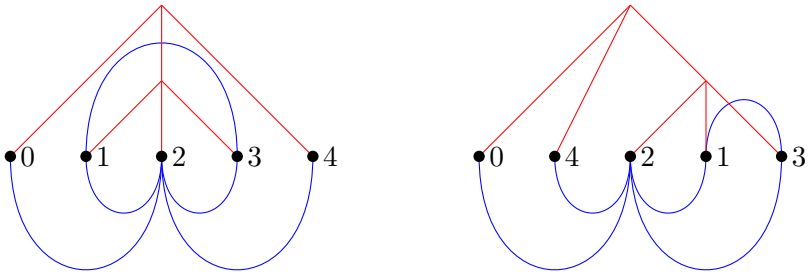

Figure 5.8.: Example of an optimized tree whose embedding is not optimal (left-hand side) and the tree with an optimal embedding (right-hand side).

6. Hilbert space-filling curve

We use a modification of Hilbert space-filling curves to embed a line into a d-dimensional cuboid. These curves were first defined by Hilbert as a special case of peano curves [14]. An interesting property of this curves is that they are local in the sense, that one can calculate good bounds for distances of points based on the distance of their indices (see Lemma 6.3.1). This chapter is built as follows: First we describe an algorithm that generalizes this curves to cuboids. Then we prove a bound for infinite space and then a bound for cuboids.

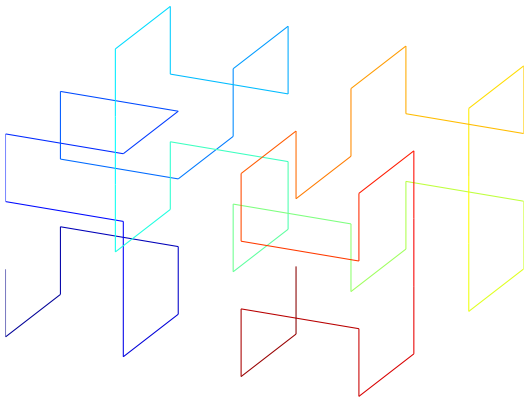

Figure 6.1.: A Hilbert space-filling curve in $[4]^3$ generated by Algorithm 8

6.1. Base Elements

First, we have to create the base elements of the curve. They can be easily expressed by bit-wise operations. In Algorithm 7 the vectors are interpreted as the bit-representation of numbers and logical operations are performed bit-wise. Furthermore, we interpret 0 as false and 1 as true. We introduce some basic operations that we will need to describe the algorithm.

Definition 6.1.1 (Bit-wise representation)**.** Let $n \in \mathbb{N}$ be an arbitrary number and $d \in \mathbb{N}$ the dimension, then we define the **bit-wise representation** $x \in \{0,1\}^d$ of n in dimension d by

$$x_i = \mathbb{1}\left(x > 2^{i-1} \mod 2^i\right)$$

Remark 6.1.2. For a bit-wise representation $x \in \{0,1\}^d$ we can calculate the represented number $n \in [2^d]$ by

$$n = \sum_{i \in [d]} x_i \cdot 2^{i-1}$$

Definition 6.1.3 (Bit-wise operations)**.** Let $x, y \in \{0,1\}^d$ be the representation of two binary numbers. Furthermore, let $i \in [d]$ be an index, then we define the following **bit-wise operations**:

- $(x \oplus y)_i = \mathbb{1}(x_i + y_i = 1)$
- $(\neg x)_i = \mathbb{1}(1 - x_i)$
- $(x \wedge y)_i = \mathbb{1}(x_i = 1 \text{ and } y_i = 1)$

Algorithm 7: Multidimensional Hilbert base-curve

input : Dimension $d \in \mathbb{N}$
output: Hilbert base-curve $\mathrm{b} : 2^d \to \{0,1\}^d$

```
b(1) := (0, ..., 0)_2   (d times)
for i = 1 to 2^d − 1 do
    Let x be the bit-wise representation of i and x_ of i − 1
    b(i + 1) := b(i) ⊕ (x ∧ (¬(x_)))// a bit of b is flipped exactly when it
        changed from 0 to 1 in x
end
return b
```

Remark 6.1.4. The Hilbert base-curves have the following properties:

- In coordinate $[d-1]$ they begin and end at 0
- In coordinate d they begin at 0 and end at 1
- The l_1 distance between two points with index $i, j \in [2^d]$ is smaller than $\min(|i-j|, d)$

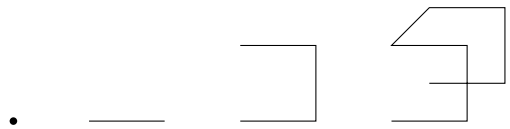

Figure 6.2.: Hilbert-base curves up to dimension 3

6.2. Divide and conquer algorithm for arbitrary cuboids

Usually we would recursively divide a cube into 2^d sub-cubes. To get cuboids we weaken this algorithm. In the first iteration we only subdivide in the longest directions. As we go on we allow two subdivide the cuboids also in shorter directions. Until now all sizes are powers of two. We use the last iteration to subdivide only some hyperplanes and get the desired sizes.

Algorithm 8: Multidimensional Hilbert Curve

input : Cuboid boundary $\mathrm{b} \in \mathbb{N}^d$

output: Curve $\mathrm{q} \in \mathcal{S}\left(\bigtimes_{i=1}^{d} [\mathrm{b}_i]\right)$

$p_i = \lfloor \log_2 \mathrm{b}_i \rfloor$
$\tilde{h}_i := 2^{p_i}$
$\mathrm{q}_1 \in \mathcal{S}\left([1]^d\right)$
$\mathrm{q}_1(1) := ((1, \ldots, 1))$
for $k = 1$ **to** $1 + \max_{i \in [d]} p_i$ **do**
  $E_{ij} := \mathbb{1}(p_j > k \vee (k = \max_{l \in [d]} p_l \wedge j < \tilde{\mathrm{b}_i} - \mathrm{b}_i))$ `// direction` $i \in [d]$ `expanded at hyperplane` j
  $\rho(x)_i := x_i + \sum_{j \in [x_i]} E_{ji}$; `// Projection of point` x
  $\mathrm{q}_{k+1} \in \mathcal{S}\left(\bigtimes_{i=1}^{d} [\min(2^{\max(k-p_i,0)}, \mathrm{b}_i)]\right)$
  $n := 0$
  for $i = 1$ **to** $|\,\mathrm{q}_k\,|$ **do**
    $d' := \sum_{l \in [d]} E_{ip_i}$
    $\mathrm{b} :=$ `AlgorithmMultidimensionalHilbertBaseCurve`(d')
    Let $T : \{0,1\}^d \to \{0,1\}^d$ be an appropriate affine transformation
    for $j = 1$ **to** $|\,\mathrm{b}\,|$ **do**
      $\mathrm{q}_{k+1}(n) := \rho(\mathrm{q}_k(i)) + T(\mathrm{b}_j)$
      $n := n + 1$
    end
  end
end

Result: $\mathrm{q}_{1+\max_{i \in [d]} p_i}$

Remark 6.2.1. Choosing the right affine transformation T is somehow a bit tricky. We always want to start next to the current last point of q_{k+1}. We want to end at a point next to the next cuboid, see Fig. 6.3. The first claim fixes all coordinates, the second only one. So we first look if the coordinate which is fixed in the beginning and in the end is fixed to the same value. If this is the case, we use the flipped coordinate of the base

curve to achieve this, if not we swap the flipped coordinate to another dimension. After this we mirror all not matching coordinates, such that the beginning of the sub-curve is at the right point. In the case that our base curve only has dimension one we are no4t able to swap the flipped coordinate to another dimension, so we would get a jump of the curve in this case, see Fig. 6.4. This is only possible for cuboids with boundaries that are not powers of 2.

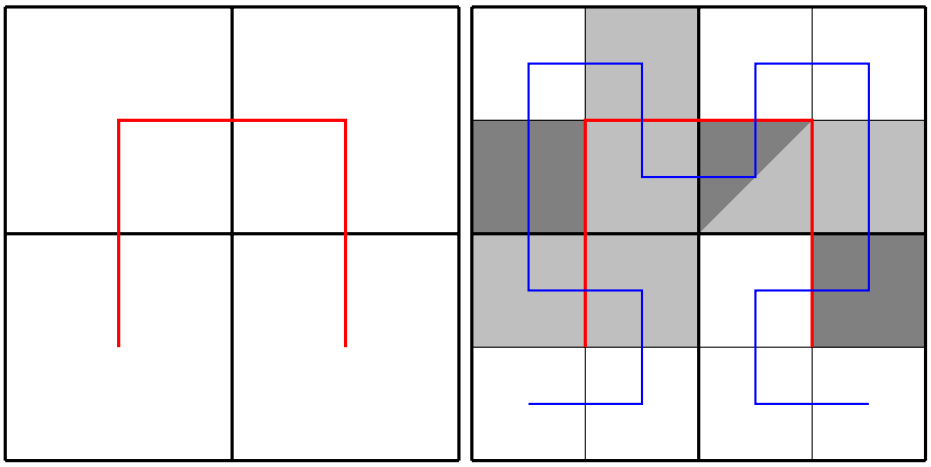

Figure 6.3.: Example of fixed positions of a Hilbert space-filling curve in a 4×4 grid. The fixed start-points are marked in gray, the end-points in light gray.

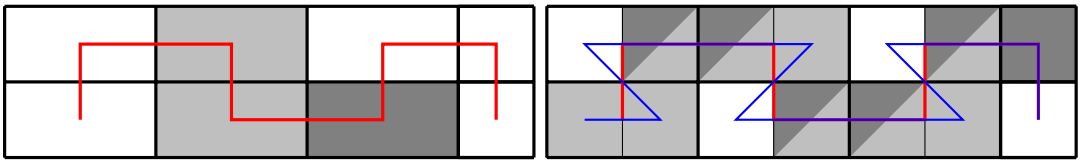

Figure 6.4.: Example of diagonal lines in a 7×2 grid. The fixed start-points are marked in gray, the end-points in light gray.

Lemma 6.2.2. *Algorithm 8 has running time* $O\left(\prod_{i\in[d]} \mathrm{b}_i\right)$

Proof. The number of points of the curve at least doubles in each iteration except for the last iteration. We can bound the next to the last and all previous iterations by the last one. So we get three times the running time of the last iteration. This gives us the desired bound. □

Definition 6.2.3 (Clustering)**.** Let S be an arbitrary finite set. We define a **clustering** as a sequence of sets $G_1 \subseteq \ldots \subseteq G_n \subset \mathcal{P}(S)$ for which

- $G_1 = \left\{ \bigtimes_{i=1}^{d} [\mathrm{b}_i] \right\}$,
- G_n contains only singletons, ie $\forall s \in G_n : |s| = 1$,
- $\forall i \in [n]\ \forall s, t \in G_i : s \cap t = \emptyset$ and
- $\forall i \in [n-1]\ \forall t \in G_i\ \exists S \subseteq G_{i+1}$ such that $t = \bigcup_{s \in S} s$.

Remark 6.2.4. We can define a clustering $G_1 \subseteq \ldots \subseteq G_n$ with $G_1 = [\mathrm{b}_1] \times \ldots \times [\mathrm{b}_d]$ for Algorithm 8, where G_i are the cuboids, which arise from iteration i in the last iteration.

Lemma 6.2.5. *Let G_k be the grid of iteration k. Then for each $s \in G_k$ we have that for all $i \in [d]$:*

$$\max_{x,y \in s} |x_i - y_i| \leq 2 \cdot \min \left\{ \mathrm{b}_i, \max_{j \in [d]} \max_{x,y \in s} |x_j - y_j| \right\}. \tag{6.1}$$

Moreover, if all cuboid boundaries b_i are powers of two we have

$$\max_{x,y \in s} |x_i - y_i| = \min \left\{ \mathrm{b}_i, \max_{j \in [d]} \max_{x,y \in s} |x_j - y_j| \right\}. \tag{6.2}$$

Proof. We will analyze the cuboids, created by the algorithm beginning by the last iteration. If all boundaries are powers of two the last iteration does nothing and all elements in $G_{1+\max_{i \in [d]} p_i}$ are only one point. Otherwise, we can have different cuboids with either one or two points in each direction. From this point on, each iteration we merge two cuboids in each direction as long as we have not reached the corresponding boundary. Thus, in the case of all boundaries being a power of two, the number of points in all directions are the same, except for the directions which the boundary is already reached. This shows equation (6.2). If this is not the case, the ratio of the maximum number of points in a direction versus the minimum number of points in another direction, in which we have not reached the boundary yet, can be bounded by two. This shows Eq. (6.1). □

6.3. Bound for infinite space

Lemma 6.3.1. *Let $d, k \in \mathbb{N}$ be two natural numbers. Then there exists a bijection $\mathrm{q} : \left[2^{k \cdot d}\right] \to \left[2^k\right]^d$ such that $||\,\mathrm{q}(i) - \mathrm{q}(j)||_1 \leq \frac{2 \cdot d + 2}{\sqrt[d]{d}} \cdot \sqrt[d]{|i - j|}$ for all $i, j \in \left[2^{k \cdot d}\right]$. Moreover, such a mapping can be computed in polynomial time.*

Proof. Let $d \in \mathbb{N}$ be the dimension. Furthermore, let $i, j \in [2^{k \cdot d}]$ be two indices. Then there exists a $t \in \{2^n, n \in \mathbb{N}\}$ (the level) and an $n \in [2^d - 1]$ with

$$t^d \cdot n < |i - j| \leq (n + 1) \cdot t^d$$

Then we can bound the distance by the following (see Fig. 6.5):

$$\begin{aligned} ||\,\mathrm{q}(i) - \mathrm{q}(j)||_1 &\leq (d + \min(n + 1, d + 2)) \cdot t \\ &< (d + \min(n + 1, d + 2)) \cdot \sqrt[d]{\frac{|i - j|}{n}} \\ &= \sqrt[d]{\frac{(d + \min(n + 1, d + 2))^d}{n} \cdot |i - j|} \\ &\leq \sqrt[d]{\frac{(2 \cdot d + 2)^d}{d} \cdot |i - j|} \\ &= \frac{2 \cdot d + 2}{\sqrt[d]{d}} \cdot \sqrt[d]{|i - j|} \end{aligned}$$

□

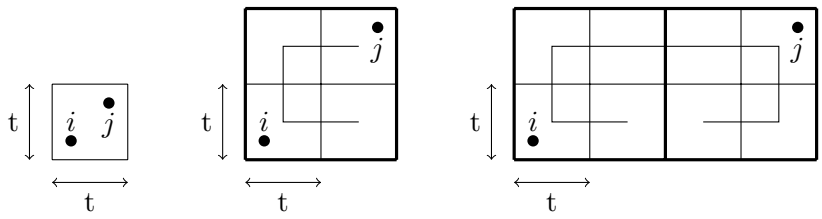

Figure 6.5.: Visualization of the l_1-distance for different values of n in Lemma 6.3.1. The worst case is that i and j lie in opposite corners of the sub-cubes, which gives us a distance of $d \cdot t$ (left). For $n \in [d-1]$ it is possible that there are n sub-cubes between the sub-cubes of i and j, so we have to add $(n+1) \cdot t$ (middle). With increasing n the possible distance of the sub-cubes also increases until we reach $n = d+1$ (right). From this point forward we can bound the additional distance by $(d+2) \cdot t$, because $|i - j| \leq (n + 1) \cdot t^d$ and thus there cannot be a cube in between.

Remark 6.3.2. This bound is not tight. For the tight bound $3 \cdot \sqrt{|i - j|} - 2$ in $\mathbb{N}^2$ and a near tight bound $4.74458 \sqrt[3]{|i - j|}$ in $\mathbb{N}^3$ we refer to [22]. Furthermore, in [26] Rotter and Vygen showed the bound $4 \cdot (d+1) \sqrt[d]{|i - j|}$ for cubes by deleting rows and columns. We found a small error in the proof and correcting this error leads to $4 \cdot (d+1) \cdot \sqrt[d]{2 \cdot |i - j|}$. This result also holds for the algorithm in this thesis. Nevertheless there is a small difference: While they delete the rows and columns we just do not insert them.

6.3.1. Blockages

Lemma 6.3.3. *Let $d, k \in \mathbb{N}$ be two natural numbers and $\mathcal{B} \subseteq \left[2^k\right]^d$ be a set of blockages. Then there exists a bijection $\mathrm{q} : \left[2^{k \cdot d} - |\,\mathcal{B}\,|\right] \to \left[2^k\right]^d \setminus \mathcal{B}$ such that*

$$||\,\mathrm{q}(i) - \mathrm{q}(j)||_1 \leq \frac{2 \cdot d + 2}{\sqrt[d]{d}} \cdot \sqrt[d]{|i-j| + |\,\mathcal{B}\,|} \quad \forall i, j \in \left[2^{k \cdot d} - |\,\mathcal{B}\,|\right]$$

Furthermore, such a mapping can be computed in polynomial time.

Proof. Let $\mathrm{q} : \left[2^{k \cdot d}\right] \to \left[2^k\right]^d$ be defined as in Lemma 6.3.1. We skip blockages in the embedding, see Fig. 6.6, by defining a new indexing function $f : \left[2^{k \cdot d} - |\,\mathcal{B}\,|\right] \to \left[2^{k \cdot d}\right]$.

$$f(i) := \min \left\{ j \in \left[2^{k \cdot d}\right], |\,\mathrm{q}([j]) \setminus \mathcal{B}\,| = i \right\}.$$

The new bijection can then be defined as $\tilde{\mathrm{q}} : \left[2^{k \cdot d} - |\,\mathcal{B}\,|\right] \to \left[2^k\right]^d \setminus \mathcal{B}, \tilde{\mathrm{q}}(i) := \mathrm{q} \circ f$. Calculating the distance leads us to

$$\begin{aligned} ||\tilde{\mathrm{q}}(i) - \tilde{\mathrm{q}}(j)||_1 &\leq ||\,\mathrm{q} \circ f(i) - \mathrm{q} \circ f(j)||_1 \\ &\leq \frac{2 \cdot d + 2}{\sqrt[d]{d}} \cdot \sqrt[d]{|f(i) - f(j)|} \\ &\leq \frac{2 \cdot d + 2}{\sqrt[d]{d}} \cdot \sqrt[d]{|i-j| + |\,\mathcal{B}\,|} \end{aligned}$$

□

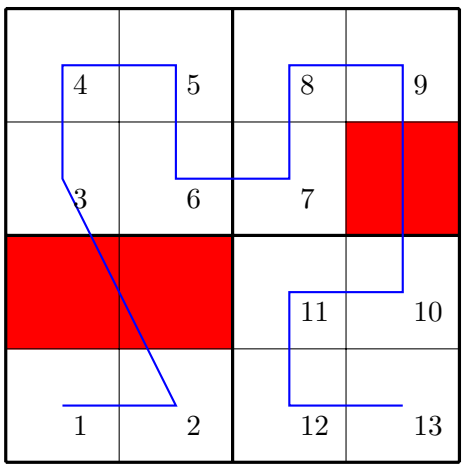

Figure 6.6.: Example of an embedding with tree blockages and a new labeling.

6.4. Bound for cuboids

Lemma 6.4.1. *Let* $\mathrm{b} \in \mathbb{N}^d$ *be the boundary,* $\mathrm{Q} := [\mathrm{b}_1] \times \ldots \times [\mathrm{b}_d]$ *and* $n \in [\mathrm{Q}]$*. Let* O *be an optimal embedded set of size* n *in the maximum norm. Furthermore, let* $\mathrm{q} : [|\,\mathrm{Q}\,|] \to \mathrm{Q}$ *be a Hilbert space-filling curve created by Algorithm 8. Then for every* $i, j \in [|\,\mathrm{Q}\,|]$ *with* $|i - j| \leq n$ *we have the following bound:*

$$||\,\mathrm{q}(i) - \mathrm{q}(j)||_\infty \leq 8 \cdot \max_{x,y \in O} ||x - y||_\infty$$

Furthermore, if all b *are powers of two we can decrease the constant:*

$$||\,\mathrm{q}(i) - \mathrm{q}(j)||_\infty \leq 4 \cdot \max_{x,y \in O} ||x - y||_\infty$$

Proof. Let G_k be the grid of iteration k. Let $k \in \left[\max_{i \in [d]} p_i\right]$ be the smallest number such that there exists $\mathrm{Q}_-, \mathrm{Q}_+ \in G_k$ with

- $\exists k \in \{i, \ldots, j\} : \mathrm{q}(\{i, \ldots, k\}) \subseteq \mathrm{Q}_-$ and $\mathrm{q}(\{k+1, \ldots, n\}) \subseteq \mathrm{Q}_+$
- $\mathrm{Q}_- \cap \mathrm{Q}_+ = \emptyset$

Lets have a closer look on the cuboids Q_- and Q_+ to see why equation 6.3 holds. In each iteration the dimensions of the cuboids in the grid can be doubled. Lets assume we take k such that all cuboids in the grid have at least the size of O, and hence at least as many elements as O. We now can follow that each length of the cuboids in the grid is smaller or equal to four times the cuboid lengths of O. We have a factor of two because we may have to take the doubled size. The disturbance of Lemma 6.2.5 causes another factor of two.

$$\begin{aligned} ||\,\mathrm{q}(i) - \mathrm{q}(j)||_\infty &\leq \max_{x,y \in \mathrm{Q}_- \cup \mathrm{Q}_+} ||x - y||_\infty \\ &\leq 2 \cdot \max \left\{ \max_{x,y \in \mathrm{Q}_-} ||x - y||_\infty, \max_{x,y \in \mathrm{Q}_+} ||x - y||_\infty \right\} \\ &\underset{6.2.5}{\leq} 8 \cdot \max_{x,y \in O} ||x - y||_\infty \end{aligned} \tag{6.3}$$

□

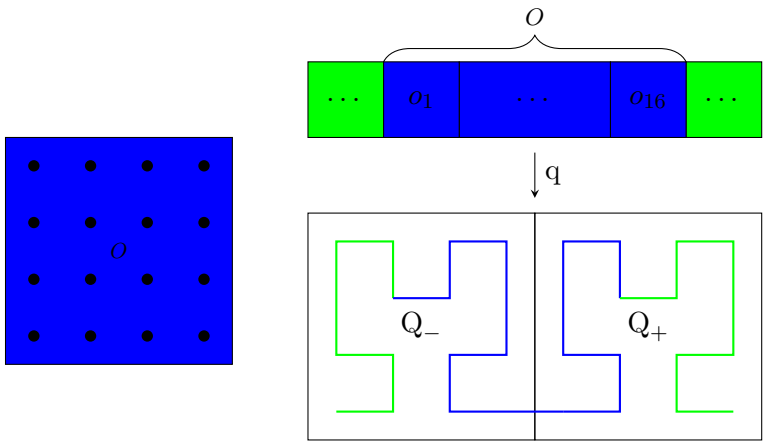

Figure 6.7.: Sketch of the distances of the Hilbert vs the optimal embedding. On the left there are 16 vertices optimally embedded in the l_∞ norm which results in a block of size 4×4. On the right we see the Hilbert embedding of this 16 vertices. In this example the maximum l_∞ distance between points in the optimal embedding is 4, the maximum l_∞ distance in the Hilbert embedding is 5.

7. Proof of main theorem

Theorem 7.1.1 (Rotter, Vygen (2013))**.** *Algorithm 1 together with Algorithm 2, 5 and 8 is a deterministic $\mathcal{O}(\log|\mathrm{V}|)$-approximation algorithm for the unbounded d-*DIMAP*.*

Proof. Let $\mathrm{G} = (\mathrm{V}, \mathrm{H})$ be a graph and $\mathrm{d} : \binom{\mathrm{V}}{2} \to \mathbb{R}_+$ be the optimum solution to the spreading LP. Let $\mathrm{T} = (\mathrm{V_T}, \mathrm{E_T})$ be the tree and $\Lambda : \mathrm{V_T} \to \mathcal{P}(\mathrm{V})$ be the references to the vertices calculated by Algorithm 5. Let $u, v \in \mathrm{V}$ be two vertices. Let $t \in \mathrm{V_T}$ be the unique tree node with $u, v \in \Lambda(t)$ and $|\Lambda(t)|$ minimal. We assume that $n := |\Lambda(t)| \geq 5$. We have $\sum_{x\in\Lambda(t)} \mathrm{d}(u,x) \geq \mathrm{sb}(n)$ where $\mathrm{sb} := \mathrm{sb}^u$ in this proof. Therefore, we can deduce the following inequality:

$$\exists w \in \Lambda(t) : \mathrm{d}(u,w) \geq \frac{\mathrm{sb}(n)}{n-1} \tag{7.1}$$

This inequality holds independently of which spreading bound we choose. Now we can bound the distances step by step:

$$\begin{aligned}
||\,\mathrm{q}(\tilde{\pi}(u)) - \mathrm{q}(\tilde{\pi}(v))||_1 \cdot \tfrac{\sqrt[d]{d}}{2\cdot d+2} &\underset{6.3.1}{\leq} \sqrt[d]{|\,\tilde{\pi}(u) - \tilde{\pi}(v)|} && (7.2)\\
&\leq \sqrt[d]{n-1} && (7.3)\\
&= (n-1)^{1+1/d}/(n-1)\\
&\underset{4.2.1}{=} 4\cdot \mathrm{sb}^u(n)/(n-1)\\
&\underset{(7.1)}{\leq} 4\cdot \mathrm{d}(u,w)\\
&\underset{5.3.4}{\leq} 4\cdot \mathrm{d^T}(u,w)\\
&\leq 4\cdot \mathrm{d^T}(u,v)
\end{aligned}$$

Summing over all edges leads us to

$$\begin{aligned}
\sum_{h\in\mathrm{H}} \mathrm{c}(h)\cdot \mathrm{BBOX}_1(\mathrm{q}(\tilde{\pi}(h))) &\leq \sum_{h\in\mathrm{H}} \tfrac{2\cdot d+2}{\sqrt[d]{d}}\cdot 4\cdot \mathrm{c}(h)\cdot \mathrm{d^T}(h)\\
&\underset{5.3.4}{\leq} \sum_{h\in\mathrm{H}} \tfrac{2\cdot d+2}{\sqrt[d]{d}}\cdot 16\cdot \ln(2)\cdot \mathrm{c}(h)\cdot \mathrm{d}(h)\cdot(1+\ln|\,\mathrm{V}|)\\
&\leq \mathrm{OPT}\cdot\mathcal{O}(\log|\,\mathrm{V}|)
\end{aligned}$$

□

Remark 7.1.2. In [26] Vygen and Rotter proved this result for the bounded d-DIMAP with a slightly worse constant and a different construction of the Hilbert space-filling curve. In contrast to this thesis they delete rows and columns which they want to get rid of.

Theorem 7.1.3. *Let $\mathcal{B} \subseteq \mathrm{Q}$ be a set of blockages then Algorithm 1 together with Algorithm 2, 5 and 8 is a deterministic $\mathcal{O}(\sqrt[d]{|\mathcal{B}|} \cdot \log|\mathrm{V}|)$-approximation algorithm for the cube-bounded d-DIMAP.*

Proof. In the case that we have a set of blockages $\mathcal{B} \subseteq \mathrm{Q}$ we would get $\sqrt[d]{n-1+|\mathcal{B}|}$ in equation (7.3) due to Lemma 6.3.3 which can be bounded by $\sqrt[d]{n-1} \cdot \sqrt[d]{|\mathcal{B}|+1}$. Now we continue with the proof like in the proof of Theorem 7.1.1. In line (7.2) we use the slightly worse constant, which was shown for cubes in [26]. Then, for the final bound, we get an asymptotic factor of $\mathcal{O}(\sqrt[d]{|\mathcal{B}|} \cdot \log|\mathrm{V}|)$ to the optimum solution. □

With this theorem we have improved the result of [12] for cubes of arbitrary dimension.

Theorem 7.1.4 (Main theorem)**.** *Algorithm 1 together with Algorithm 2, 5 and 8 is a deterministic $\mathcal{O}(\log|\mathrm{V}|)$-approximation algorithm for cuboid bounded d-DIMAP.*

Proof. Let $\mathrm{G} = (\mathrm{V}, \mathrm{H})$ be a graph and $\mathrm{d} : \binom{\mathrm{V}}{2} \to \mathbb{R}_+$ be the optimum solution to the spreading LP. Let $\mathrm{T} = (\mathrm{V_T}, \mathrm{E_T})$ be the tree and $\Lambda : \mathrm{V_T} \to \mathcal{P}(\mathrm{V})$ be the references to the vertices calculated by Algorithm 5.
Let $u, v \in \mathrm{V}$ be two vertices. Let $t \in \mathrm{V_T}$ be the unique tree node with $u, v \in \Lambda(t)$ and $n := |\Lambda(t)|$ minimal. Let $O \subseteq [\mathrm{b}_1] \times \ldots \times [\mathrm{b}_d]$ be an optimal embedded set of size $|O| = n$ to the maximum norm. Then we get the following inequalities:

$$
\begin{aligned}
||\,\mathrm{q}(\tilde{\pi}(u)) - \mathrm{q}(\tilde{\pi}(v))||_1 &\leq d \cdot ||\,\mathrm{q}(\tilde{\pi}(u)) - \mathrm{q}(\tilde{\pi}(v))||_\infty && (7.4)\\
&\underset{6.4.1}{\leq} d \cdot 8 \cdot \max_{o,p \in O} ||o-p||_\infty \\
&\leq d \cdot 16 \cdot \max_{o \in O} ||o - \lfloor \mathrm{b}/2 \rfloor||_\infty \\
&\underset{4.2.8}{\leq} d \cdot 16 \cdot \left(2 \cdot \frac{\mathrm{sb}_\infty(n)}{n} + 1\right) \\
&\leq d \cdot 16 \cdot \left(2 \cdot \frac{\mathrm{sb}_1(n)}{n} + 1\right) && (7.5)\\
&\underset{(7.1)}{\leq} d \cdot 48 \cdot \mathrm{d}(u,w) \\
&\underset{5.3.4}{\leq} d \cdot 48 \cdot \mathrm{d^T}(u,w) \\
&\leq d \cdot 48 \cdot \mathrm{d^T}(u,v)
\end{aligned}
$$

Summing over all edges leads to

$$\begin{aligned}\sum_{h\in\mathrm{H}} \mathrm{c}(h)\cdot \mathrm{BBOX}_1(\mathrm{q}(\tilde{\pi}(h))) &\leq \sum_{h\in\mathrm{H}} d\cdot 48\cdot \mathrm{c}(h)\cdot \mathrm{d}^{\mathrm{T}}(h)\\ &\underset{5.3.4}{\leq} \sum_{h\in\mathrm{H}} d\cdot 48\cdot 4\cdot \ln(2)\cdot \mathrm{c}(h)\cdot \mathrm{d}(h)\cdot(1+\ln|\,\mathrm{V}\,|)\\ &\leq \mathrm{OPT}\cdot\mathcal{O}(\log|\,\mathrm{V}\,|)\end{aligned}$$

□

Proposition 7.1.5. *As we pointed out in Remark 3.1.7 we can also consider this problem with other Minkowski metrics. In the case of the maximum norm we can drop the constant d from the approximation bound.*

Proof. We only state the differences to the proof of Theorem 7.1.4. In line (7.4) we do not need to swap to the maximum norm, so we do not have to insert the factor d. In line (7.5) we are not switching back to the l_1-norm. In the spreading LP we have to use the spreading bound for the maximum-norm sb_∞. □

8. Tests

In this chapter we provide some test results of the program we wrote. We first analyze how the runtime of solving the spreading LP behaves for combinations of different graphs and parameters. Then we compare the different algorithms which construct the tree metrics.

8.1. Instances

Name	Explanation	Nodes	Edges	H. edges	Domain	OPT
Gr16	Gridgraph 16×16	256	480	0	16×16	480
HGr12	Hypergridgraph 12×12	144	0	121	12×12	121
Suzanne	Suzanne from blender	507	0	500	$8 \times 8 \times 8$	
Rnd	Random graph	60	120	0	16×16	
Peko	Constructed VLSI	256	173	130	16×16	≤ 552
C1y	VLSI	828	1749	0	16×16	
hsa05221	Homo sapiens genom	32	39	0	32×32	46

Table 8.1.: Testbed

The construction of the instances Gr16, HGr12 is described in Appendix A.1.4. Suzanne is a standard 3d-model of the modeling software Blender and was created by Willem-Paul van Overbruggen [4]. This instance is interesting, because it is very local, not regular and contains only hyperedges. Peko was taken from paper [6]. We cut a rectangle with bounds of $[16] \times [16]$ out of the first instance of first test-suite. The abbreviation stands for placement example with known optimal wire-length. Instance C1y is a VLSI instance from paper [24]. This instance only has conventional edges and is rather sparse, except for one vertex which has a very high degree. In the original paper it was considered as a linear arrangement problem. Instance hsa05221 is an instance created by a human genom which was solved optimally in [23].

8.2. Runtime

For most graphs the linear program is the computationally most intensive part of the algorithm. We tested the following settings and inverted one parameter for each

run.

LP-type	normal	sparse
pre-calculate lower bounds	true	true
post-optimize	true	not available
pre-calculate upper bounds	false	false
pre-calculate triagle inequalities	false	not available

Table 8.2.: Standard settings of the lp solvers

We included support for two LP-solvers: Cplex® [15], which is a productive tool issued by IBM®, and Qsopt [2], which was written by Applegate, Cook, Dash and Mevenkamp and is free to use for research purpose. The tests presented in this thesis were done with Qsopt on an Intel® Core™ i5-7500 CPU at 3.40GHz with 16 GB ram. We used openmp [7] to parallelize the Floyd-Warshall algorithm, Dijkstra from many roots and the pessimistic tree approximation algorithm.

In Section 8.2 we can see that the optimizations indeed have a positive effect on the runtime or even make some problems feasible. Furthermore, there is no clear winner between the spreading LP and the sparse spreading LP.

LP-type	Inverted	Gr16	HGr12	Suzanne	Rnd	Peko	C1y	hsa05221
normal	normal	0.075	64.05	61.57	2.07	197.3	1289.8	0.0024
	lower	t-out	t-out	t-out	67.43	t-out	t-out	1.1416
	postopt	0.041	t-out	t-out	7.88	t-out	t-out	0.0018
	upper	0.073	65.74	61.53	2.07	196.8	1286.2	0.0022
	triangle	1.368	t-out	m-limit	71.95	570.1	m-limit	0.0367
sparse	normal	0.011	31.84	19.12	0.60	7.7	t-out	0.0007
	lower	t-out	85.45	t-out	17.51	115.5	t-out	0.0275
	upper	0.011	38.74	19.32	0.58	7.3	t-out	0.0010

Table 8.3.: Running times of LP-solver in seconds including the runtime of the oracle and post-optimization for different graphs. The time-limit was set to 3600 seconds. In each run we inverted one setting of Section 8.2.

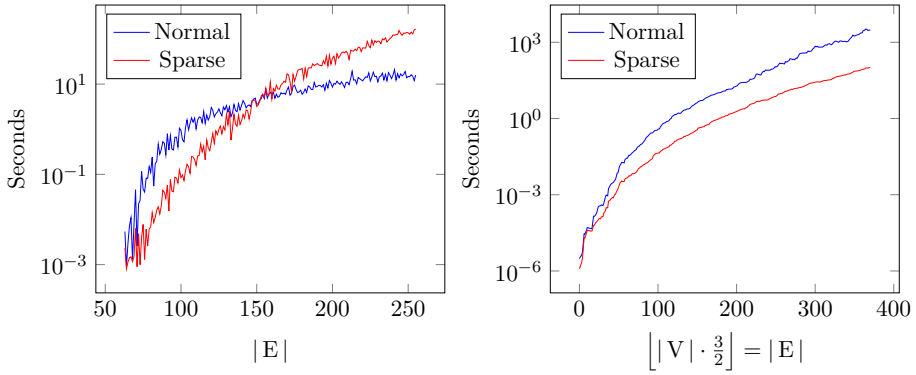

Figure 8.1.: Comparison of sparse and normal spreading LP for random graphs, on the left-hand side with 64 vertices and a increasing number of edges. On the right-hand side with $\lfloor |V| \cdot \frac{3}{2} \rfloor = |E|$ and a lowpass over the data. We can see that for denser graphs the normal LP-formulation is much faster, but for sparse graphs one can benefit from the small number of variables.

8.3. Results

8.3.1. Spreading metric

In Section 8.3.1 we can see, how the LP behaves if we insert a few random edges to an existing grid. As we might expect the grid would more disrupted by an optimal solution of the spreading LP if we assign a higher weight to the random edges. An interesting observation about the figure on the left-hand side is, that the longest random edge has a very short length. The LP does not care about the position of the vertices in the graph. Vertices at the border have less neighbors, which makes the spreading bounds less restrictive.

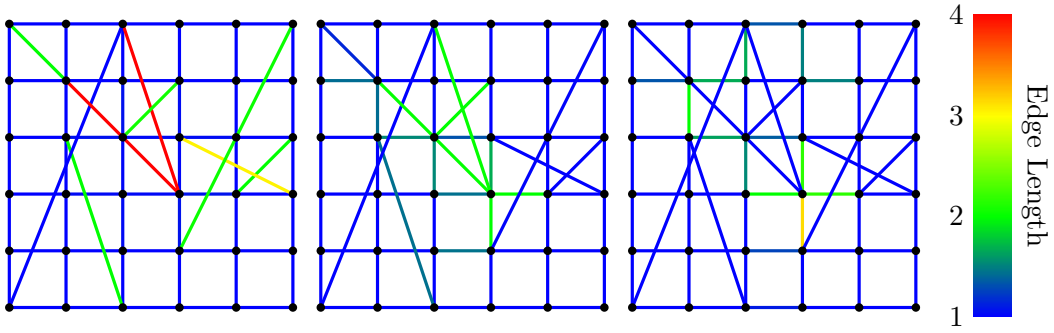

Figure 8.2.: Visualization of the edge length of an optimal LP solution. We have a grid with edge weight one on each edge. Furthermore, we have inserted some random edges with variable edge weights. At the left-hand side with weight one quarter, in the middle with weight one and at the right-hand side with weight four. For higher weights the lp disrupts the original metric.

Gridgraphs are interesting tests, because they provide a very simple perfect embedding (see Appendix A.1.4). Let $d \in \mathbb{N}$ be the dimension and $\mathrm{b} \in \mathbb{N}^d$ be a boundary. Let $\mathrm{G} = (\mathrm{V}, \mathrm{H})$ be a grid-hypergraph with boundary b and hyperedges of dimension k for $k \leq d$. If we now want to embed G into $[\mathrm{b}_1] \times \ldots \times [\mathrm{b}_d]$ then the optimal embedding is the embedding which is given by the construction. The value of this embedding is $k \cdot \prod_{i \in [d]} (\mathrm{b}_i - 1)$ in l_1 and $\prod_{i \in [d]} (\mathrm{b}_i - 1)$ in l_∞.

8.3.2. Tree metric

In Fig. 8.3 we see the resulting tree metric of Algorithm 4, Algorithm 5 and Algorithm 6 for instance HGr12. The edges are plotted as faces, colored and labeled by their length according to the tree metric. We can see that the probabilistic approach makes bad decisions resulting in unnecessarily many edges being cut on high levels. The estimators algorithm is the only algorithm that prevents the highest cut by choosing a vertex in the middle. For smaller root radii it is better to begin with other vertices as we can see in the lower middle.

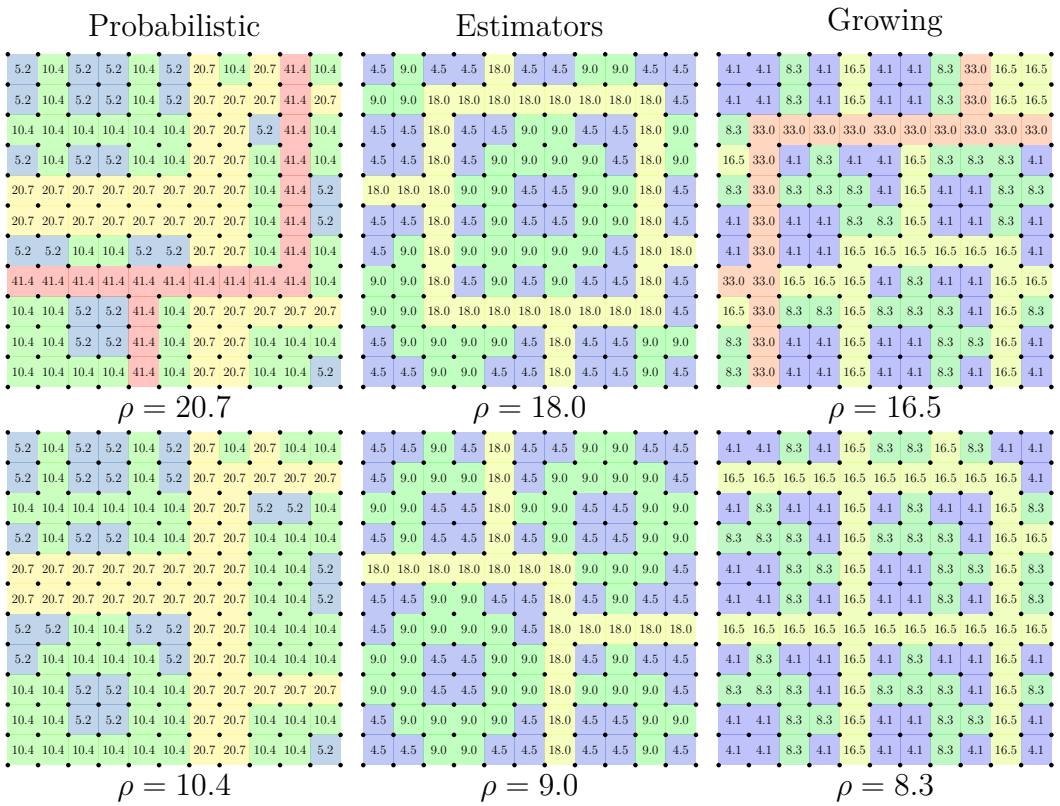

Figure 8.3.: Comparison of Algorithm 4, Algorithm 5 and Algorithm 6 on instance HGr12. All edges have length 1.5 in the solution of the spreading LP. ρ is the start-radius of the tree metric algorithms.

For gridgraphs with growing dimension we could expect a logarithmic grow of the ratio between our and the optimal solution value. Indeed this is the case as shown in Section 8.3.2. Furthermore, the growing tree algorithm seems to result in the best embeddings, and the probabilistic in the worst. Thus, even though the bound for the growing tree metric algorithm is worse than the pessimistic estimators derandomization, we should bear in mind that this only counts for a large number of edges.

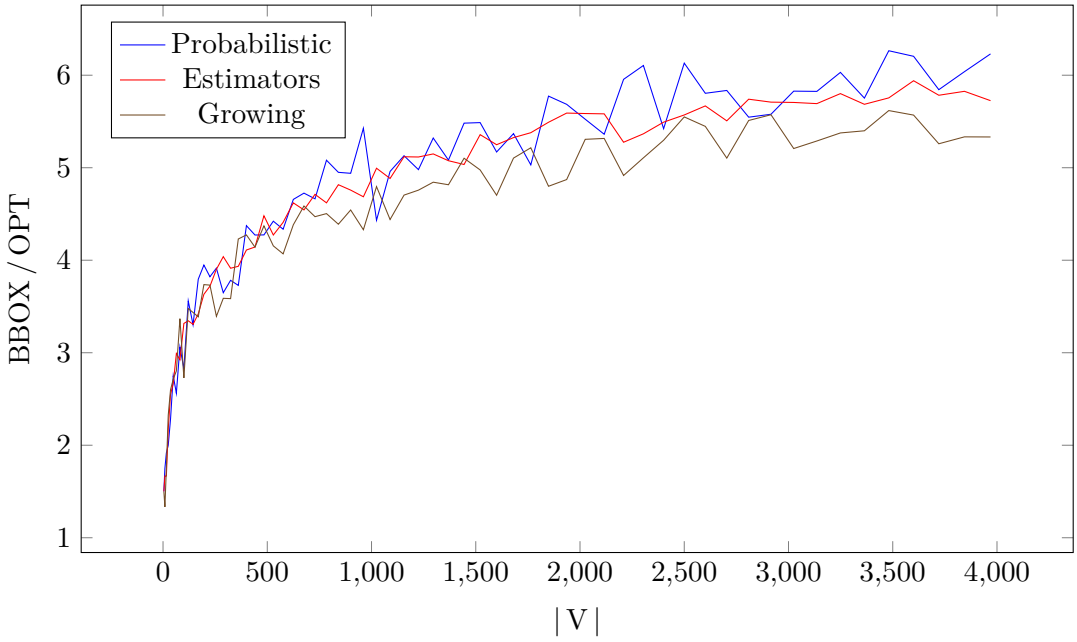

Figure 8.4.: Factor of Bounding-Box solution value to the optimum for an two dimensional gridgraph with increasing size and hyperedges of dimension one.

In Table 8.4 we can see that the choice of start radius and algorithm has a great impact on the volume of the closed tree. As we already mentioned by taking half the radius these values would only decrease. Furthermore, as one might expect, the results of the probabilistic tree algorithm are much worse than the others. The gap between the estimators and the growing algorithm is too small to deduce that one is significantly better.

In Table 8.5 we printed the linear arrangement value before we transformed the vertices with the Hilbert curve. There is no direct connection to the closed tree volume visible. Differences between the algorithms mostly cancel out. Moreover, we can see that the tree optimization improves the results by a large factor. The difference between the greedy and the enumerating optimization linear arrangement length is not very big. In some runs the greedy optimization leads to better results. In Fig. 5.8 we can see an example that proves the possibility of this behavior.

In Table 8.6 we see the bounding box value for the final embedding. Here the differences become much smaller again. This could be expected, because with the Hilbert-curve the length of an edge approximately transforms to the root d of its original length. We can still see that the optimization improves the embedding. The choice of the tree algorithm and rootradius only results in minor differences.

Radius	Algorithm	Gr16	HGr12	Suzanne	Rnd	Peko	C1y	hsa05221	Geom. av.
normal	probabilistic	7879.8	2139.1	8060.3	1135.4	5228.8	56 186	435.1	4107.2
	estimators	5523.8	1386	4898.9	1053.3	2691.5	26 299	228	2515.6
	growing	5261.3	1476.8	6326.1	1082.0	2991.2	27 308	266.8	2739.8
half	probabilistic	6714.4	1533.1	6176.1	939.2	3902.4	39 305	348.7	3166.8
	estimators	4965	1216.9	4362.0	769.5	2416.5	25 589	210	2217.1
	growing	5148.8	1126.1	4701.4	785.1	2527.0	25 454	217.3	2257.8

Table 8.4.: Closed tree volume

Halfrad	Optimization	Algorithm	Gr16	HGr12	Suzanne	Rnd	Peko	C1y	hsa05221	Geom. av.
false	none	probabilistic	9749	2873	28 328	1545	6866	113 666	167	5538.0
		estimators	12 944	3488	19 666	1452	8308	165 340	234	6345.8
		growing	10 568	2454	31 959	1404	6317	181 391	207	5987.9
	greedy	probabilistic	6768	1830	15 442	1323	4572	104 975	115	3909.8
		estimators	6613	1835	11 810	1079	4067	81 797	101	3394.7
		growing	6535	1657	14 386	1157	3770	84 202	123	3545.1
	enumerate	probabilistic	6741	1838	15 285	1313	4565	98 551	113	3854.5
		estimators	7717	1882	11 664	1083	3927	80 362	87	3379.7
		growing	6091	1653	14 380	1044	3837	81 172	121	3439.7
true	none	probabilistic	10 528	3871	24 038	1626	7153	114 175	172	5811.0
		estimators	11 314	2860	23 810	1661	6625	170 424	153	5801.0
		growing	10 065	2319	35 877	1962	9390	178 946	213	6671.1
	greedy	probabilistic	6011	1778	15 838	1205	4822	87 050	119	3737.5
		estimators	6483	1852	12 244	1025	3847	86 696	107	3411.4
		growing	6081	1723	13 625	988	3847	86 889	147	3537.4
	enumerate	probabilistic	6306	1855	15 845	1203	4792	86 988	119	3781.7
		estimators	6360	1836	12 186	1032	3768	80 369	107	3352.4
		growing	6229	1640	13 615	985	3851	75 625	136	3415.9

Table 8.5.: Linear arrangement value

Halfrad	Optimization	Algorithm	Gr16	HGr12	Suzanne	Rnd	Peko	C1y	hsa05221	Geom. av.
false	none	probabilistic	2257	887	3201	523	2006	13 768	107	1387.6
		estimators	2418	823	2982	456	1755	13 698	112	1328.0
		growing	2110	775	3198	464	1702	13 881	101	1285.2
	greedy	probabilistic	1830	677	2709	463	1265	11 575	73	1075.2
		estimators	1895	681	2451	431	1307	11 001	71	1048.3
		growing	1787	599	2573	423	1250	10 318	73	1013.3
	enumerate	probabilistic	1779	670	2692	453	1281	11 271	75	1066.9
		estimators	1957	691	2418	423	1288	10 814	65	1032.7
		growing	1651	650	2569	406	1227	10 998	73	1014.1
true	none	probabilistic	2308	925	3024	556	1871	12 821	104	1367.8
		estimators	2254	871	2960	471	1631	13 586	87	1268.2
		growing	2051	719	3158	534	1822	13 434	107	1306.9
	greedy	probabilistic	1879	721	2541	429	1329	10 416	77	1067.0
		estimators	1883	700	2503	375	1167	11 022	69	1013.5
		growing	1629	685	2548	382	1263	11 463	75	1023.8
	enumerate	probabilistic	1640	657	2540	437	1342	11 882	81	1064.1
		estimators	1926	704	2443	422	1145	11 621	69	1036.3
		growing	1663	649	2555	391	1198	10 967	82	1021.6

Table 8.6.: BBOX_1-value

9. Conclusion

We could show that the already known algorithm could be generalized to a bigger range of problems and provide a derandomization for the tree metric algorithm. Furthermore, we showed that one can find better constants for most approximation bounds, even though the bound we reach in the end is far away from any useful embedding. For most graphs it might be harder to find an embedding which is worse than the approximation bound than to find one which fulfills it. Anyway, the results of our program are much better, but still far away from any optimal solution and the runtime increases rapidly with larger graphs, making it impractical for any productive application.

9.1. Open problems

As we saw, the Hilbert space-filling curve seems to be perfectly suited for the Minkowski l_∞-metric but not for the l_1 metric. The reason is that it recursively subdivides l_∞ cubes. The question is if we could find curves which are more local in the sense of the l_1 metric and by this get rid of the factor d. The first approach may be to pack diamond bounded shapes together, but they unfortunately do not fit together in a way that we can build a bigger diamond shape out of some small shapes without leaving out points. Even so this might lead to a solution which scales better in higher dimensions. A possible curve could consist of diamond shaped grids packed together, leaving out some gaps, and then jump from adjacent points into these gaps.

Another problem, which is also connected to the embedding curve, is if we could consider domains which have more complex shapes. For simple shapes one might put together the domain by a few big, equal sized cubes, giving them an ordering and using Algorithm 8 to get the desired curve. But depending on the domain the constant of the bound would get worse and if we have a domain which is more complex we cannot use this approach.

Index

z

Variable	Explanation	Definition	Type
b	Cuboid boundary	3.1.6	Vector
$\mathcal{B}$	Blockages	3.1.8	Set
c	Edge cost function	2.2.10	Function
d	Dimension		Natural number
d	Distance function	2.2.7	Function
E	Edges	2.2.1	Set
G	Graph	2.2.1	Tuple
H	Hyperedges	2.2.1	Set
V	Vertices	2.2.1	Set
ρ	Root radius		Real positive number
P	Path	2.2.12	Triple
π	Graph embedding	3.1.6	Function
$\tilde{\pi}$	Linear embedding	5.5	Function
q	Hilbert injection		Function
Q	Domain	3.1.6	Set
sb	Spreading bound	4.2.3	Function
T	Tree	5.0.1	Triple
τ	Vertex permutation	Function	
$\mathcal{T}$	Terminals	3.1.9	Set
w	Edge volume	2.2.10	Function
Λ	Vertex clustering	5.0.6	Function

Table 9.1.: List of the variables which name the samee thing over the thesis

Bibliography

[1] Simon Ahrens. Generalizations of the d-dimensional arrangement problem. Master's thesis, Research Insttute for Discrete Mathematics, University of Bonn, 2013.

[2] David Applegate, William Cook, Sanjeeb Dash, and Monika Mevenkamp. QSopt Linear Programming Solver. `https://www.math.uwaterloo.ca/~bico/qsopt/`, 2018. [Online; accessed 03-September-2018].

[3] Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander Flows, Geometric Embeddings and Graph Partitioning. *J. ACM*, 56(2):5:1–5:37, April 2009.

[4] Blender Foundaition. Blender 3d computer graphics software, 2018. [Online; accessed 08-Oktober-2018].

[5] Ulrich Brenner, Daniel Rotter, and Ulrike Schorr. *2-dimensional Arrangement in Rectangles*. Forschungsinstitut für Diskrete Mathematik Rheinische Friedrich-Wilhelms-Universität Bonn, 2014.

[6] Chin-Chih Chang, Jason Cong, Michail Romesis, and Min Xie. Optimality and scalability study of existing placement algorithms. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23(4):537–549, 2004.

[7] Leonardo Dagum and Ramesh Menon. Openmp: an industry standard api for shared-memory programming. *Computational Science & Engineering, IEEE*, 5(1):46–55, 1998.

[8] Erik Dahlström, Patrick Dengler, Anthoni Grasso, Chris Lilley, Cameron McCormack, Doug Schepers, Jonathan Watt, Jon Ferraiolo, FUJISAWA Jun, and Dea Jackson. Scalable vector graphics (svg). Technical report, W3C, 2011.

[9] Guy Even, Joseph Seffi Naor, Satish Rao, and Baruch Schieber. Divide-and-conquer approximation algorithms via spreading metrics. *J. ACM*, 47(4):585–616, July 2000.

[10] S. Even and Y Shiloah. Np-completeness of several arrangement problems. *Technical Report ; Department of computer Science*, 43, 1975.

[11] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A Tight Bound on Approximating Arbitrary Metrics by Tree Metrics. In *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing*, STOC '03, pages 448–455, New York, NY, USA, 2003. ACM.

[12] Anupam Gupta and Anastasios Sidiropoulos. Minimum D-dimensional Arrangement with Fixed Points. In *Proceedings of the Twenty-fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '14, pages 1727–1738, Philadelphia, PA, USA, 2014. Society for Industrial and Applied Mathematics.

[13] Mark D Hansen. Approximation algorithms for geometric embeddings in the plane with applications to parallel processing problems. In *Foundations of Computer Science, 1989., 30th Annual Symposium on*, pages 604–609. IEEE, 1989.

[14] David Hilbert. Über die stetige Abbildung einer Line auf ein Flächenstück. *Mathematische Annalen*, 38(3):459–460, sep 1891.

[15] IBM. CPLEX Optimizer. `https://www.ibm.com/analytics/cplex-optimizer`.

[16] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311. ACM, 1984.

[17] Leonid Genrikhovich Khachiyan. A Polynomial Algorithm in Linear Programmin. *Doklady, 20, 191-194*, 1979.

[18] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms (Algorithms and Combinatorics)*. Springer, Berlin, 4 edition, 2008.

[19] T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In *[Proceedings 1988] 29th Annual Symposium on Foundations of Computer Science*, pages 422–431, Oct 1988.

[20] James D. Murray and William vanRyper. *Encyclopedia of Graphics File Formats (2Nd Ed.)*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1996.

[21] Gi-Joon Nam and Jarrod Roy. GSRC Bookshelf Format for placement instances. `http://vlsicad.eecs.umich.edu/BK/ISPD06bench/BookshelfFormat.txt`, 2018. [Online; accessed 28-Augist-2018].

[22] Rolf Niedermeier and Peter Sanders. On the Manhattan distance between points on space filling mesh indexings. Technical report, 1996.

[23] Marcus Oswald, Gerhard Reinelt, and Stefa Wiesberg. Exact solution of the 2-dimensional grid arrangement problem. *Discrete Optimization*, 9(3):189 – 199, 2012.

[24] Jordi Petit. Combining spectral sequencing and parallel simulated annealing for the MinLA problem. *Parallel Processing Letters*, 13(01):77–91, 2003.

[25] Prabhakar Raghavan. Probabilistic construction of deterministic algorithms: approximating packing integer programs. *Journal of Computer and System Sciences*, 37(2):130–143, 1988.

[26] Daniel Rotter and Jens Vygen. D-dimensional arrangement revisited. *Inf. Process. Lett.*, 113(13):498–505, July 2013.

[27] Gunnar Schramm, Eva-Maria Surmann, Stefan Wiesberg, Marcus Oswald, Gerhard Reinelt, Roland Eils, and Rainer König. Analyzing the regulation of metabolic pathways in human breast cancer. *BMC medical genomics*, 3(1):39, 2010.

[28] Wikipedia contributors. Trivial graph format — Wikipedia, the free encyclopedia, 2018. [Online; accessed 26-Augist-2018].

[29] Richard T. Wong. Linear programming, by vasek chvatal, w. h. freeman and company, new york, 1983, 478 pp., price: $24.95. *Networks*, 15(3):393, 1985.

List of Algorithms

List of Figures

List of Tables

A. Appendix

A.1. Program

In this chapter we describe the usage of the program which was written as a part of this thesis. We will describe the most important parameters, commands, graph generators and the input and output.

A.1.1. Parameters

Performance

Variable	Type	Default	Explanation
lp_tr_sc_factor	double	70	Weight factor for triangle inequalities
lp_hide_empty_cols	bool	false	Hide inactive cols from the solver
lp_ineq_eps	double	0.00001	Minimum violation of inequalities
lp_precalc_tr	bool	false	Precalculate triangle inequalities, see Section 4.3.4
lp_precalc_lb	bool	true	Use lower bounds, see Section 4.3.3
lp_precalc_ub	bool	false	Use upper bounds, see Section 4.3.6
lp_rand_ineq	bool	true	Randomize choice of inequalities
lp_remove_sl_factor	double	0.1	Remove inequalities after they have been slacked factor times resolves times in a row
lp_postopt	bool	true	Use Postoptimization, see Section 4.3.2
lp_type	string	normal	Lp-type, possible values are normal and sparse, see Definition 4.2.9 and Definition 4.3.1)
lp_solver	string		Which lp-solver should be used possible values are qsopt and cplex
num_threads	integer	0	Maximum number of parallel threads

Table A.1.: The most important program parameters

Tree algorithm

Variable	Type	Default	Explanation
optimize_tree	string	greedy	Which tree optimization should be used, possible values are none, greedy, enumerate, see Section 5.5
tree_algorithm	string	estimators	Algorithm which approximates the tree, possible values are growing, probabilistic, estimators see Chapter 5
tree_halfradius	bool	true	Half tree root-radius, see Remark 5.2.4
tree_rootradius	double	NaN	Overwrites the choice of the rootradius
tree_permutation	integer list		Overwrites the choice of the permutation

Table A.2.: Most important algorithm parameters

Instance

Variable	Type	Default	Explanation
size	integer list		The size of the cuboid in which the graph should be embedded
lp_spreading_bound	string	l1	Possible values are l1 and linfty, see Definition 4.2.3.

Table A.3.: Problem Instance Parameters

Logging

Variable	Type	Default	Explanation
loglevel	integer	0	Logging increases with higher values
output_dir	string	./output/	Directory for all output files
plot_graph_flat	bool	false	Print graph as a matrix. Use with patience, the file size is in $\Theta(\lvert V \rvert \cdot \lvert H \rvert)$
plot_tikz	bool	true	Print images which can be used in latex.
plot_svg	bool	true	Write Scalable Vector Graphic Plots, see [8]
plot_obj	bool	true	Write a Wavefront plot of the embedding, see [20]
clear_output_dir	bool	false	Clear output directiory at startup

Table A.4.: Logging parameters

A.1.2. Input

Command	Argument	Explanation
gridgraph	integer list	See Appendix A.1.4
torusgraph	integer list	See Appendix A.1.4
gridhypergraph	integer list bool list	See Appendix A.1.4
torushypergraph	integer list bool list	See Appendix A.1.4
randgraph	integer integer	
readbookshelfgraph	auxilary-file	See [21]
readtgfgraph	tgf-file	See [28]
readobjgraph	obj-file	See [20]
readvertexpositions	vertex position listTfile	See Appendix A.2.3

Table A.5.: File input commands

Trivial graph file-format

A definition of the format can be found in [28]. Graphs in this format can be loaded with the command readtgfgraph.

Bookshelf graph

Bookshelf graphs can be read with the command readbookshelfgraph <Auxiliary-File>. A description of the file format can be found on [21]. If there is a placement file in the same directory this is read as plot positions.

Wavefront file-format

A description of the file format can be found in the book [20]

A.1.3. Output

All output of the program is written to the folder defined in the option output_dir. The most important file is result.txt. The positions of the embedding are written into this file as a vertex position list (see Appendix A.2.3). Furthermore, information about the results and several plots can be found in the file log.html.

Debugging information

Variable	Explanation
Oracle Calls	Number of calls of the lp oracle
Lp inequalities	Number of inequalities at termination
Postopt calls	Number of calls of the post-optimization
LP variables	Number of variables of the LP solver at termination
LP variables real	Number of variables of the LP at termination

Table A.6.: Debugging information

Solution information

Variable	Explanation
Closed Tree Volume	The volume of the tree according to Definition 5.1.2
BBOX l1 value	The bounding box value in the l_1 norm
BBOX linf value	The bounding box value in the l_∞ norm

Table A.7.: Solution information

A.1.4. Graph generators

Gridgraph

Let $d \in \mathbb{N}$ be the dimension and $\mathrm{b} \in \mathbb{Z}^d$ be the cuboid boundaries. Then we first define the vertices:

$$\mathrm{V} = \bigtimes_{i=1}^{d} [\mathrm{b}_i]$$

We define the edges of dimension $k \in \{0, \ldots, d\}$ as

$$\mathrm{H}_k := \{h \subseteq \mathrm{V} : \mathrm{BBOX}_\infty(h) = 1 \wedge |h| = 2^k\}$$

We say G is a **grid-hypergraph** with boundary b and hyperegdes of dimension k if $G = (\mathrm{V}, \mathrm{H}_k)$.

Torusgraph

Let $d \in \mathbb{N}$ be the dimension and $\mathrm{b} \in \mathbb{Z}^d$ be the cuboid boundaries. Then we first define the vertices:

$$\mathrm{V} = \bigtimes_{i=1}^{d} [\mathrm{b}_i]$$

We define the edges of dimension $k \in \{0, \ldots, d\}$ as:

$$\mathrm{H}_k := \{h \subseteq \mathrm{V} : \exists v \in \mathrm{V} \text{ such that } \mathrm{BBOX}_\infty(h + v \bmod \mathrm{b}) = 1 \wedge |h| = 2^k\}$$

We say that G is a **torus-hypergraph** with boundary b and hyperegdes of dimension k if $G = (\mathrm{V}, \mathrm{H}_k)$.

A.2. File formats

A.2.1. Edge list

In the first line the number of vertices is stated, followed by lines which define the edges. The vertices are numbered from 0 to $|\,\mathrm{V}\,| - 1$

```
<number of vertices>
<vertex> ... <vertex>
...
<vertex> ... <vertex>
```

Figure A.1.: Edge list

A.2.2. Weighted edge list

In the first line the number of vertices is stated, followed by lines which define the edges. The vertices are numbered from 0 to $|\,\mathrm{V}\,| - 1$. The weights must take values in $\mathbb{R}_{>0}$.

```
<number of vertices>
<weight> <vertex> ... <vertex>
...
<weight> <vertex> ... <vertex>
```

Figure A.2.: Weighted edge list

A.2.3. Vertex position list

Each line defines a position of a vertex.

```
<coordinate> ... <coordinate>
...
<coordinate> ... <coordinate>
```

Figure A.3.: Vertex position list.