D-Width, Metric Embedding, and Their Connections

by

Mohammad Ali Safari

M.Math., University of Waterloo, 2003

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE STUDIES

(Computer Science)

The University of British Columbia

September 2007

Abstract

Embedding between metric spaces is a very powerful algorithmic tool and has been used for finding good approximation algorithms for several problems. In particular, embedding to an ℓ_1 norm has been used as the key step in an approximation algorithm for the sparsest cut problem. The sparsest cut problem, in turn, is the main ingredient of many algorithms that have a divide and conquer nature and are used in various fields.

While every metric is embeddable into ℓ_1 with distortion $O(\log n)$ [13], and the bound is tight [39], for special classes of metrics better bounds exist. Shortest path metrics for trees and outerplanar graphs are isometrically embeddable into ℓ_1 [41]. Series-parallel graphs [28] and k-outerplanar graphs [19](for constant k) are embeddable into ℓ_1 with constant distortion, planar graphs and bounded tree-width graphs are conjectured to have constant distortion in embedding to ℓ_1. Bounded tree-width graphs are one of most general graph classes on which several hard problems are tractable.

We study the embedding of series-parallel graphs (or, more generally, graphs with tree-width two) into ℓ_1 and also the embedding between two line metrics. We then move our attention to the generalization of tree-width to digraphs and hypergraphs and study several relevant problems.

Contents

Abstract ii

Contents iii

List of Tables vii

List of Figures viii

Acknowledgements x

1 Introduction 1

1.1 Metric Embedding 1

1.1.1 Important Metrics 2

1.2 ℓ_1 Metrics 3

1.2.1 An application 3

1.3 Line Metrics 8

1.4 Directed Metrics 10

1.5 Bounded tree-width graphs and digraphs 10

1.5.1 Undirected Tree Width 11

1.5.2 Directed Tree Width 12

1.5.3 Directed tree-width and Directed Metrics 13

1.5.4 Hyper-D-width 14
1.6 Organization 15

2 ℓ_1 embedding of series-parallel graphs 16
2.1 Introduction 16
2.2 Constructing the embedding 17
2.3 Flattening 20
2.4 Distortion 9.0 24
2.5 Distortion 6.0 25
2.6 Lower bound 29
2.7 Conclusion and Future Work 31
2.8 Proof of Lemma 4 32

3 Embedding between Line Metrics 40
3.1 Introduction 40
3.2 Preliminaries 42
3.3 Forbidden Permutations 44
3.4 Embedding between two line metrics 45
3.4.1 Algorithm 45
3.4.2 Largest Eigenvalue 46
3.4.3 Bounding d_k 48
3.5 Computing Separability 49
3.6 Pattern matching for permutations 50
3.7 Sortable Permutations 53
3.8 Conclusions and Future Work 55

4 D-Width **57**

4.1 Introduction 57

4.2 Definitions 60

4.2.1 D-width 60

4.2.2 Directed tree-width and haven order 62

4.2.3 Cops and robber game 64

4.3 Directed One Trees 66

4.3.1 Algorithmic results 70

4.4 Comparing D-width and directed tree-width 72

4.4.1 Arbitrary gap between different games 73

4.4.2 Arbitrary gap between D-width, directed tree-width, and haven order 76

4.5 Upper Bounds for D-width 78

4.5.1 Computing the strongly cop-monotonicity 80

4.6 Conclusion and Future Work 82

5 Hyper-D-width **83**

5.1 Introduction 83

5.2 Background 83

5.3 Hyper-D-width 87

5.3.1 Definition 87

5.3.2 Basic Properties 88

5.3.3 Stability 88

5.3.4 Comparison 90

5.3.5 cops and robber game 92

5.3.6 Applications 93

5.4 Introducing hyper-T-width 96
5.4.1 Properties 98
5.4.2 Computation 99

6 Conclusions and Research Directions 103
6.1 ℓ_1 metrics 104
6.2 Line metrics 104
6.3 D-width 105
6.4 Hyper-D-width 105

Bibliography 107

List of Tables

3.1 Distortion of $\hat{\pi}_k$ for several values of k. 49

3.2 d_k. 49

List of Figures

2.1 The graph $G^{x,e}$ for $e = (s_6, t_6)$. 20

2.2 Flattening a triangle. The right figure is really two degenerate triangles. 21

2.3 A pair of flattened triangles in a triangle sequence from x to an ancestor edge with endpoint y. Considering this case provides some intuition for our stronger, parameterized, inequality. 26

2.4 Lower bound example 30

3.1 The 4-separable permutation $(2, 4, 1, 3)$. 44

3.2 Algorithm. 46

3.3 Illustration of permutation $\hat{\pi}_{15}$. 48

3.4 A separation tree. 51

3.5 5-sortable permutation ('-' indicates coupling) 54

4.1 A digraph (left) with its D-decomposition of width one (right). . . . 61

4.2 Graph on which 4 cops have a winning strategy but 5 cops are required for robber-monotone strategy. 75

4.3 Graph on which 4 cops have a robber-monotone winning strategy but 5 cops are required for cop-monotone strategy. 75

4.4 Finding a strongly cop-monotone winning strategy 81

5.1 A hyper-D-decomposition with width two. 87

5.2 Solving decomposable problem P on a bounded hyper-D-width hypergraph 95

Acknowledgements

This thesis and my entire career owes a lot to many people. I want to take the opportunity to thank them all for their support and encouragement without which I could not reach where I am.

Throughout my PhD I have benefited from invaluable help that I received from my supervisor, Professor William Evans. I had regular weekly meetings with him during which I learned a lot from his insightful comments and brilliant mind and learned how to discuss scientific problems and how to efficiently do research. Will generously shared with me his valuable experience in research and helped me do my PhD research. I would like to deeply thank him for the opportunity that I had to have him as my supervisor.

I am grateful to my supervisory committee members, Prof. David Kirkpatrick and Prof. Pavol Hell, who helped me during the preparation of this thesis. I would like to thank David, Pavol, and also the university examiners, Prof. Joel Friedman and Prof. Richard Anstee for carefully reading the dissertation and for their valuable comments for improving it.

Before coming to UBC, I studied at the university of Waterloo and at Sharif university of technology. I have learned a lot by working with Prof. Prabhakar Ragde, Alex Lopez Ortiz, Therese Biedl, Mohammad Ghodsi, and Jafar Habibi and am grateful to them.

My time in UBC couldn't be as successful without having so many great and supportive friends. Thank you Armin, Alireza, Baharak, Bahram, Hamid, Hossein, Mahsa, Majid, Mohsen, Ramin, Reza, Saeid, Yaser, Zahra, and many others whose name is missing for all your help and support.

21 years of continuous studying wouldn't have been possible without the support of my great dad, mom, sisters and brothers. I missed them ever since I came to Canada, but they always supported me by regularly talking to me on the phone and praying for my success.

Last, but definitely not least, one person has given the most effort during these 4 years; my dear wife, Maryam. She made our home a very happy place to

live and always helped to relieve all of the stresses around me.

This thesis is dedicated to my parents, brothers, and sisters, to my dear Maryam, and to my dear Mana, my little daughter.

Credits. Parts of chapter 4 are based on an ongoing collaboration with Paul Hunter.

MOHAMMAD ALI SAFARI

The University of British Columbia
August 2007

Chapter 1

Introduction

The purpose of this chapter is to introduce metric embedding, directed tree-width and study their connections. Its content is a mixture of background information, newest research results in the literature on relevant topics, and a summary of our results which are fully explained in other chapters of the thesis.

1.1 Metric Embedding

A *metric* $M = (X, d)$ is a set of points X with non-negative distance function d defined on any pair of points with the constraint that the distances should satisfy the triangle inequality, i.e. $d(x, y) \leq d(x, z) + d(z, y)$, for all x, y, and z; moreover, distances are symmetric, i.e. $d(x, y) = d(y, x)$ for all x and y, and $d(x, x) = 0$, for all x[1].

Remark 1. *Throughout this thesis, we may specify a metric $M = (X, D)$ by its distance function D whenever X is clear from the context. So, we may write a metric D which, in fact, means a metric whose distance function is D.*

[1] In a metric, $d(x, y)$ is zero if and only if $x = y$. If we remove this constraint and allow zero distance between different points then the metric is called a *semi-metric*.

An embedding from one metric $A = (X_A, d_A)$ into another metric $B = (X_B, d_B)$ is a mapping f from X_A to X_B. The *expansion* of f, denoted by $exp(f)$, is the largest expansion over all distances, i.e. $\sup_{x,y} \frac{d_B(f(x),f(y))}{d_A(x,y)}$. The *contraction* of embedding f, denoted by $con(f)$, is the largest contraction over all distances, i.e. $\sup_{x,y} \frac{d_A(x,y)}{d_B(f(x),f(y))}$. The *distortion* of embedding f is defined as $exp(f) \times con(f)$.

During the last decade or so, low distortion embedding between metrics has been used extensively to design efficient approximation algorithms. The reason is simple: Some problems P are easily solvable if their input comes from some metric class $\mathcal{B}$ but are hard for inputs from some other metric class $\mathcal{A}$. If one can embed an input metric in $\mathcal{A}$ to a new metric in $\mathcal{B}$ with low distortion and solve P on the new metric and translate the result on the original metric, this usually yields an approximation algorithm for metrics in $\mathcal{A}$.

1.1.1. Important Metrics

Some metrics are of particular interest for researchers because of their role in finding approximation algorithms, or their connection to other interesting problems (see the survey by Piotr Indyk [33]). Every edge-weighted, undirected graph defines a metric where the points in the metric are vertices in the graph and the distance between two points is the shortest path length between the corresponding vertices. Notice that every metric corresponds to a complete graph with edge weights equal to the distance between edge end points. Some important metric classes of this type are those derived from planar graphs (that correspond to shortest paths of weighted planar graphs), trees, outer-planar graphs, k-outer-planar graphs, and bounded tree-width graphs.

Another type of metric that appears frequently in the literature is ℓ_k^d metric

which is the set of points in $\mathbb{R}^d$ where for any two points $X = (x_1, x_2, \cdots, x_d)$ and $Y = (y_1, y_2, \cdots, y_d)$ the distance between them is defined as

$$||X - Y||_k = \sqrt[k]{|x_1 - y_1|^k + |x_2 - y_2|^k + \cdots + |x_d - y_d|^k} \tag{1.1}$$

for $k = \infty$, $||X - Y||_\infty = \max\{|x_1 - y_1|, |x_2 - y_2|, \cdots, |x_d - y_d|\}$. In particular, the ℓ_1 metric, for its close connection with cut problems, the ℓ_2 or Euclidean metric, for its familiarity, and the ℓ_∞, for its simplicity of computation, are of interest for many researchers. Note that we use ℓ_k instead of ℓ_k^d when dimension is not a concern.

Embeddings between various important metrics have been studied before. For example, every n-point metric is isometrically embeddable(i.e. with distortion 1) into ℓ_∞, though with many dimensions ($O(n)$), and is embeddable into the ℓ_k metric with distortion $O(\frac{\log n}{k})$ [13, 39].

1.2 ℓ_1 Metrics

There are several reasons why ℓ_1 metrics are important to study. One main reason is their close connection with the multi-commodity flow problem and its dual version, the sparsest cut problem. While the maximum multi-commodity flow problem is solvable in polynomial time, the sparsest cut problem is known to be NP-hard [24]. In general, if for any length assignment of edges on a given graph the associated shortest path metric can be embedded into ℓ_1 with distortion α, then the sparsest cut problem can be solved on that graph with approximation factor at most α[28].

1.2.1 An application

In this section we show how embedding a graph metric into the ℓ_1 metric is used to obtain an $O(\log k)$ approximation for the sparsest cut problem with k terminal

pairs [6].

Let $G = (V, E)$ be an undirected graph with capacities on its edges. Given k terminal pairs (s_i, t_i) $(i = 1, 2, \cdots, k)$, $s_i, t_i \in V$, and k real valued demands, dem_i, the goal is to find a cut S, a subset of V, that minimizes the value $C(S)/dem(S)$, where $C(S)$ is the total capacity of edges between S and $\bar{S} = V - S$ and $dem(S)$ is the total demand of those pairs (s_i, t_i) that have one node in S and one in $\bar{S}$, i.e. $|\{s_i, t_i\} \cap S| = 1$.

Just like the maximum flow minimum cut relationship, there is a generalized maximum cut problem that corresponds to the sparsest cut problem. In the *concurrent flow problem* (or demand flow problem) k terminal pairs, (s_i, t_i) for $i = 1, 2, \cdots, k$, are given each having an associated demand, dem_i for the i^{th} commodity. The goal is to find the maximum fraction λ and concurrently routed flows, while respecting edge capacity constraints, such that the flow corresponding to each commodity is at least λ times their demands. One easy way to model this is by considering the flow associated with every path between commodity pairs. Let P_i be the set of all paths from s_i to t_i. Let p_i^j be the j^{th} path in P_i and f_i^j be the flow routed through it. The linear program is then as follows.

$$
\begin{array}{llll}
 & \textit{Maximize} \quad \lambda \textit{ subject to} & & \\
< d_e > & \sum_{e \in p_i^j} f_i^j \quad \leq c_e & \forall e & (LP1) \\
< \varphi_i > & \sum_j f_i^j \quad \geq \lambda dem_i & \forall i &
\end{array}
$$

The dual of LP_1 is as follows.

$$
\begin{array}{lll}
\textit{Minimize} \quad \sum_e c_e d_e \textit{ subject to} & & \\
\sum_{e \in p_i^j} d_e \quad \geq \varphi_i & \forall (i,j) & (LP2) \\
\sum_i dem_i \varphi_i \quad \geq 1 & \forall i &
\end{array}
$$

One can view d_e's as distances on edges. The first set of inequalities means φ_i is at most the shortest path value from s_i to t_i. As we better have φ_i as large as possible in the second set of inequalities, the above LP reduces to

$$\begin{array}{lll} \textit{Minimize} & \sum_e c_e d_e & \\ \sum_i dem_i d(s_i,t_i) & \geq 1 & \forall i \end{array} \qquad (LP3)$$

where $d(s_i,t_i)$ is the shortest distance between s_i and t_i defined by d_e's. Equivalently we can simplify $LP3$ as

$$\min_{d \text{ is a metric on } G} \frac{C(d)}{D(d)} \tag{1.2}$$

where $C(d) = \sum_e c_e d_e$ and $D(d) = \sum_{i=1\cdots k} dem_i d(s_i,t_i)$.

Let OPT^* be the solution to (LP3). For any partition $(S, V-S)$ the total capacity of edges between S and $V-S$ is $C(S)$ and there is a total demand of $dem(S)$ for commodities that have $|\{s_i,t_i\} \cap S| = 1$. Hence, the maximum fraction of demands that can simultaneously be satisfied is at most $C(S)/dem(S)$. Consequently, the minimum sparsest cut is at least OPT^*.

Let d be a distance function on the vertices of G such that there exists a set S and for every edge $e = \{x,y\} \in E_G$, $d_e = d(x,y) = 1$ if exactly one of x and y is in S, and is 0, otherwise. Such a distance function d defines a *cut metric* on the vertices of G. One can write the sparsest cut problem as a linear programming problem as follows.

$$\min_{d \text{ defines a cut-metric}} \frac{\sum_{(x,y)\in E} c(x,y)d(x,y)}{\sum_{i=1\cdots k} dem_i d(s_i,t_i)} = \frac{C(d)}{D(d)} \tag{1.3}$$

If we relax the condition that d defines a cut metric and allow it to be any ℓ_1 metric then the answer would be the same because of the following two lemmas.

Definition 1. *Given m metrics $M_1 = (X, d_1)$, $M_2 = (X, d_2)$, $\cdots$, $M_m = (X, d_m)$, their sum, $M = M_1 + M_2 + \cdots + M_m$ is a metric (X, d) such that for any pair (x, y), $d(x, y) = \sum_{i=1}^{m} d_i(x, y)$. M is a positive weighted sum of metrics $M_1, M_2, \cdots, M_m$ if there exist positive values $w_1, w_2, \cdots, w_m$ such that $d(x, y) = \sum_{i=1}^{m} w_i d_i(x, y)$ for all pairs (x, y).*

Lemma 1 (Folklore). *Every ℓ_1 metric can be written as a weighted sum of cut metrics.*

Proof. Every ℓ_1 metric is a sum of line metrics (i.e. ℓ_1^1 metrics) that correspond to each dimension. So, it suffices to prove it for line metrics. Assume that M is a line metric of n one-dimensional points $x_1, x_2, \cdots, x_n$ and $x_1 \leq x_2 \leq \cdots \leq x_n$. Let S_i be a cut metric in which the distance between points p and q (where $p < q$) is one only if $p \leq i$ and $q > i$ and is zero, otherwise. Let $\alpha_i = x_{i+1} - x_i$. We claim that M can be written as $\sum_i \alpha_i S_i$. Let p and q be two points and $p < q$. The distance between points p and q is $x_q - x_p = \sum_{i=p}^{q-1} \alpha_i$ which is equal to their distance in $\sum_i \alpha_i s_i$. □

Lemma 2. *For positive real values $a_i, \alpha_i, \beta_i (i = 1, 2, \cdots, n)$,*

$$\min_i \frac{\alpha_i}{\beta_i} \leq \frac{\sum_i a_i \alpha_i}{\sum_i a_i \beta_i}$$

Proof. Let $\lambda = \min_i \frac{\alpha_i}{\beta_i}$. Then, $\sum_i a_i \alpha_i \geq \lambda \sum_i a_i \beta_i$. Hence, $\frac{\sum_i a_i \alpha_i}{\sum_i a_i \beta_i} \geq \lambda$. □

Let δ be an ℓ_1 metric that minimizes the value $\frac{C(d)}{D(d)}$ over all ℓ_1 metrics. According to Lemma 1 we can write δ as a weighted sum of cut metrics. Let $\delta = w_1\delta_1 + w_2\delta_2 + \cdots + w_m\delta_m$ where each δ_i is a cut metric and w_i's are all positive. Hence, $C(\delta) = \sum_{i=1}^{m} w_i C(\delta_i)$ and $D(\delta) = \sum_{i=1}^{m} w_i D(\delta_i)$. Therefore, according to Lemma 2, there exists some index j such that $\frac{C(\delta_j)}{D(\delta_j)} \leq \frac{C(\delta)}{D(\delta)}$. Since δ minimizes the

value $\frac{C(d)}{D(d)}$ over all ℓ_1 metrics (that include cut metrics as well) we conclude that $\frac{C(\delta_j)}{D(\delta_j)} = \frac{C(\delta)}{D(\delta)}$.

Consequently, we can formulate the sparsest cut problem as a minimization problem over ℓ_1 metrics as follows.

$$\min_{d \text{ is an } \ell_1 \text{ norm}} \frac{\sum_{(x,y)\in E} c(x,y)d(x,y)}{\sum_{i=1\cdots k} dem_i d(s_i,t_i)} \tag{1.4}$$

By Lemmas 1 and 2, the value of (1.4) is equal to (1.3) and a solution to (1.3) can be easily obtained from any solution to (1.4).

Let OPT be the answer to the sparsest cut problem. As mentioned before, $OPT \geq OPT^*$. Let d^* be the corresponding distance function to OPT^*.

How much does OPT^* differ from OPT? According to Bourgain's theorem [13] d^* can be embedded into some ℓ_1 metric d^+ with distortion $O(\log n)$. Without loss of generality assume that the embedding from d^* to d^+ has no expansion. So,

$$\begin{aligned} OPT \geq OPT^* = \frac{\sum c_e d_e^*}{\sum dem_i d^*(s_i,t_i)} &\geq \frac{\sum c_e d_e^+}{\sum dem_i d^*(s_i,t_i)} \\ &\geq \frac{\sum c_e d_e^+}{O(\log n)\sum dem_i d^+(s_i,t_i)} \geq \frac{OPT}{O(\log n)} \end{aligned}$$

Hence, d^+ gives an $O(\log n)$ approximation for the minimum sparsest cut.

Notice that what is important in the above inequality is the maximum contraction of distances $d^+(s_i,t_i)$. With a slight modification to the embedding algorithm one can use Bourgain's theorem and make an embedding that is not an expansion and guarantees that the contraction for only terminal vertices is not more than $O(\log k)$, where k is the number of terminals. So, we can improve the approximation factor to $O(\log k)$. See [6] for more details.

In order to improve the approximation factor, one need only improve the distortion of the embedding into ℓ_1. For some graphs, in particular unit-weight expander graphs, this is not possible and the $O(\log n)$ bound for the distortion is tight [39]. Several researchers have then tried to find better distortion for various classes of graphs. Planar graphs and bounded tree-width graphs are two widely known classes that are conjectured to be embeddable into ℓ_1 with constant distortion. Rao [43] proves distortion $O(r^3\sqrt{\log n})$ for any $K_{r,r}$ minor free class of graphs. This yields $O(\sqrt{\log n})$ for planar graphs and bounded tree-width graphs. Outerplanar graphs are a class of graphs that are isometrically embeddable into ℓ_1. Gupta et al. [28] show that the distortion for series-parallel graphs is at most $7 + 4\sqrt{3} \simeq 13.928$. We have improved this value to 6.0 and proved a lower bound 3.0 for the embedding algorithm provided by Gupta et al.. Later, Chekuri et al. [19] prove that k-outerplanar graphs can be embedded into random trees, and hence into ℓ_1, with constant distortion, namely, $O(c^k)$, for some constant c. Finally, Carrol et al. [16] recently found a low distortion embedding into ℓ_1 for bounded bandwidth graphs.

1.3 Line Metrics

A simple and interesting subclass of ℓ_1 metrics are line metrics. A line metric is a set of points on a real line with distances measured using the ℓ_1 norm (using any ℓ_k norm is equivalent). Thus, line metrics are one dimensional versions of ℓ_1 metrics (ℓ_1^1). Because of their simplicity and their many applications, line metrics are often the target metric for low distortion embedding.

Bădoiu et al. [9] consider the problem of embedding a fixed graph metric into the best line metric. That is, the authors choose the position of the points on

a line to minimize distortion.

For the case that G is unit-weighted and has an optimal line embedding with distortion c, they propose a $O(n^3c)$ time algorithm that finds an embedding with distortion $O(c^2)$. Since they can always find an embedding with distortion $O(n)$ (in linear time) the best of these two embeddings gives an $O(\sqrt{n})$-approximation for the optimal embedding. For unit-weighted trees, they propose an embedding with distortion $8\Delta\sqrt{c\log c}+4c$ where Δ is some parameter known to be at most c. This yields a distortion $\tilde{O}(n^{1/3})$ in general[2]. They also provide an exact algorithm which has running time $n^{O(c)}$. In case that G is weighted and $c = 1+\epsilon < 1.5$, they obtain an $O(n^2)$ algorithm that finds an embedding f with distortion $1+O(\epsilon)$.

Later on, Bădoiu et al.[8] consider the problem of embedding metrics corresponding to weighted graphs into the line. Let the minimum inter-point distance be 1 and the maximum be Δ. They propose an approximate embedding with distortion $O(\Delta^{3/4}c^{11/4})$ and $c^{O(1)}$ for embedding general weighted graphs and weighted trees, respectively. For the latter case, they prove that it is hard to approximate the optimal embedding by $\Omega(\sqrt{n})$. In all cases, c is the optimal distortion. We discuss this result further in Section 3

Kenyon et al. [36] consider the problem of optimally embedding one fixed line metric into another fixed one. This problem is different from what we have seen so far in the sense that the target metric is fixed and one only needs to find the right mapping between points in the input and target metrics. Such a problem has applications in shape matching and object recognition. Kenyon et al. propose a dynamic programming based algorithm that computes the optimal embedding in time $O(n^{12})$ in case the distortion is less than $3+2\sqrt{2} \simeq 5.829$. We later describe

[2] $\tilde{O}$ means a rough approximation of O with log factors ignored. For example, $n\log n = \tilde{O}(n)$.

a family of dynamic programming algorithms that compute optimal embeddings in polynomial time provided the distortion is less than 13.60. The latter result was independently found by Kenyon et al. in an extension of their conference paper and also by Chandran et al. [17].

1.4 Directed Metrics

There are several problems whose underlying metric is not necessarily symmetric. A trivial example is optimization metric problems on directed graphs such as finding a shortest path. There have been some recent attempts to extend symmetric (undirected) metrics to asymmetric (directed) ones. Inspired by the directed version of cut problems, Charikar et al. [18] study directed metrics for the first time and propose directed invariants of ℓ_1 metrics, ℓ_2^2 metrics, and some other metrics and study their relationship with directed cut metrics.

1.5 Bounded tree-width graphs and digraphs

Tree-width has many connections to what we have talked about so far. For ℓ_1 metrics, it is conjectured that bounded tree-width graphs are embeddable into ℓ_1 with constant distortion. Gupta et al. [28] show that the distortion for series-parallel graphs (and, in fact, for all graphs with tree-width 2) is at most $7 + 4\sqrt{3} \simeq 13.928$. Trees, that have tree-width one, are also isometrically embeddable into ℓ_1. For the multi-cut problem[3], Calinescu et al. [15] provide a polynomial time approximation scheme (PTAS) for bounded degree and bounded tree-width graphs and digraphs.

[3]The multi-cut problem is related to the sparsest cut problem. In the multi-cut problem, there are k pairs of terminals and the aim is to delete edges of minimum total weight to disconnect all terminal pairs.

In this section we review the definitions of tree-width for graphs and digraphs and discuss some related results.

1.5.1 Undirected Tree Width

The notion of tree-width is considered as a generalization of trees (trees have tree-width 1) and many intractable problems are efficiently solvable on bounded tree-width graphs. Examples include Hamiltonian cycle, graph isomorphism, vertex coloring, and edge coloring.

A *tree-decomposition* of an undirected graph $G = (V, E)$ is a pair (T, W), where T is a tree, and W is a function that assigns to every node i of T a subset W_i of vertices of G such that

1. $\bigcup_{i \in T} W_i = V$.
2. For each edge $(u, v) \in E$, there exists some node i of T such that $\{u, v\} \subset W_i$.
3. For all nodes i, j, k in T, if j is on the unique path from i to k then $W_i \cap W_k \subset W_j$.

The *width* of a tree-decomposition (T, W) is the maximum of $|W_i| - 1$ over all nodes i of T. The *tree-width* of G is the minimum width over all tree-decomposition of G.

Notice that the above conditions can be interpreted as follows. For any connected set S of G,

(G1) $T|_S := \{t | W_t \cap S \neq \emptyset\} \neq \emptyset$, and

(G2) The subgraph of T with vertex set $T|_S$ and edges $\{(s, t) | W_s \cap W_t \cap S \neq \emptyset\}$ forms a connected subtree of T.

It can be easily shown that it suffices that the conditions G1 and G2 be true for only edges and vertices, i.e. minimal connected sets. A connected set S is minimal if there do not exist connected proper subsets A and B of S such that $A \cup B = S$ and $A \cap B \neq \emptyset$.

1.5.2 Directed Tree Width

In 1996 Reed et al. [46] proved Youngers's conjecture [53] roughly saying that every directed graph has either a large set of disjoint directed circuits or a small set of vertices that cover all directed circuits. In their paper, they defined a version of *well-linked* sets for directed graphs and since the size of the largest well-linked set in undirected graphs has close relationship with tree-width[45] they suggested that the analogous definition of tree-width for directed graphs might be very useful, as pointed out in [44]. We believe that a proper definition should ideally measure the global connectivity of a digraph. For example the tree-width of a directed acyclic graph(DAG) should be small because it has low connectivity.

Unfortunately finding a definition for directed tree-width analogous to the undirected case is not easy, since almost all concepts related to undirected tree-width behave differently in directed graphs. [For example, the bramble number is equal to the haven order in undirected graphs, while they may differ by a factor of two in directed graphs [48].] There is not an agreed-upon generalization of tree-width for directed graphs.

For the first time Johnson, Robertson, Seymour, and Thomas[34] gave a formal definition of directed tree-decomposition (called *arboreal-decomposition* in their paper) and directed tree-width, and proved some theorems relating directed tree-width and haven order. Other researchers proposed different definitions for

the directed tree-width. Safari [47] introduces D-width as an alternative definition for directed tree-width and proved some facts to justify his definition as a proper measure for directed tree-width.

Definition 2 (D-decompositions and D-width). *A* D-decomposition *of a directed graph G is a pair (T, W) where T is a tree and $W = \{W_t | t \in V(T)\}$ is a family of subsets of $V(G)$ such that for every strongly connected set $S \subseteq V(G)$:*

(D1) $T|_S := \{t | W_t \cap S \neq \emptyset\} \neq \emptyset$, *and*

(D2) The subgraph of T with vertex set $T|_S$ and edges $\{(s,t) | W_s \cap W_t \cap S \neq \emptyset\}$ forms a connected subtree of T.

A subset S of vertices of G is strongly connected if $G[S]$ is strongly connected. The width *of a D-decomposition (T, W) is the minimum k such that $|W_i| \leq k + 1$ for all $W_i \in W$. The* D-width *of a directed graph G is the minimum width over all D-decompositions of G.*

In Chapter 4 we further extend these results and obtain lower and upper bounds for D-width in terms of certain cops/robber games on digraphs and other parameters defined on digraphs. We also characterize the class of digraphs whose D-width is one.

1.5.3 Directed tree-width and Directed Metrics

The fact that bounded tree-width graphs have a close relation with ℓ_1 metrics quickly brings to mind that bounded tree-width digraphs might have good connections to directed metrics. Bounded tree-width digraphs are actually known to be connected to cut problems: Calinescu et al. [15] propose a PTAS for bounded degree, bounded tree-width digraphs for the directed multicut problem on unit-weighted graphs. A

directed multicut in a digraph is a set of edges (or vertices in the vertex version) whose removal leaves no strongly connected component containing both vertices of a terminal pair (s_i, t_i).

1.5.4 Hyper-D-width

The way that D-width is defined suggests that it can be extended to hypergraphs. In a D-decomposition of a digraph G, the subtrees corresponding to vertices of every strongly connected set S must form a connected subtree together. S is, in fact, taken from the set of minimal connected units of digraph G. If G was undirected then S was any edge or any vertex. In case of hypergraphs, the minimal connected units are single vertices or hypergraph edges. Let us formally define hyper-D-width.

Let $H = (V, E)$ be hypergraph. A hyper-D-decomposition of a H is a pair (T, W) where T is a tree and $W = \{W_t | t \in V(T)\}$ is a family of subsets of $V(H)$ such that for every connected set $e \in E(H)$:

(H1) $T|_e := \{t | W_t \cap e \neq \emptyset\} \neq \emptyset$, and

(H2) The subgraph of T with vertex set $T|_e$ and edges $\{(s,t) | W_s \cap W_t \cap e \neq \emptyset\}$ forms a connected subtree of T.

The width of a hyper-D-decomposition (T, W) is the maximum of $|X_i| - 1$ over all nodes $i \in T$. The hyper-D-width of a hypergraph is the minimum width over all its hyper-D-decompositions.

Hyper-D-width is useful for solving several hard problems. In particular, we will show, in Chapter 5, how we can find polynomial-time approximation schemes (PTAS) for vertex cover, dominating set, and multicut problems on hypergraphs when the hyper-D-width of the input hypergraph is constant.

Next, for the purpose of computability, we introduce another measure, called hyper-T-width, which is slightly different from hyper-D-width, inherits almost all algorithmic and structural properties of hyper-D-width, and, in contrast to hyper-D-width, is computable when hyper-T-width is constant.

1.6 Organization

In Chapter 2 we go through the details of our results on embedding series parallel graphs into ℓ_1 with distortion 6.0. In Chapter 3 we talk about embedding between line metrics and discuss the usage of k-separable permutations in embedding between fixed line metrics and in other applications. We then move our attention to tree-width on digraphs and hypergraphs. In Chapter 4 we review existing results on generalization of tree-width on digraphs. In particular, we study D-width and characterize the class of digraphs with D-width one. We also compare D-width with several other parameters defined on digraphs. Next, we generalize tree-width to hypergraphs in Chapter 5 and compare our definition, hyper-D-width, with other existing connectivity measures on hypergraphs. We also find several algorithmic applications of hyper-D-width. We finally introduce hyper-T-width which is slightly different from hyper-D-width and has the advantage that it is computable in polynomial time when hyper-T-width is constant.

Chapter 2

ℓ_1 embedding of series-parallel graphs

2.1 Introduction

The ℓ_1 metric embedding is of particular interest for its connection to the sparsest cut problem which, in turn, is the main ingredient of various algorithms that have a divide and conquer nature [28]. As outlined in Section 1.2.1, the sparsest cut problem can be interpreted as a minimization problem over ℓ_1 metrics. One can solve the problem as a minimization over all shortest path metrics defined on the underlying graph, by linear programming, and then embed the solution into ℓ_1. The distortion incurred by the embedding is essentially the same as the approximation factor. In fact, if for any edge weights on a graph G, we can embed the corresponding metric into ℓ_1 with distortion c then the sparsest cut could be approximated on G with factor c.

While every metric is embeddable into ℓ_1 with distortion $O(\log n)$ [13] and the bound is realized by graph metrics for expander graphs [39], for special classes

of metrics better bounds exist. Graph metrics for trees and outerplanar graphs are isometrically embeddable into ℓ_1 [41]. In fact, a shortest path metric corresponding to a graph G is isometrically embeddable into ℓ_1 if and only if G exclude $K_{2,3}$ as a minor [28]. Series-parallel graphs [28] and k-outerplanar graphs [19](for constant k) are embeddable into ℓ_1 with constant distortion. Planar graphs and bounded tree-width graphs are two widely know classes that are conjectured to be embeddable into ℓ_1 with constant distortion. Rao [43] proves distortion $O(r^3\sqrt{\log n})$ for any $K_{r,r}$ minor free class of graphs. This yields distortion $O(\sqrt{\log n})$ for planar graphs and bounded tree-width graphs.

In this chapter, we prove an upper bound of 6.0 on the distortion of embedding series-parallel graphs into ℓ_1. We also prove a lower bound of 3.0 for the embedding algorithm given by Gupta et al. [28] even when the input metric is isometrically embeddable into ℓ_1.

2.2 Constructing the embedding

In this section, we outline the method that Gupta et al. [28] use to obtain a constant-distortion embedding of series-parallel graphs into ℓ_1.

Series-parallel graphs are often defined in a recursive fashion: An edge (s,t) is a series-parallel graph with terminals s and t. If G_1 (resp. G_2) is a series-parallel graph with terminals s_1 and t_1 (resp. s_2 and t_2) then a *series* construction creates a new series-parallel graph, with terminals s_1 and t_2, by taking the union of G_1 and G_2 and unifying t_1 with s_2. A *parallel* construction creates a new series-parallel graph, with terminals s_1 and t_1, by taking the union of G_1 and G_2, and unifying s_1 with s_2 and t_1 with t_2.

An alternative way of constructing series-parallel graphs is more incremental.

We start with an edge. At each step, we choose an existing edge (s,t), introduce a new vertex x, and connect it to both s and t by edges (x,s) and (x,t). At the end of the construction, we may remove any subset of edges. This actually constructs all tree-width-2 graphs, which are more general than series-parallel graphs. Also we may assume that no edges are removed at the end of the construction since we may choose the weight of every removed edge to be infinity. Gupta et al. use this incremental construction to define an ℓ_1-embedding of the graph, which is the embedding that we analyze in this section. Consequently, all the results in this section apply to tree-width-2 graphs.

Gupta et al.[28] present two fundamentally different methods for embedding series-parallel graphs into ℓ_1 with constant distortion. The first one, which yields a distortion factor at most 13.92, recursively computes an ℓ_1-embedding as a sum of cut-metrics (See the definition in Section 1.2.1). Their second approach is to represent series-parallel graphs as a probabilistic sum of trees and bundles (special series-parallel graphs in which all paths between the two terminals have the same length) with distortion at most 8. Using the fact that trees are isometrically embeddable into ℓ_1 and bundles are ℓ_1-embeddable with distortion at most 2, they conclude with an ℓ_1-embedding with distortion at most 16.

We focus on their first approach. They use the incremental construction of series-parallel graphs to compute the ℓ_1-embedding as follows:

Let $\mu(x,y)$ be the shortest path distance between two vertices x and y and $\tilde{\mu}(x,y)$ be the ℓ_1-distance to be computed. Initially, when the construction starts with a single edge (s,t) we set $\tilde{\mu}(s,t) = \mu(s,t)$. Assume that in one step we introduce one vertex x and attach it to the endpoints of an existing edge (s,t). Let

$$\delta = \frac{\mu(x,s) + \mu(x,t) - \mu(s,t)}{2} \quad \text{and} \quad P_s = \frac{\mu(x,t) - \mu(x,s) + \mu(s,t)}{2\mu(s,t)}$$

and for every existing vertex y let

$$\tilde{\mu}(x,y) = \delta + P_s\tilde{\mu}(s,y) + (1-P_s)\tilde{\mu}(t,y). \tag{2.1}$$

First, $\tilde{\mu}$ is isometrically embeddable into ℓ_1 since it is the sum of a cut metric and two isometrically embeddable metrics. Next, to show that $\tilde{\mu}$ has low distortion, it is easy to verify that $\tilde{\mu}$ preserves edge lengths, i.e., for every edge $(x,y) \in G$, $\tilde{\mu}(x,y) = \mu(x,y)$. However if x and y are not adjacent in G, we need to show that $\tilde{\mu}(x,y) \geq \mu(x,y)/c$ to prove that the distortion is at most c. The fact that $\tilde{\mu}(x,y) \leq \mu(x,y)$ follows from μ being a shortest path metric and from every edge length being preserved in the new distance.

To show that $\tilde{\mu}(x,y) \geq \mu(x,y)/c$, Gupta et al. consider two cases based on the ancestor relation between x and y. The *ancestor edges* of vertex x are the *parent edge* (s,t) to which x is attached during the incremental construction plus all the ancestor edges of s and t. Their cases are:

Case 1: y lies on an ancestor edge of x.

Case 2: Neither x nor y lies on an ancestor edge of the other.

For case 1, which turns out to determine the distortion of the embedding, Gupta et al. use an inductive argument to prove that

$$\tilde{\mu}(x,y) \geq \frac{(1-\xi)(2\xi-1)}{1+\xi}\mu(x,y)$$

for all $\xi \in (\frac{1}{2}, 1)$. In particular, for $\xi = \sqrt{3}-1$ this gives the best bound $\frac{(1-\xi)(2\xi-1)}{1+\xi} \simeq \frac{1}{13.92}$. We show that the worst distortion in case 1 occurs when the sequence of ancestor edges from x to y has a special degenerate form (Lemma 3). Using this fact and the proof technique of Gupta et al., we can show that the distortion of $\tilde{\mu}$ is at most 9.0. However, in order to obtain our result, that the distortion of $\tilde{\mu}$ is at

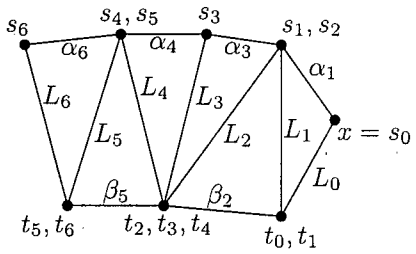

Figure 2.1: The graph $G^{x,e}$ for $e = (s_6, t_6)$.

most 6.0, we need a more precise inductive argument than the one used by Gupta et al. This argument appears in Lemma 4.

We also show, in section 2.6, that the algorithm of Gupta et al. produces an ℓ_1 metric $\tilde{\mu}$ with distortion at least 3.0 on a family of outer-planar graph metrics; metrics that are known to be isometrically embeddable into ℓ_1.

2.3 Flattening

For a vertex x and an ancestor edge $e = (s, t)$ of x, let $\langle (s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k) = (s, t) \rangle$ be the sequence of ancestor edges of x from x to (s, t). That is, (s_i, t_i) is the parent edge of either s_{i-1} or t_{i-1} depending on whether $t_i = t_{i-1}$ or $s_i = s_{i-1}$ respectively. To simplify our definitions, we assume $s_0 = x$ and $t_0 = t_1$. Note that for every $1 \leq i \leq k$ either $t_i = t_{i-1}$ or $s_i = s_{i-1}$. Let $G^{x,e}$ be the induced subgraph of G that contains x and $\{s_i, t_i | 1 \leq i \leq k\}$. The graph $G^{x,e}$ is a sequence of edge-weighted triangles. Let $L_i = \mu(s_i, t_i)$, $\alpha_i = \mu(s_{i-1}, s_i)$, and $\beta_i = \mu(t_{i-1}, t_i)$. See Figure 2.1. It is important to note that the shortest path between any two vertices in $G^{x,e}$ is the same as the shortest path between those two vertices in G. Also, the definition of $\tilde{\mu}$ on the (series-parallel) graph $G^{x,e}$ is the same as $\tilde{\mu}$ on the original graph G restricted to the vertices in $G^{x,e}$, as long as the order of construction used in the two definitions is the same.

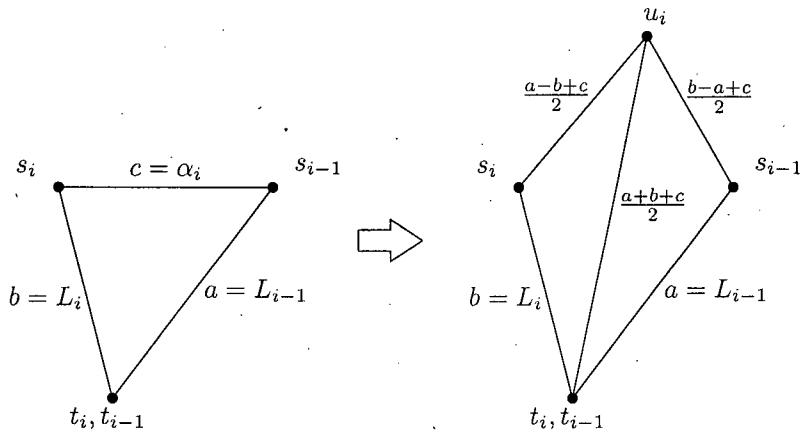

Figure 2.2: Flattening a triangle. The right figure is really two degenerate triangles.

A triangle with edge lengths a, b, and c is *flat* if $a = b + c$ or $a = |b - c|$. The *flattened* version, $F^{x,e}$, of $G^{x,e}$ contains two flat triangles for every triangle in $G^{x,e}$. If $s_i \neq s_{i-1}$ (and $t_i = t_{i-1}$) then $F^{x,e}$ contains the flat triangles (s_{i-1}, u_i, t_{i-1}) and (s_i, u_i, t_i) where u_i is a new vertex not in G and $\mu(s_{i-1}, u_i) = \frac{L_i - L_{i-1} + \alpha_i}{2}$, $\mu(u_i, t_{i-1}) = \frac{L_i + L_{i-1} + \alpha_i}{2}$, $\mu(s_{i-1}, t_{i-1}) = L_{i-1}$, $\mu(s_i, u_i) = \frac{L_{i-1} - L_i + \alpha_i}{2}$, and $\mu(s_i, t_i) = L_i$. If $t_i \neq t_{i-1}$ (and $s_i = s_{i-1}$) then $F^{x,e}$ contains the flat triangles (t_{i-1}, v_i, s_{i-1}) and (t_i, v_i, s_i) where v_i is a new vertex not in G and $\mu(t_{i-1}, v_i) = \frac{L_i - L_{i-1} + \beta_i}{2}$, $\mu(v_i, s_{i-1}) = \frac{L_i + L_{i-1} + \beta_i}{2}$, $\mu(s_{i-1}, t_{i-1}) = L_{i-1}$, $\mu(t_i, v_i) = \frac{L_{i-1} - L_i + \beta_i}{2}$, and $\mu(s_i, t_i) = L_i$. For example, see Figure 2.2.

The graph $F^{x,e}$ is series-parallel and it defines the same graph metric on the vertices of $G^{x,e}$ as the graph G does. That is, the shortest path distance between any two vertices in $G^{x,e}$ remains unchanged in $F^{x,e}$. We may also construct $F^{x,e}$ following the order induced by the construction order of G with u_i added after s_i and before s_{i-1} (and v_i added between t_i and t_{i-1}). Using this construction order, let $\tilde{\mu}_F$ be the ℓ_1 distance obtained by Gupta et al.'s construction (definition (2.1)) on $F^{x,e}$. We first prove that $\tilde{\mu}_F(x, y) \leq \tilde{\mu}(x, y)$.

Lemma 3. *For an endpoint y of an ancestor edge e of x,*

$$\tilde{\mu}_F(x,y) \leq \tilde{\mu}(x,y).$$

Proof. We prove, by induction of the order of addition of the vertices, that $\tilde{\mu}_F(w,y) \leq \tilde{\mu}(w,y)$ for every vertex w in $G^{x,e}$. If w is an endpoint of e then $\tilde{\mu}_F(w,y) = \tilde{\mu}(w,y) = \mu(w,y)$. Otherwise, assume $w = s_{i-1}$ and $s_{i-1} \neq s_i$ (the case $w = t_{i-1}$ is similar). According to the induction hypothesis, $\tilde{\mu}_F(s_i,y) \leq \tilde{\mu}(s_i,y)$ and $\tilde{\mu}_F(t_i,y) \leq \tilde{\mu}(t_i,y)$. Let $a = L_{i-1}$, $b = L_i$, and $c = \alpha_i$. By definition (2.1) of $\tilde{\mu}_F$, for $p_F = \frac{2a}{a+b+c}$,

$$\begin{aligned}
\tilde{\mu}_F(s_{i-1},y) &= 0 + p_F\tilde{\mu}_F(u_i,y) + (1-p_F)\tilde{\mu}_F(t_i,y) \\
&= p_F\left(\frac{a-b+c}{2} + \tilde{\mu}_F(s_i,y)\right) + (1-p_F)\tilde{\mu}_F(t_i,y) \\
&\leq p_F\left(\frac{a-b+c}{2} + \tilde{\mu}(s_i,y)\right) + (1-p_F)\tilde{\mu}(t_i,y)
\end{aligned}$$

By definition (2.1) of $\tilde{\mu}$, for $p = \frac{a-c+b}{2b}$,

$$\tilde{\mu}(s_{i-1},y) = \frac{c+a-b}{2} + p\tilde{\mu}(s_i,y) + (1-p)\tilde{\mu}(t_i,y)$$

Hence,

$$\begin{aligned}
\tilde{\mu}_F(s_{i-1},y) - \tilde{\mu}(s_{i-1},y) &\leq (p_F-p)(\tilde{\mu}(s_i,y) - \tilde{\mu}(t_i,y)) - (1-p_F)\frac{c+a-b}{2} \\
&\leq b|p_F-p| - (1-p_F)\frac{c+a-b}{2} \\
&= \frac{(a-b+c)(b-a+c)}{2(a+b+c)} - \frac{(a-b+c)(b-a+c)}{2(a+b+c)} \\
&= 0
\end{aligned}$$

□

As Gupta et al. mention [28], we can view the construction of $\tilde{\mu}$ as a probabilistic process. If we are at a vertex x with parent edge (s,t), we accumulate δ (for the triangle (x,s,t)) and then collapse (move) to either vertex s with probability P_s or to vertex t with probability $1-P_s$. By repeating this process, we move from x to the edge (s_k,t_k) and accumulate δ for some triangles in the sequence. Let P_s^i be the probability that when x moves to the edge (s_i,t_i) it moves to s_i and let $P_t^i = 1-P_s^i$. The expected sum of the accumulated δ's plus $P_s^k L_k$ (resp. $P_t^k L_k$) is $\tilde{\mu}(x,y)$ if $y=t_k$ (resp. $y=s_k$). Define Δ^i to be the expected sum of the δ's accumulated over all triangles up to the edge (s_i,t_i). So, for example, $\Delta^0 = 0$. Let $\Delta = \Delta^k$. Then

$$\tilde{\mu}(x,t_k) = \Delta + P_s^k L_k \qquad \text{and} \qquad \tilde{\mu}(x,s_k) = \Delta + P_t^k L_k.$$

As a corollary of Lemma 3, we have

Corollary 1. *For an endpoint y of an ancestor edge e of x,*

$$\tilde{\mu}_F(x,y) + \Delta_F \leq \tilde{\mu}(x,y) + \Delta$$

where Δ (resp. Δ_F) is the expected total of δ's over all triangles through which x is collapsed to y in $G^{x,e}$ (resp. $F^{x,e}$).

Proof. Let $e=(w,y)$ and $L=\mu(w,y)$. Assume when x collapses to e in $G^{x,e}$ (resp. $F^{x,e}$) it collapses to vertex y with probability p_G (resp. p_F) and to w with probability q_G (resp. q_F). According to Lemma 3, $\tilde{\mu}_F(x,y) \leq \tilde{\mu}(x,y)$ and $\tilde{\mu}_F(x,w) \leq \tilde{\mu}(x,w)$. Hence $\Delta_F + q_F L = \tilde{\mu}_F(x,y) \leq \tilde{\mu}(x,y) = \Delta + q_G L$. Similarly, $\Delta_F + p_F L = \tilde{\mu}_F(x,w) \leq \tilde{\mu}(x,w) = \Delta + p_G L$. Hence $\Delta_F \leq \Delta + \min\{q_G - q_F, p_G - p_F\} = \Delta + \min\{p_F - p_G, p_G - p_F\} \leq \Delta$. Consequently, $\tilde{\mu}_F(x,y) + \Delta_F \leq \tilde{\mu}(x,y) + \Delta$. □

2.4 Distortion 9.0

In this section we show how the flattening lemma (Lemma 3) enables us to use Gupta et al.'s proof, with minor changes, to get a distortion 9.0 bound for series-parallel graphs.

They prove the following three inequalities for any $\xi \in (\frac{1}{2}, 1)$.

(a) If $P_s^{i-1} \geq \xi$ then $\tilde{\mu}(x, s_i) \geq \tilde{\mu}(x, s_{i-1}) + (2\xi - 1)\alpha_i$.

(b) If $P_t^{i-1} \geq \xi$ then $\tilde{\mu}(x, s_i) \geq \tilde{\mu}(x, t_{i-1}) + (2\xi - 1)L_i$.

(c) Otherwise, $1 - \xi \leq P_s^{i-1} \leq \xi$ and $\tilde{\mu}(x, s_i) + \frac{2\xi}{1-\xi}(\Delta^i - \Delta^{i-1}) \geq \tilde{\mu}(x, s_{i-1}) + \alpha_i$.

The above three inequalities imply

$$\tilde{\mu}(x, s_i) + \frac{2\xi}{1-\xi}(\Delta^i - \Delta^{i-1}) \geq \min\{\tilde{\mu}(x, s_{i-1}) + (2\xi - 1)\alpha_i, \tilde{\mu}(x, t_{i-1}) + (2\xi - 1)L_i\} \tag{2.2}$$

The left hand side of the inequality, when accumulated over all values of i, generates a value not more than $(1 + \frac{2\xi}{1-\xi})\tilde{\mu}(x, s_k)$. The right hand side is a $(2\xi - 1)$ factor of some path from x to s_k. (At step i we choose either the edge with length α_i or the one with length L_i depending on the minimizing argument.) Hence, $\tilde{\mu}(x, s_k)$ is at least a factor $(2\xi - 1)/1 + \frac{2\xi}{1-\xi} = \frac{(2\xi-1)(1-\xi)}{1+\xi}$ of $\mu(x, s_k)$. Choosing $\xi = \sqrt{3} - 1$ to maximize this factor gives a distortion bound of at most 13.92.

Given flattened triangles, we can improve inequality (2.2) and obtain a better

distortion bound. If a triangle is decreasing, i.e. $L_i = L_{i-1} - \alpha_i$,

$$\begin{aligned}\tilde{\mu}(x, s_i) &= \Delta^i + P_t^i L_i \\ &= \Delta^{i-1} + P_t^{i-1} L_i + P_s^{i-1}\alpha_i \\ &= \Delta^{i-1} + P_t^{i-1}(L_i + \alpha_i) + (P_s^{i-1} - P_t^{i-1})\alpha_i \\ &= \tilde{\mu}(x, s_{i-1}) + (P_s^{i-1} - P_t^{i-1})\alpha_i\end{aligned}$$

Similarly, $\tilde{\mu}(x, s_i) = \tilde{\mu}(x, t_{i-1}) + (P_t^{i-1} - P_s^{i-1})L_i$. For a decreasing triangle, $P^s = 1$ in Equation (2.1). Thus, the probability of a move from s_{i-1} to t_i is 0, which means $P_t^i = P_t^{i-1}$ and $\Delta^i - \Delta^{i-1} = P_s^{i-1}\alpha_i$. Thus,

(a) if $P_t^{i-1} \geq \frac{2}{3}$ then $\tilde{\mu}(x, s_i) = \tilde{\mu}(x, t_{i-1}) + (P_t^{i-1} - P_s^{i-1})L_i \geq \tilde{\mu}(x, t_{i-1}) + \frac{L_i}{3}$,

(b) otherwise, $P_t^{i-1} < \frac{2}{3}$ and $\tilde{\mu}(x, s_i) + 2(\Delta^i - \Delta^{i-1}) = \tilde{\mu}(x, s_i) + 2P_s^{i-1}\alpha_i \geq \tilde{\mu}(x, s_{i-1}) + (3P_s^{i-1} - P_t^{i-1})\alpha_i \geq \tilde{\mu}(x, s_{i-1}) + \frac{\alpha_i}{3}$.

Together, these inequalities imply

$$\tilde{\mu}(x, s_i) + 2(\Delta^i - \Delta^{i-1}) \geq \min\left\{\tilde{\mu}(x, s_{i-1}) + \frac{\alpha_i}{3}, \tilde{\mu}(x, t_{i-1}) + \frac{L_i}{3}\right\}. \tag{2.3}$$

If the triangle is increasing, i.e. $L_i = L_{i-1} + \alpha_i$, then $\tilde{\mu}(x, s_i) = \tilde{\mu}(x, s_i) + \alpha_i$, which again implies inequality (2.3). Using inequality (2.3) in place of (2.2) in Gupta et al.'s proof gives us distortion at most 9.0.

2.5 Distortion 6.0

The inductive construction of the graph and equation (2.1) encourage us to express $\tilde{\mu}(s_{i-1}, y)$ in terms of $\tilde{\mu}(s_i, y)$ and $\tilde{\mu}(t_i, y)$, and to reverse the direction of induction used in Gupta et al.'s proof. Also, where the above approach derives an inequality based on a single vertex s_i (or, symmetrically, t_i), we find that a parameterized

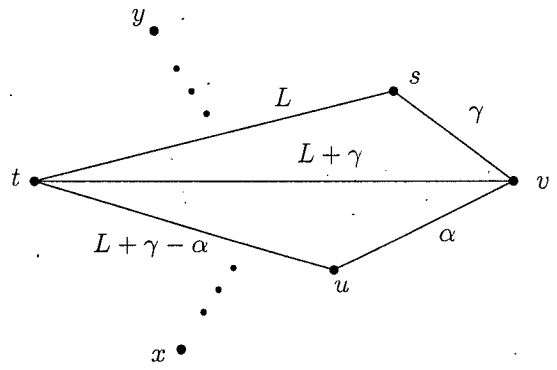

Figure 2.3: A pair of flattened triangles in a triangle sequence from x to an ancestor edge with endpoint y. Considering this case provides some intuition for our stronger, parameterized, inequality.

inequality based on an average of $\tilde{\mu}(s_i, y)$ and $\tilde{\mu}(t_i, y)$ leads to a better distortion bound. The price we pay for using this stronger inequality is a much more complicated inductive step. In fact, the details of the proof are five pages of dense mathematics. In this section, we give some intuition for the result but leave the details to Section 2.8.

Rather than a single flat triangle, consider the pair of triangles in Fig. 2.3. For that triangle pair,

$$\tilde{\mu}(u, y) = p\tilde{\mu}(v, y) + (1 - p)\tilde{\mu}(t, y) = p\tilde{\mu}(s, y) + (1 - p)\tilde{\mu}(t, y) + p\gamma$$

and

$$\mu(u, y) = \min\{\mu(s, y) + \alpha + \gamma, \mu(t, y) + L + \gamma - \alpha\}$$

where $p = \frac{L+\gamma-\alpha}{L+\gamma}$.

Any distortion inequality of the form $\tilde{\mu}(u, y) \geq c\mu(u, y)$ translates, using the above equations, into an inequality that has contribution from both s and t:

$$p\tilde{\mu}(s, y) + (1 - p)\tilde{\mu}(t, y) + p\gamma \geq c \min\{\mu(s, y) + \alpha + \gamma, \mu(t, y) + L + \gamma - \alpha\}.$$

By replacing α with the equivalent $(1-p)(L+\gamma)$ and moving $p\gamma$ to the right we obtain:

$$\begin{aligned} p\tilde{\mu}(s,y) + (1-p)\tilde{\mu}(t,y) \quad &\geq c\min\{ \quad \mu(s,y) + (1-p)L + (2-p-p/c)\gamma, \\ &\qquad \mu(t,y) + pL - p\gamma(1-1/c)\} \end{aligned}$$

When we include a Δ^i term on the left, this is very similar to the inequality (2.2) used by Gupta et al. Notice that we can view γ as a parameter and obtain all flattened triangle pairs. The distortion 6.0 result is based on inequality (2.4) by setting $c = \frac{1}{3}$, using Δ^i with coefficient one on the left side and replacing 2γ with β. We also add $2/3\min\{P_s^i, P_t^i\}L_i$ to the right side to make the induction work.

Let $\tilde{\mu}_s^i = \tilde{\mu}(s_i, y)$ and $\mu_s^i = \mu(s_i, y)$ for all i ($\tilde{\mu}_t^i$ and μ_t^i are defined similarly). Let $\Delta_i = \Delta - \Delta^i$, i.e., Δ_i is the expected sum of the δ's accumulated over all triangles starting from the edge (s_i, t_i) up to (s_k, t_k).

The precise form our intuition takes is

$$\Psi_s^i(\beta) = 1/3\min\{\mu_s^i + P_t^i L_i + \beta(1-2P_s^i), \mu_t^i + P_s^i L_i - P_s^i\beta\} + 2/3\min\{P_s^i, P_t^i\}L_i,$$

and

$$\Psi_t^i(\beta) = 1/3\min\{\mu_s^i + P_t^i L_i - P_t^i\beta, \mu_t^i + P_s^i L_i + \beta(1-2P_t^i)\} + 2/3\min\{P_s^i, P_t^i\}L_i.$$

Lemma 4. *For flattened triangle sequences, for all $0 \leq i \leq k$ and all $\beta \geq 0$,*

$$P_s^i\tilde{\mu}_s^i + P_t^i\tilde{\mu}_t^i + \Delta_i \geq \max\{\Psi_s^i(\beta), \Psi_t^i(\beta)\}$$

Proof. See Section 2.8. □

Let us derive the corollary that is used in proving that $\tilde{\mu}$ has distortion at most 6.0.

Corollary 2. *For flattened triangle sequences from x to an endpoint y of an ancestor edge of x,*

$$\tilde{\mu}_F(x,y) + \Delta_F \geq \frac{\mu(x,y)}{3}.$$

Proof. Since $P_s^0 = 1$, $\Delta = \Delta_0$, and $\mu_t^0 + L_0 \geq \mu_s^0 = \mu(x,y)$, we have

$$P_s^0\tilde{\mu}_s^0 + P_t^0\tilde{\mu}_t^0 + \Delta_0 = \tilde{\mu}(x,y) + \Delta \geq \Psi_s^0(0) = \Psi_t^0(0) \geq \frac{\mu(x,y)}{3}.$$

□

The following theorem shows that Gupta et al.'s construction produces an ℓ_1-embedding with distortion at most 6.0.

Theorem 1. *For any two vertices x and y, $\tilde{\mu}(x,y) \geq \frac{\mu(x,y)}{6}$.*

Proof. We have two cases.

Case 1: y lies on an ancestor edge of x.

In this case, by Lemma 3 and Corollary 2, $2\tilde{\mu}(x,y) \geq 2\tilde{\mu}_F(x,y) \geq \tilde{\mu}_F(x,y) + \Delta_F \geq \frac{\mu(x,y)}{3}$ which yields a distortion of 6.0.

Case 2: neither x nor y lies on an ancestor edge of the other.

The proof of this case is essentially the same as in Gupta et al. [28].

If x and y are not ancestors of each other then let (s,t) be the last edge added in the construction that is an ancestor edge of x and an ancestor edge of y. If (s,t) separates x and y (i.e. every path from x to y passes through s or t) then

$$\tilde{\mu}(x,y) = \Delta^x + \Delta^y + (P_s^x P_t^y + P_t^x P_s^y)L$$

where Δ^x (resp. Δ^y) is the expected sum of the δ's accumulated over all triangles starting from x (resp. y) to the edge (s,t), P_s^x (resp. P_s^y) is the probability that

x (resp. y) collapses to s when it moves to (s,t) (similarly for P_t^x and P_t^y), and $L = \mu(s,t)$. We know $P_s^x P_t^y + P_t^x P_s^y \geq \frac{1}{2}\min\{P_s^x + P_s^y, P_t^x + P_t^y\}$. Without loss of generality, suppose $P_s^x + P_s^y = \min\{P_s^x + P_s^y, P_t^x + P_t^y\}$. So,

$$\begin{aligned}
\tilde{\mu}(x,y) &= \Delta^x + \Delta^y + (P_s^x P_t^y + P_t^x P_s^y)L \\
&\geq \Delta^x + \Delta^y + \frac{P_s^x + P_s^y}{2} L \\
&= \frac{\Delta^x + \tilde{\mu}(x,t)}{2} + \frac{\Delta^y + \tilde{\mu}(y,t)}{2} \\
&\geq \frac{\mu(x,t) + \mu(y,t)}{6} && \text{by Corollaries 1 and 2} \\
&\geq \frac{\mu(x,y)}{6}.
\end{aligned}$$

If (s,t) doesn't separate x and y then there must be a vertex q whose parent edge is (s,t) with (s,q) an ancestor edge of x and (t,q) an ancestor edge of y. This case reduces to the previous case by noting that $\tilde{\mu}(x,y)$ is preserved by reordering the construction sequence in such a way that (s,q) becomes a common ancestor edge of x and y. (See Gupta et al. [28] for details.) Consequently, the distortion is at most 6.0. □

2.6 Lower bound

There are series-parallel graphs that cannot be embedded into ℓ_1 without some distortion. The best lower bound on this distortion, of which we are aware, is 3/2 and is obtained by showing that any ℓ_1-embedding of the unweighted bipartite graph $K_{2,n}$ (which is series-parallel) has distortion at least 3/2 [4].

If we restrict our embeddings to be those produced by Gupta et al.'s construction, we can prove a better lower bound. The following theorem shows that there exist outerplanar graphs whose ℓ_1 embedding via that construction incurs dis-

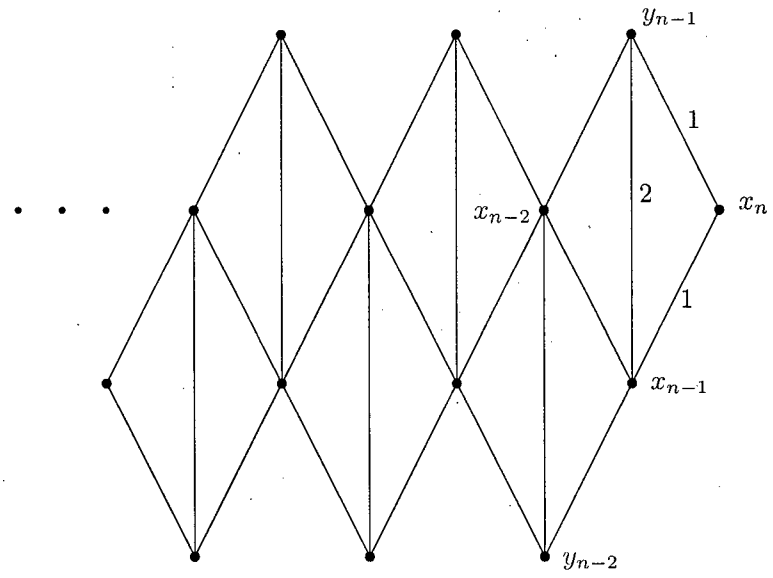

Figure 2.4: Lower bound example

tortion arbitrarily close to 3.0, even though outerplanar metrics are isometrically embeddable into ℓ_1.

Theorem 2. *There exists a family of outer-planar graphs whose ℓ_1 embedding, by the construction of Gupta et al., has distortion arbitrarily close to* 3.0.

Proof. Let G be defined as follows. $V(G) = \{x_i : 0 \leq i \leq n\} \cup \{y_i : 1 \leq i \leq n-1\}$ and $E(G) = \{(x_i, x_{i+1}) \text{ of length } 1 \ : 0 \leq i \leq n-1\} \cup \{(x_i, y_{i+1}) \text{ of length } 1 \ : 0 \leq i \leq n-2\} \cup \{(x_i, y_{i-1}) \text{ of length } 1 \ : 2 \leq i \leq n\} \cup \{(x_i, y_i) \text{ of length } 2 \ : 1 \leq i \leq n-1\}$. See Fig. 2.6.

Every shortest path in G from x_0 to x_n has length n. Assume that the first edge added in the incremental construction is (x_0, x_1). The construction order of vertices is fixed once this first edge is added; they are added in increasing order of their index, i.e. (x_0, x_1), y_1, x_2, y_2, $\cdots$. Letting $\tilde{\mu}_i = \tilde{\mu}(x_i, x_0)$, we have

$$\tilde{\mu}_n = \frac{\tilde{\mu}_{n-1} + \tilde{\mu}_{n-2} + 1}{2}$$

where $\tilde{\mu}_0 = 0$ and $\tilde{\mu}_1 = 1$. Let $g_n = \tilde{\mu}_n - \frac{n}{3}$, then,

$$g_n = \frac{g_{n-1} + g_{n-2}}{2}$$

where $g_0 = 0$ and $g_1 = \frac{2}{3}$. This implies that $0 \leq g_n \leq 2/3$ for all n, which means $\frac{n}{3} \leq \tilde{\mu}_n \leq \frac{n}{3} + 2/3$ or equivalently

$$\frac{1}{3} \leq \frac{\tilde{\mu}_n}{\mu_n} \leq \frac{1}{3} + \frac{2}{3n}.$$

As n increases, we obtain distortion arbitrarily close to 3.0.

If the order of incremental construction starts with the edge (x_j, v) for $v \in \{x_{j+1}, y_{j+1}, y_{j-1}, y_j\}$, then either $j \leq n/2$ or $j > n/2$ and the same argument, with $\tilde{\mu}_i = \tilde{\mu}(x_j, x_{j+i})$ or $\tilde{\mu}_i = \tilde{\mu}(x_j, x_{j-i})$, respectively, gives distortion approaching 3.0 as n approaches infinity. □

2.7 Conclusion and Future Work

In this chapter we provided a careful analysis of Gupta et al.'s construction and obtained an upper bound of 6.0 and a lower bound of 3.0 on the distortion of embedding series-parallel graphs into ℓ_1 using that construction.

One might also consider the problem of minimizing the dimension of the ℓ_1 metric as well as its distortion. Brinkman and Charikar [14] show that embedding certain series-parallel graphs (diamond graphs) into ℓ_1 with constant distortion requires $n^{\Omega(1)}$ dimensions. Whether low dimensional, constant distortion embeddings for all series-parallel graphs can be constructed efficiently is an open problem.

A very challenging problem is to embed bounded tree-width graphs into ℓ_1 with low distortion and, as a first step, graphs with tree-width 3.

2.8 Proof of Lemma 4

For flattened triangle sequences, for all $0 \leq i \leq k$ and all $\beta \geq 0$,

$$P_s^i \tilde{\mu}_s^i + P_t^i \tilde{\mu}_t^i + \Delta_i \geq \max\{\Psi_s^i(\beta), \Psi_t^i(\beta)\} \tag{2.4}$$

Proof. (of Lemma 4) The proof is by induction. We assume the lemma is true for $i, i+1, \ldots, k$ and show it is true for $i-1$.

We know

$$\Delta_{i-1} = \begin{cases} \Delta_i & \text{when } L_i = L_{i-1} + \alpha_i \text{ or } L_i = L_{i-1} + \beta_i \\ P_s^i \alpha_i + \Delta_i & \text{when } L_i = L_{i-1} - \alpha_i, \\ P_t^i \beta_i + \Delta_i & \text{when } L_i = L_{i-1} - \beta_i. \end{cases}$$

Since inequality (2.4) is symmetric in s_i and t_i, we may assume without loss of generality, that $t_i = t_{i-1}$ which means s_{i-1} collapses to (s_i, t_i).

The proof relies on four technical lemmas (Lemmas 5, 6, 7, and 8).

Case 1: (increasing triangle) $L_i = L_{i-1} + \alpha_i$.

Notice that in this case $\Delta_i = \Delta_{i-1}$ and $P_s^i = P_s^{i-1} \frac{L_{i-1}}{L_i}$.

$$\begin{aligned} P_s^{i-1} \tilde{\mu}_s^{i-1} + P_t^{i-1} \tilde{\mu}_t^{i-1} + \Delta_{i-1} &= P_s^{i-1} \left(\frac{L_{i-1}}{L_i} \tilde{\mu}_s^i + \frac{\alpha_i}{L_i} \tilde{\mu}_t^i\right) + P_t^{i-1} \tilde{\mu}_t^{i-1} + \Delta_{i-1} \\ &= P_s^i \tilde{\mu}_s^i + P_t^i \tilde{\mu}_t^i + \Delta_i \\ &\geq \Psi_s^i(\beta) \qquad \text{(by the induction hypothesis)} \\ &\geq \Psi_s^{i-1}(\beta) \qquad \text{(by Lemma 5).} \end{aligned}$$

We also need to show that $P_s^{i-1} \tilde{\mu}_s^{i-1} + P_t^{i-1} \tilde{\mu}_t^{i-1} + \Delta_{i-1} \geq \Psi_t^{i-1}(\beta)$.

If $P_t^{i-1} \geq \frac{1}{2}$ or $\mu_s^{i-1}+P_t^{i-1}L_{i-1} \leq \mu_t^{i-1}+P_s^{i-1}L_{i-1}$ then $\Psi_t^{i-1}(\beta) \leq \Psi_t^{i-1}(0)$ for all $\beta \geq 0$. In this case $P_s^{i-1}\tilde{\mu}_s^{i-1} + P_t^{i-1}\tilde{\mu}_t^{i-1} + \Delta_{i-1} \geq \Psi_s^{i-1}(0) = \Psi_t^{i-1}(0) \geq \Psi_t^{i-1}(\beta)$ for all $\beta \geq 0$ and we are done.

Otherwise, $P_t^{i-1} < \frac{1}{2}$ and $\mu_s^{i-1} + P_t^{i-1}L_{i-1} > \mu_t^{i-1} + P_s^{i-1}L_{i-1}$.

Define

$$B_i = \frac{\mu_s^i - \mu_t^i + (P_t^i - P_s^i)L_i}{P_s^i}. \tag{2.5}$$

This is the value of β that maximizes $\Psi_t^i(\beta)$ by making the two values in the first min of $\Psi_t^i(\beta)$ equal, i.e.,

$$\mu_s^i + P_t^i L_i - P_t^i B_i = \mu_t^i + P_s^i L_i + B_i(1 - 2P_t^i)$$

Note that $B_{i-1} > 0$ since $\mu_s^{i-1}+P_t^{i-1}L_{i-1} > \mu_t^{i-1}+P_s^{i-1}L_{i-1}$. Also note that $B_i > 0$, since

$$\begin{aligned}(\mu_s^i + P_t^i L_i) - (\mu_t^i + P_s^i L_i) \\ &\geq (\mu_s^{i-1} - \alpha_i + P_t^{i-1}L_{i-1} + \alpha_i) - (\mu_t^{i-1} + P_s^{i-1}L_{i-1}) \\ &= (\mu_s^{i-1} + P_t^{i-1}L_{i-1}) - (\mu_t^{i-1} + P_s^{i-1}L_{i-1})\end{aligned}$$

Hence

$$\begin{aligned}P_s^{i-1}\tilde{\mu}_s^{i-1} + P_t^{i-1}\tilde{\mu}_t^{i-1} + \Delta_{i-1} &= P_s^i\tilde{\mu}_s^i + P_t^i\tilde{\mu}_t^i + \Delta_i \\ &\geq \max\{\Psi_t^i(B_i), \Psi_t^i(0)\} \qquad \text{(by induction)} \\ &\geq \Psi_t^{i-1}(B_{i-1}) \qquad \text{(by Lemma 6)} \\ &\geq \Psi_t^{i-1}(\beta).\end{aligned}$$

Case 2: (decreasing triangle) $L_i = L_{i-1} - \alpha_i$.

In this case $P_s^i = P_s^{i-1}$, $\tilde{\mu}_s^{i-1} = \tilde{\mu}_s^i + \alpha_i$, $\mu_s^{i-1} = \mu_s^i + \alpha_i$, and $\Delta_{i-1} = \Delta_i + P_s^i \alpha_i$, so

$$\begin{aligned} P_s^{i-1}\tilde{\mu}_s^{i-1} + P_t^{i-1}\tilde{\mu}_t^{i-1} + \Delta_{i-1} &= P_s^i(\tilde{\mu}_s^i + \alpha_i) + P_t^i\tilde{\mu}_t^i + \Delta_i + P_s^i\alpha_i \\ &= P_s^i\tilde{\mu}_s^i + P_t^i\tilde{\mu}_t^i + \Delta_i + 2P_s^i\alpha_i \\ &\geq \Psi_s^i(\beta + 2\alpha_i) + 2P_s^i\alpha_i \qquad \text{(by induction)} \\ &\geq \Psi_s^{i-1}(\beta) \qquad \text{(by Lemma 7).} \end{aligned}$$

We also need to show that $P_s^{i-1}\tilde{\mu}_s^{i-1} + P_t^{i-1}\tilde{\mu}_t^{i-1} + \Delta_{i-1} \geq \Psi_t^{i-1}(\beta)$. In this case,

$$\begin{aligned} P_s^{i-1}\tilde{\mu}_s^{i-1} + P_t^{i-1}\tilde{\mu}_t^{i-1} + \Delta_{i-1} &= P_s^i\tilde{\mu}_s^i + P_t^i\tilde{\mu}_t^i + \Delta_i + 2P_s^i\alpha_i \\ &\geq \Psi_t^i(\max\{B_i, 0\}) + 2P_s^i\alpha_i \qquad \text{(by induction)} \\ &\geq \Psi_t^{i-1}(B_{i-1}) \qquad \text{(by Lemma 8)} \\ &\geq \Psi_t^{i-1}(\beta). \end{aligned}$$

Base Case

The only remaining step is to prove the base case. Assume $t_k = y$. We have $\mu_s^k = \tilde{\mu}_s^k = L_k$ and $\mu_t^k = \tilde{\mu}_t^k = 0$. Thus, $P_s^k\tilde{\mu}_s^k + P_t^k\tilde{\mu}_t^k + \Delta_k = P_s^k L_k$ and

$$\begin{aligned} \Psi_s^k(\beta) &= 1/3\min\{\mu_s^k + P_t^k L_i + \beta(1 - 2P_s^k), \mu_t^k + P_s^k L_i - P_s^k\beta\} \\ &\quad + 2/3\min\{P_s^k, P_t^k\}L_k \\ &\leq 1/3P_s^k L_k + 2/3P_s^k L_k \\ &= P_s^k L_k. \end{aligned}$$

As for the inequality $P_s^k\tilde{\mu}_s^k + P_t^k\tilde{\mu}_t^k + \Delta_k \geq \Psi_t^k(\beta)$, as mentioned earlier, if $P_t^k \geq \frac{1}{2}$ then we are done because $P_s^k\tilde{\mu}_s^k + P_t^k\tilde{\mu}_t^k + \Delta_k \geq \Psi_s^k(0) = \Psi_t^k(0) \geq \Psi_t^k(\beta)$. Otherwise, assume $P_t^k < \frac{1}{2}$. It's clear that $B_k = \frac{\mu_s^k - \mu_t^k + (P_t^k - P_s^k)L_k}{P_s^k} = \frac{2P_t^k}{P_s^k}L_k$ and

$$\begin{aligned}
\Psi_t^k(B_k) &= 1/3\left(\mu_t^k + P_s^k L_k + B_k(1-2P_t^k)\right) + 2/3\min\{P_s^k, P_t^k\}L_k \\
&= 1/3\left(P_s^k L_k + \frac{2P_t^k(1-2P_t^k)}{P_s^k}L_k\right) + 2/3P_t^k L_k \\
&= P_s^k L_k - \frac{2L_k(P_s^k - P_t^k)^2}{3P_s^k} \\
&\leq P_s^k L_k \\
&= P_s^k\tilde{\mu}_s^k + P_t^k\tilde{\mu}_t^k + \Delta_k
\end{aligned}$$

The proof is now complete. □

Lemma 5. *If $L_i = L_{i-1} + \alpha_i$ then for all $\beta \geq 0$*

$$\Psi_s^i(\beta) \geq \Psi_s^{i-1}(\beta).$$

Proof. Notice that since $L_i = L_{i-1} + \alpha_i$, $\Delta_i = \Delta_{i-1}$ and $P_s^i = P_s^{i-1}\frac{L_{i-1}}{L_i}$.

$$\begin{aligned}
\Psi_s^i(\beta) &= 1/3\min\{\mu_s^i + P_t^i L_i + \beta(1-2P_s^i), \mu_t^i + P_s^i L_i - P_s^i\beta\} \\
&\quad + 2/3\min\{P_s^i, P_t^i\}L_i \\
&\geq 1/3\min\{\mu_s^{i-1} + P_t^{i-1}L_{i-1} + \beta(1-2P_s^i), \mu_t^{i-1} + P_s^{i-1}L_{i-1} - P_s^i\beta\} \\
&\quad + 2/3\min\{P_s^i, P_t^i\}L_i
\end{aligned}$$

since $\mu_s^{i-1} \leq \alpha_i + \mu_s^i$, $P_t^i = \frac{P_t^{i-1}L_{i-1}+\alpha_i}{L_i}$, $t_i = t_{i-1}$, and $P_s^i = P_s^{i-1}\frac{L_{i-1}}{L_i}$

$$\begin{aligned}
&\geq 1/3\min\{\mu_s^{i-1} + P_t^{i-1}L_{i-1} + \beta(1-2P_s^{i-1}), \mu_t^{i-1} + P_s^{i-1}L_{i-1} - P_s^{i-1}\beta\} \\
&\quad + 2/3\min\{P_s^{i-1}, P_t^{i-1}\}L_{i-1}
\end{aligned}$$

since $P_s^i \leq P_s^{i-1}$

$$= \Psi_s^{i-1}(\beta).$$

□

Recall the definition of B_i from the equation 2.5.

Lemma 6. *If $L_i = L_{i-1} + \alpha_i$ and $B_{i-1} > 0$ then*

$$\max\{\Psi_t^i(B_i), \Psi_t^i(0)\} \geq \Psi_t^{i-1}(B_{i-1}).$$

Proof. If $P_t^i \leq 1/2$ then $\Psi_t^i(B_i) \geq \Psi_t^i(0)$ and

$$\begin{aligned}
\Psi_t^{i-1}(B_{i-1}) - \Psi_t^i(B_i) &= 1/3(\tilde{\mu}_t^{i-1} + P_s^{i-1}L_{i-1} + B_{i-1}(1-2P_t^{i-1}) + 2/3P_t^{i-1}L_{i-1} \\
&\quad -1/3(\tilde{\mu}_t^i + P_s^i L_i + B_i(1-2P_t^i)) - 2/3P_t^i L_i \\
&= 1/3B_{i-1}(1-2P_t^{i-1}) - 1/3B_i(1-2P_t^i) - 2/3\alpha_i \\
&= 1/3B_{i-1}(2P_s^{i-1}-1) - 1/3B_i(2P_s^i - 1) - 2/3\alpha_i \\
&= 1/3\frac{\mu_s^{i-1} - \mu_t^{i-1} + (P_t^{i-1} - P_s^{i-1})L_{i-1}}{P_s^{i-1}}(2P_s^{i-1}-1) \\
&\quad -1/3\frac{\mu_s^i - \mu_t^i + (P_t^i - P_s^i)L_i}{P_s^i}(2P_s^i - 1) - 2/3\alpha_i \\
&\leq 1/3\frac{\mu_s^{i-1} - \mu_t^{i-1} + (P_t^{i-1} - P_s^{i-1})L_{i-1}}{P_s^{i-1}}(2P_s^{i-1}-1) \\
&\quad -1/3\frac{\mu_s^{i-1} - \mu_t^{i-1} + (P_t^{i-1} - P_s^{i-1})L_{i-1}}{P_s^i}(2P_s^i - 1) - 2/3\alpha_i \\
&= -2/3\alpha_i \\
&\quad +1/3(\mu_s^{i-1} - \mu_t^{i-1} + (P_t^{i-1} - P_s^{i-1})L_{i-1})(\frac{1}{P_s^i} - \frac{1}{P_s^{i-1}}) \\
&\leq -2/3\alpha_i + 1/3(2P_t^{i-1}L_{i-1})(\frac{\alpha_i}{P_s^{i-1}L_{i-1}}) \\
&= -2/3\alpha_i \frac{P_s^{i-1} - P_t^{i-1}}{P_s^{i-1}} \\
&\leq 0
\end{aligned}$$

Otherwise, if $P_t^i > 1/2$ then $\Psi_t^i(0) > \Psi_t^i(B_i)$ and

$$\begin{aligned}
\Psi_t^{i-1}(B_{i-1}) - \Psi_t^i(0) &= 1/3(\tilde{\mu}_t^{i-1} + P_s^{i-1}L_{i-1} + B_{i-1}(1-2P_t^{i-1}) + 2/3P_t^{i-1}L_{i-1} \\
&\quad -1/3(\tilde{\mu}_t^i + P_s^i L_i) - 2/3P_s^i L_i \\
&= 1/3B_{i-1}(1-2P_t^{i-1}) + 2/3(P_t^{i-1} - P_s^{i-1})L_{i-1} \\
&= 1/3\frac{\mu_s^{i-1} - \mu_t^{i-1} + (P_t^{i-1} - P_s^{i-1})L_{i-1}}{P_s^{i-1}}(2P_s^{i-1} - 1) \\
&\quad +2/3(P_t^{i-1} - P_s^{i-1})L_{i-1} \\
&\leq 1/3\frac{2P_t^{i-1}L_{i-1}}{P_s^{i-1}}(2P_s^{i-1} - 1) + 2/3(P_t^{i-1} - P_s^{i-1})L_{i-1} \\
&\leq 2/3P_t^{i-1}L_{i-1}(2 - \frac{1}{P_s^{i-1}}) + 2/3(P_t^{i-1} - P_s^{i-1})L_{i-1} \\
&= -2/3\frac{(P_t^{i-1} - P_s^{i-1})^2 L_{i-1}}{P_s^{i-1}} \\
&\leq 0
\end{aligned}$$

□

Lemma 7. *If $L_i = L_{i-1} - \alpha_i$ then for all $\beta \geq 0$*

$$\Psi_s^i(\beta + 2\alpha_i) + 2P_s^i\alpha_i \geq \Psi_s^{i-1}(\beta).$$

Proof. Let $\beta' = \beta + 2\alpha_i$.

$$\begin{aligned}
\Psi_s^i(\beta') + 2P_s^i\alpha_i &= 1/3\min\{\mu_s^i + P_t^i L_i + \beta'(1-2P_s^i), \\
&\qquad \mu_t^i + P_s^i L_i - P_s^i\beta'\} + 2/3\min\{P_s^i, P_t^i\}L_i + 2P_s^i\alpha_i \\
&\geq 1/3\min\{\mu_s^i + P_t^i L_i + 3P_s^i\alpha_i + \beta'(1-2P_s^i), \\
&\qquad \mu_t^i + P_s^i L_i + 3P_s^i\alpha_i - P_s^i\beta'\} + 2/3\min\{P_s^i, P_t^i\}L_{i-1}
\end{aligned}$$

(since $2/3\min\{P_s^i, P_t^i\}L_i + P_s^i\alpha_i = 2/3\min\{P_s^i, P_t^i\}(L_{i-1} - \alpha_i) + P_s^i\alpha_i \geq$

$$2/3\min\{P_s^i, P_t^i\}L_{i-1} + 1/3P_s^i\alpha_i \geq 2/3\min\{P_s^i, P_t^i\}L_{i-1})$$

$$= 1/3\min\{\mu_s^{i-1} + P_t^i L_{i-1} - \alpha_i(2 - 4P_s^i) + \beta'(1 - 2P_s^i),$$
$$\mu_t^i + P_s^i L_{i-1} + 2P_s^i\alpha_i - P_s^i\beta'\} + 2/3\min\{P_s^i, P_t^i\}L_{i-1}$$

since $\tilde{\mu}_s^{i-1} = \tilde{\mu}_s^i + \alpha_i$ and $L_i = L_{i-1} - \alpha_i$

$$= 1/3\min\{\mu_s^{i-1} + P_t^{i-1}L_{i-1} + \beta(1 - 2P_s^{i-1}),$$
$$\mu_t^{i-1} + P_s^{i-1}L_{i-1} - P_s^{i-1}\beta\} + 2/3\min\{P_s^{i-1}, P_t^{i-1}\}L_{i-1}$$

since $P_s^i = P_s^{i-1}$ and $t_i = t_{i-1}$

$$= \Psi_s^{i-1}(\beta).$$

□

Recall the definition of B_i from the equation 2.5.

Lemma 8. *If $L_i = L_{i-1} - \alpha_i$, $P_t^i < 1/2$, and $B_{i-1} > 0$ then*

$$\Psi_t^i(\max\{0, B_i\}) \geq \Psi_t^{i-1}(B_{i-1}) - 2P_s^i\alpha_i.$$

Proof. If $B_i \geq 0$ then

$$\Psi_t^i(B_i) = 1/3(\mu_s^i + P_t^i L_i - P_t^i B_i) + 2/3\min\{P_s^i, P_t^i\}L_i$$
$$= 1/3\left(\mu_s^i + P_t^i L_i - \frac{P_t^i}{P_s^i}(\mu_s^i - \mu_t^i + (P_t^i - P_s^i)L_i)\right) + 2/3P_t^i L_i$$
$$= 1/3\left(\mu_s^{i-1} - \alpha_i + P_t^i(L_{i-1} - \alpha_i)\right)$$
$$- 1/3\left(\frac{P_t^i}{P_s^i}(\mu_s^{i-1} - \alpha_i - \mu_t^{i-1} + (P_t^i - P_s^i)(L_{i-1} - \alpha_i)\right)$$
$$+ 2/3P_t^i(L_{i-1} - \alpha_i)$$

since $\mu_s^i = \mu_s^{i-1} - \alpha_i$, $L_i = L_{i-1} - \alpha_i$, and $P_t^i = P_t^{i-1} < 1/2$

$$\begin{aligned}
&= \Psi_t^{i-1}(B_{i-1}) - 1/3\alpha_i(1 + 4P_t^i - \frac{P_t^i}{P_s^i}(1 + P_t^i)) \\
&= \Psi_t^{i-1}(B_{i-1}) - 2P_s^i\alpha_i + 1/3\alpha_i(9P_s^i - 4) + 2/3\alpha_i\frac{P_t^{i\,2}}{P_s^i} \\
&> \Psi_t^{i-1}(B_{i-1}) - 2P_s^i\alpha_i \qquad (\text{since } P_t^i < 1/2).
\end{aligned}$$

If $B_i < 0$ then $\mu_s^i + P_t^i L_i < \mu_t^i + P_s^i L_i$ and $\Psi_t^i(0) = 1/3(\mu_s^i + P_t^i L_i) + 2/3P_t^i L_i$. Since $B_{i-1} > 0$, B_i can't be too negative. In fact,

$$\begin{aligned}
B_i = \frac{\mu_s^i - \mu_t^i + (P_t^i - P_s^i)L_i}{P_s^i} &= \frac{\mu_s^{i-1} - \mu_t^{i-1} + (P_t^{i-1} - P_s^{i-1})L_{i-1} - 2P_t^{i-1}\alpha_i}{P_s^i} \\
&= B_{i-1} - \frac{2P_t^i\alpha_i}{P_s^i} > -\frac{2P_t^i\alpha_i}{P_s^i}
\end{aligned}$$

Thus,

$$\begin{aligned}
\Psi_t^i(0) &= 1/3(\mu_s^i + P_t^i L_i) + 2/3P_t^i L_i \\
&= \Psi_t^i(B_i) + 1/3P_t^i B_i \\
&= \Psi_t^{i-1}(B_{i-1}) - 1/3\alpha_i(1 + 4P_t^i - \frac{P_t^i}{P_s^i}(1 + P_t^i)) + 1/3P_t^i B_i \\
&\geq \Psi_t^{i-1}(B_{i-1}) - 1/3\alpha_i(1 + 4P_t^i - \frac{P_t^i}{P_s^i}(1 + P_t^i)) - 2/3\alpha_i\frac{P_t^{i\,2}}{P_s^i} \\
&= \Psi_t^{i-1}(B_{i-1}) - 2P_s^i\alpha_i + 1/3\alpha_i(9P_s^i - 4) \\
&> \Psi_t^{i-1}(B_{i-1}) - 2P_s^i\alpha_i \qquad (\text{since } P_t^i < 1/2).
\end{aligned}$$

□

Chapter 3

Embedding between Line Metrics

3.1 Introduction

In this chapter, we focus on computing an optimal embedding between two fixed line metrics. A line metric is a set of points on a real one-dimensional line with the distance between any pair of points being their ℓ_1 distance (any ℓ_k distance is equivalent for one-dimensional points).

As we mentioned earlier, Kenyon et al. [36] consider the problem of optimally embedding one fixed line metric into another fixed one. They propose a polynomial-time, dynamic programming based, algorithm that computes the optimal embedding if the distortion is less than $3 + 2\sqrt{2}$. To this aim, they show that any permutation that contains $(3, 1, 4, 2)$ (see Fig. 3.1) as a sub-permutation corresponds to an embedding with distortion at least $3 + 2\sqrt{2}$. All permutations that do not contain a $(3, 1, 4, 2)$ sub-permutation have a nice structure that allows finding the optimal such permutation using dynamic programming in polynomial time. It

is worth mentioning that the problem is recently proven to be NP-hard when the optimal distortion is unrestricted. Hall et al. [29] prove that if the optimal distortion δ is at least n^{ϵ}, for some constant ϵ, then the distortion is not even approximable with a factor $\delta^{1-\epsilon'}$, for any ϵ', unless P=NP.

In this chapter, we extend the result of Kenyon et al. [36] by considering a less restricted class of permutations called k-separable permutations. In particular, we improve the threshold value on distortion below which an optimal embedding can be found in polynomial time from $3+2\sqrt{2}$ to 13.602. We also study several interesting problems related to permutations such as forbidden permutations, pattern matching, and stack sorting.

We recently found that Kenyon et al. (in an extension of their conference paper [36]) and Chandran et al. [17] have also extended the 2-separable result to k-separable permutations. By considering 9-separable permutations, they obtain a polynomial time dynamic programming solution that works in those cases when the distortion is less than $5+2\sqrt{6} \simeq 9.90$.

We obtain (independently) the same result and show that the same technique can be used, by considering larger values of k, to find optimal embeddings when the distortion is less than 13.602. We (as well as they) show that the technique does not work for distortion greater than 13.928 no matter how large k is chosen to be.

The structure of this chapter is as follows. After introducing basic definitions, in Section 3.2, we prove, in Section 3.3, that the class of k-separable permutations have a finite set of forbidden permutations. Then, in Section 3.4, we propose a polynomial time algorithms for embedding between two fixed line metrics provided the optimal embedding permutation is k-separable. We also study some algorithmic and non-algorithmic related results such as computing separability. Sections 3.6 is

devoted to the problem of finding a k-separable permutation in a text permutation. We propose a polynomial time dynamic programming algorithm for the problem. In Section 3.7, we interpret k-separable permutations in terms of the way they are sortable using a queue. Finally, in Section 3.8, we address some open problems related to our work.

3.2 Preliminaries

Notice that we can view any embedding as a mapping from source points to destination points or, simply, as a permutation. Assume the optimal embedding between U and V is the permutation π. We specify a permutation π with the notation $(\pi(1), \pi(2), \cdots, \pi(n))$.

Permutation π_n of size n *contains* permutation π_k of size k, if there exist indices $l_1 < l_2 < \cdots < l_k$ such that for all $1 \leq i < j \leq k$, $\pi_k(i) < \pi_k(j)$ iff $\pi_n(l_i) < \pi_n(l_j)$. In this case, we refer to π_k as a *sub-permutation* of π_n. In particular, $\pi_n^{x,y}$ is the unique permutation of size $y - x + 1$ such that $\pi_n^{x,y}(i) < \pi_n^{x,y}(j)$ iff $\pi_n(i + 1 - x) < \pi_n(j + 1 - x)$.

A *nice interval* I in π is either a singleton or is a set of at least two consecutive numbers from 1 to n such that their mapping, via π, is still a set of consecutive numbers. For example, the permutation $(4, 3, 1, 2)$ contains several nice intervals: $[1, 2]$, $[3, 4]$, $[2, 4]$ and $[1, 4]$.

If the interval $[1, n]$ can be decomposed into a constant number of sub-intervals such that each sub-interval is mapped, via π, to a sub-interval in V and this property recursively holds for all sub-intervals, then we can use dynamic programming and find the optimal embedding. More formally, an interval I is *k-separable*, with respect to π, if either it has at most k points or it can be partitioned into

nice sub-intervals $I_1, I_2, \cdots, I_m$ $(1 < m \leq k)$ such that each I_i is k-separable. π is k-separable iff the interval $[1, n]$ is k-separable with respect to π. The *separability* of π is the minimum $k > 1$ such that π is k-separable.

For example, the permutation $\pi = (2, 4, 3, 6, 5, 1)$ is 3-separable. $I_1 = [1, 3]$, $I_2 = [4, 5]$, $I_3 = [6]$, and it is clear that I_1, I_2, and I_3 are 3-separable as well.

Every 3-separable permutation is 2-separable, since for any three nice sub-intervals that partition a permutation, two may be merged to form a nice sub-interval. Therefore, we don't have any permutation with separability 3. It's also easy to see that for $k \geq 4$, there exist permutations of size k with separability k. These permutation could be interpreted in a simpler way: they don't have any nice interval except the interval $[1, k]$. We refer to these special k-separable permutations as *non-separable* permutations.

The distortion incurred by a permutation π, denoted by $d(\pi)$, is the minimum distortion incurred by embedding any two line metrics U and V via π. For example, $d(\pi)$ for the permutation in Fig. 3.1 equals $3 + 2\sqrt{2}$ and happens when $[a, b, c, x, y, z] = [1, \sqrt{2}, 1, 1, \sqrt{2}, 1]$. As we see later, Theorem 5 states that $d(\pi)$ equals the largest eigenvalue of a 0-1 matrix corresponding to π.

Corresponding to every permutation π of size n, there exist three permutations π^0, π^1, and π^{-1} that are similar to π and incur the same distortion. For all i's, $\pi^0(i) = n + 1 - \pi(i)$, $\pi^1(\pi(i)) = i$, and $\pi^{-1}(i) = n + 1 - \pi^1(i)$. For example, if $\pi = (2, 4, 1, 3)$, $\pi^0 = \pi^1 = (3, 1, 4, 2)$ and $\pi^{-1} = \pi$. Throughout this chapter, we always assume that a permutation comes with all its four symmetric forms. For example, when we say 2-separable permutations avoid $\pi = (2, 4, 1, 3)$ we mean they avoid $(3, 1, 4, 2)$ as well.

Let Π_k be the set of all non-separable permutations of size k. Let d_k be the

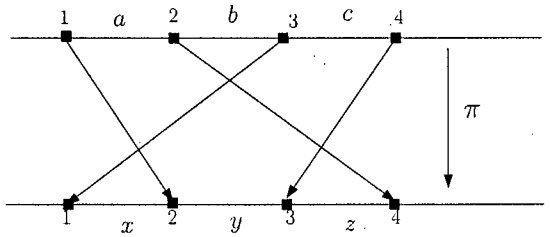

Figure 3.1: The 4-separable permutation $(2, 4, 1, 3)$.

minimum distortion over all permutations in Π_k. For example, $\Pi_4 = \{(2, 4, 1, 3)\}$, $\Pi_5 = \{(2, 4, 1, 5, 3), (2, 5, 3, 1, 4), (3, 5, 1, 4, 2)\}$, and it's not hard to see that $d_4 = d_5 = 3 + 2\sqrt{2}$. Note that by $\pi \in \Pi_k$ we implicitly mean $\pi^0, \pi^1, \pi^{-1} \in \Pi_k$ as well. So, $(3, 1, 5, 2, 4)$ is also in Π_5.

3.3 Forbidden Permutations

One commonly asked question regarding many permutation classes is whether they can be characterized by a finite forbidden set of permutations or not. For example, a permutation is 2-separable if and only if it contains neither $(2, 4, 1, 3)$ nor $(3, 1, 4, 2)$[12].

Interestingly one can generalize this statement for k-separable permutations.

Theorem 3. *[3] A permutation is k-separable if and only if*

- *For odd k, it doesn't contain any permutation in Π_{k+1}.*
- *For even k, it contains neither a permutation in Π_{k+1} nor π^*_{k+2}.*

where π^*_{2m} is the permutation of size $2m$ in which $\pi^*(2i)_{2m} = i$ and $\pi^*(2i-1)_{2m} = i + m$.

Albert and Atkinson [3] (See Theorem 4 in their paper) use the notion *simple*

for non-separable and call π^*_{2m} an exceptional permutation. They obtain their result by using some results from Schmerl and Trotter [49] on partially ordered sets.

3.4 Embedding between two line metrics

In this section we prove the following theorem which is a generalization of Kenyon et al.'s result.

Theorem 4. *For any two line metrics U and V and any k either the distortion of the optimal embedding between U and V is greater than d_{k+1} or there exists an $O(n^{5k+3})$ time algorithm for computing the optimal embedding.*

Recall that d_{k+1} is the minimum distortion over all permutations in Π_{k+1}. Let π be the optimal embedding permutation. If π is not k-separable then, according to Theorem 3, π contains either a permutation in Π_{k+1} or π^*_{k+2} (in case that k is even).

$d(\pi^*_{k+2})$ is increasing and one can easily see that $d(\pi^*_{12}) \simeq 21.954$[1]. Since $d_{k+1} \leq 7+4\sqrt{3}$, according to Theorem 6, we conclude that $d(\pi) \geq d_{k+1}$. Otherwise, if π is k-separable, an algorithm for finding the optimal embedding follows.

3.4.1 Algorithm

If the optimal embedding π is k-separable then we can compute it in time $O(n^{5k+3})$ by a dynamic programming approach. Every sub-problem is a mapping between two sub-intervals $I = \{u_i, u_{i+1}, \cdots, u_{i+m-1}\}$ and $J = \{v_j, v_{j+1}, \cdots, v_{i+m-1}\}$ that we specify by the triple (i, j, m). Moreover, we need to know the mappings of the four boundary points of both intervals for computing the distortion. Thus, each entry

[1] $\pi^*_{12} = (7,1,8,2,9,3,10,4,11,5,12,6)$. In fact, it's not had to prove that $d(\pi^*_{2k}) = 2k + 2\sqrt{k(k-1)} - 1$.

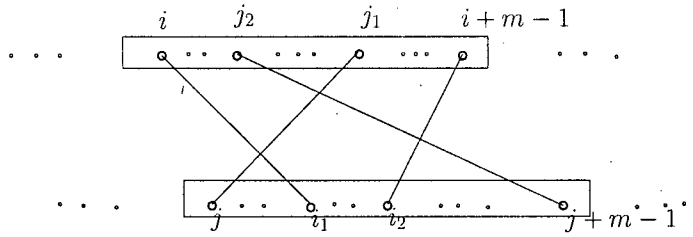

Figure 3.2: Algorithm.

in the dynamic programming table corresponds to 7 variables $(i, j, m, i_1, i_2, j_1, j_2)$ in which i and j are the start of both sub-intervals, m is the length of sub-intervals, $i_1 = \pi(i)$, $i_2 = \pi(i+m-1)$, $j_1 = \pi^{-1}(j)$, and $j_2 = \pi^{-1}(j+m-1)$. (See Fig. 3.2.)

Assume I is partitioned into at most k sub-intervals $I_1, I_2, \cdots, I_k$ such that each I_x is mapped to an interval in J. There are $O(n^{k-1})$ possibilities for partitioning I into I_x's. There are at most $k!$ possibilities for J_x's. (We assume that J_x is the mapping of I_x.) We also need to know the mapping for each boundary of I_x's and J_x's which has $O(n^{4k-4})$ possibilities. (The mapping for four of the boundary points is already known.) Once we have fixed all the sub-intervals and the mapping for all boundary points, we can compute the distortion by using the distortion corresponding to every sub-interval as well as the expansion and inverse expansion corresponding to single edges between boundaries of consecutive sub-intervals. In total, we need to consider $O(n^{5k-5})$ cases and it takes $O(n)$ to compute the distortion for each case. Since our dynamic programming table has $O(n^7)$ entries, the total running time would be $O(n^{5k+3})$.

3.4.2 Largest Eigenvalue

Assume the distortion corresponding to a permutation π of $[1, n]$ is λ. That means that for any two line metrics of n points each, the distortion using π is at least λ and

there exists a pair of line metrics whose distortion, using π, is exactly λ. In fact it is not hard to see that the maximum expansion and inverse expansion in embedding U to V happens for a pair of consecutive points, so we need to care only about them. Finding $d(\pi)$ corresponds to solving a set of linear equations. For example, for the permutation in Fig. 3.1, the linear equations are as follows.

$$\begin{aligned} y+z &\leq \sqrt{\lambda}a \\ x+y+z &\leq \sqrt{\lambda}b \\ x+y &\leq \sqrt{\lambda}c \\ a+b &\leq \sqrt{\lambda}x \\ a+b+c &\leq \sqrt{\lambda}y \\ b+c &\leq \sqrt{\lambda}z \end{aligned}$$

or equivalently $AX \leq \sqrt{\lambda}X$, where A is the adjacency matrix corresponding to π and X is $[a, b, c, x, y, z]^T$. In general, for a permutation π of size n that corresponds to embedding between two line metrics of size n, A has $2n-2$ rows and columns where, for all $0 \leq i, j < n$, $A[i, j] = A[n+i, n+j] = 0$ and $A[i, n+j] = A[n+i, j]$ is one iff the interval $[\pi(i), \pi(i+1)]$ (or $[\pi(i+1), \pi(i)]$ if $\pi(i) > \pi(i+1)$) contains the interval $[j, j+1]$ and is zero otherwise.

We can also assume that, by scaling edge weights in U or V if necessary, the expansion and contraction both equal $\sqrt{\lambda}$. Thus, for any single edge in U and V we write an inequality to make sure that its corresponding expansion does not exceed $\sqrt{\lambda}$.

Since we are interested in minimizing λ we better make the equality $AX = \sqrt{\lambda}X$. Therefore, $\sqrt{\lambda}$ is an eigenvalue of A. It is well know that ([30], Chapter 8.2.) the only eigenvalue whose corresponding eigenvector is positive is the largest

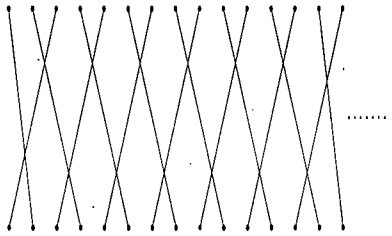

Figure 3.3: Illustration of permutation $\hat{\pi}_{15}$.

eigenvalue. Thus, $\sqrt{\lambda}$ is the largest eigenvalue of A.

Theorem 5. *Let A_π be the 0-1 matrix corresponding to π and let its largest eigenvalue be λ. Then, the distortion of π is exactly λ^2 and is obtained when the edge lengths are taken according to the eigenvector corresponding to λ.*

3.4.3 Bounding d_k

Although d_k is increasing in k, it remains bounded. This is somewhat disappointing since if it was unbounded we could imagine an algorithm that finds an optimal embedding for any two line metrics, no matter how large the optimal distortion is, whose running time is a function of the distortion.

Theorem 6. *For any value k there exists a non-separable permutation π_k whose distortion is at most $7 + 4\sqrt{3}$.*

Proof. Let $\hat{\pi}_{2n}$ be the permutation on $[1, 2n]$ where $\hat{\pi}_{2n}(1) = 2$, $\hat{\pi}_{2n}(2n) = 2n - 1$, $\hat{\pi}_{2n}(2i) = 2i + 2$, and $\hat{\pi}_{2n}(2i + 1) = 2i - 1$, for $i = 1, 2, \cdots, n - 1$. Similarly, $\hat{\pi}_{2n+1}$ is defined as follows. $\hat{\pi}_{2n+1}(i) = \hat{\pi}_{2n}(i)$, for $i = 1, 2, \cdots, n - 1$, $\hat{\pi}_{2n+1}(2n) = 2n + 1$, and $\hat{\pi}_{2n+1}(2n + 1) = 2n - 1$(See Fig. 3.3).

Set $d_U(2i - 1, 2i) = 1$, $d_U(2i, 2i + 1) = \sqrt{3}$, $d_V(2i - 1, 2i) = 2 + \sqrt{3}$, and $d_V(2i, 2i + 1) = 3 + 2\sqrt{3}$. The distortion corresponding to this pair of point sets is $7 + 4\sqrt{3}$ which means $d_k \leq 7 + 4\sqrt{3} \simeq 13.928$.

k	5	7	9	11	13	15	17	19
distortion	8.352	10.896	12.045	12.651	13.007	13.233	13.385	13.492

Table 3.1: Distortion of $\hat{\pi}_k$ for several values of k.

k	4	6	9	12	15	24
d_k	5.828	8.352	9.899	10.896	11.571	12.850
k	30	34	38	42	46	
d_k	13.131	13.316	13.443	13.534	13.602	

Table 3.2: d_k.

□

Table 3.1 shows the exact distortion of such permutations for small values of k. Finding d_k for small k's (by computing the eigenvalue corresponding to all permutations in Π_k and taking the minimum) suggests that d_k converges to $7+4\sqrt{3}$. Table 3.2 shows the value of d_k for different k's.

Consequently we improve the result of Kenyon et al. [36] from $3 + 2\sqrt{2} \simeq 5.828$ to 13.602.

Theorem 7. *There exists a polynomial algorithm for computing the optimal distortion embedding between two line metrics, provided the optimal distortion does not exceed* 13.602.

3.5 Computing Separability

Given a permutation π of size n one can compute its separability by the following greedy algorithm. Initially set $x = 1$. Find the largest nice interval $[x, y]$ (in case $x = 1$ don't choose $y = n$), set $x = y + 1$, and repeat. Recursively find the optimal separation for each nice interval.

Theorem 8. *The above greedy algorithm is correct.*

Proof. It suffices to show that the first step is correct. Assume I is the largest nice interval that contains x. Suppose an optimal algorithm OPT behaves differently; let $I_1, I_2, \cdots, I_k$ be all intervals returned by OPT that have non-empty intersection with I. Since $k \geq 2$, because of the maximality of I, we could take I and $I_k - I$ instead of those k intervals in OPT and get a better or equal separability consistent with our greedy algorithm. It is very easy to see that $I_k - I$ is a nice interval. □

3.6 Pattern matching for permutations

The question of finding whether a permutation contains another permutation is of interest for many people because of its applications. It is sometimes called *pattern matching* in the literature and comes as either a decision or a counting problem: Given two permutations σ and π decide if σ contains π or count the number of occurances of π in σ. The greedy algorithm for recognizing k-separable permutations in subsection 3.5 is a similar problem: Given a permutation π, does it avoid all permutations in $\Pi_{k+1} \cup \pi^*_{k+2}$?

Bose et al. [12] considered recognition of 2-separable permutations and proposed an efficient algorithm for both the decision and counting problem. They also show that the general decision problem is NP-Complete and the counting problem is $\#P$-Complete. An alternative way to recognize non-2-separable graphs, as pointed out in [12], is to consider the corresponding permutation graph. It is not hard to see that a permutation is non-2-separable iff its corresponding permutation graph is P_4-free, meaning that it does not have any induced path of length four. One can use the linear time algorithm in [21] to recognize P_4-free graphs. It doesn't seem

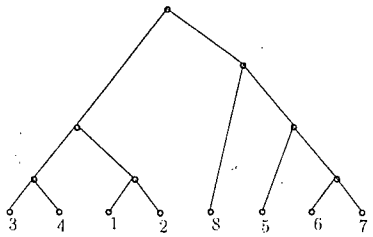

Figure 3.4: A separation tree.

to us that permutation graphs corresponding to k-separable graphs, for $k > 2$, have any particular structure.

In proposing the linear time algorithm, Bose et al. [12] introduce a special ordered binary tree called a *separation tree* corresponding to a permutation π of 1 to n with the following properties:

1. Leaves are $\pi(1), \pi(2), \cdots, \pi(k)$ in order.

2. For any node v, if the set of leaves of the subtree rooted at v is $\{\pi(i), \pi(i+1), \cdots, \pi(i+j)\}$ then $[i, i+j]$ must be a nice interval.

A separation tree corresponding to the permutation $(3, 4, 1, 2, 8, 5, 6, 7)$ is depicted in Fig. 3.4.

Similarly we can extend the definition of separation tree and allow it to be k-ary instead of binary. The resulting tree is equivalent to k-separable permutations.

Theorem 9. *A permutation π is k-separable iff it has a k-separation tree.*

Proof. Given a k-separable permutation π, one can easily build a k-separation tree. Assume $I_1, I_2, \cdots, I_k$ are the k k-separable nice intervals. Recursively build a k-separation tree for each I_j and then connect the roots of all these k trees to a new root.

For the reverse direction, assume a k-separation tree corresponding to π is given. It is obvious that the set of numbers in the subtree rooted at the j^{th} child is actually the j^{th} k-separable nice interval. □

Bose et al. [12] use the separation tree and obtain a dynamic programming algorithm to decide if any 2-separable permutation π is contained in a (larger) permutation σ.

A recursive problem instance in their approach is given a sub-permutation $\sigma' = \sigma^{i,j}$ of σ and any node u, that defines a subtree T_u, of the separation tree associated with π, decide if π_u is contained in σ', where π_u is the permutation corresponding to T_u.

Not surprisingly, we can use a similar dynamic programming approach and compute matchings for k-separable permutations.

Theorem 10. *Given a k-separable permutation π and a permutation σ of size n and m, respectively, one can compute the number of matchings of π in σ in time $O(mn^{k+2})$.*

Proof. Let $M(u, i, j)$ be the number of matchings of π_u in $\sigma' = \sigma^{i,j}$. To avoid double counting, let's assume that i is used in every matching, i.e for every matching $(i_1, i_2, \cdots, i_k)$, $\pi_u(i_x) = i$, for some x.

Assume that u has k children $u_1, u_2, \cdots, u_k$ in order of their appearance in π_u, i.e. every element in π_{u_x} is less than every element in $\pi_{u_{x+1}}$, for all x's. π_u is, in fact, partitioned into π_{u_x}'s. It can be easily proven that

$$M(u, i, j) = \sum_{(i_1, i_2, \cdots i_k)} \prod_{x=1,\cdots,k} M(u_x, i_x, i_{x+1} - 1)$$

(Assuming that $i_1 = i$ and $i_{k+1} = j+1$.) We basically partition $\sigma' = \sigma^{i,j}$ into k ranges $[i_x, i_{x+1} - 1]$ (for $x = 1, 2, \cdots, k-1$) and compute the number of matchings of each u_i into corresponding sub-permutation of T'. Computing $M(u, i, j)$ takes $O(n^{k-1})$ time and there are $O(mn^2)$ possibilities for u, i, j. Thus, a dynamic programming approach takes time $O(mn^{k+1})$ to compute $M(u, i, j)$ for all values of u, i, and j. Finally, the value $\sum_{i=1,\cdots,n} M(u_0, i, n)$ equals the number of matchings of π into T, where u_0 is the root of the corresponding separation tree. One can easily augment the algorithm to output the actual matching or list all the possible matchings as well. □

3.7 Sortable Permutations

Another interesting topic related to permutations is characterizing classes of permutations in terms of whether they are sortable using tools like stacks and queues.

The simplest versions of this problem were studied by Knuth[38] who imagined the elements of a permutation being an input sequence to a stack. A sequence of push and pop operations results in an output sequence (the popped elements) and the question is what input permutation can be sorted (yield an ordered output sequence). Tarjan[52], Even and Itai[23], and Pratt[42] generalized the model to allow multiple stacks and queues.

The answer to the simplest case is known: Single stack sortable permutations can be recognized in linear time, are characterized by the forbidden permutation $[2, 3, 1]$, and there are $\binom{2n}{n}/(n+1)$ (the n^{th} Catalan number) of them of length n. Other researchers have studied variations of stacks: Avis and Newborn[7] considered a less powerful stack called *pop-stack* in which PUSH operations are as usual, but POP operations, called *MPOP*, pop the entire stack. They provide enumeration

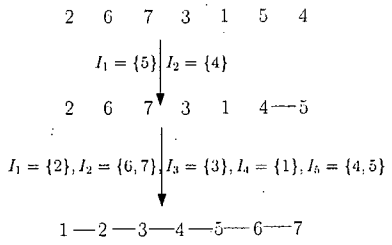

Figure 3.5: 5-sortable permutation ('-' indicates coupling)

answers when we use unlimited pop-stacks or we use a fixed number of pop-stacks in series.

Bose et al.[12] also interpreted 2-separable permutations as sortable permutations by the following device. The permutation is originally on a queue; each time we can pick a range of elements and reverse their order. Once we do so, all elements of that range are coupled and remain coupled forever. For other related results like parallel-stack sortable permutations the reader is referred to [5] or [11].

Here we give an interpretation of k-separable permutations in terms of sorting. Given a permutation π on a queue, we are allowed to do the following operations:

- Each time we can pick up to k consecutive ranges $I_1, I_2, \cdots, I_k$ of elements of π_k and sort them. Once we do so, The entire sub-range $I_1 \cup I_2 \cup \cdots \cup I_k$ gets coupled and remains coupled forever.
- We are not allowed to pick a portion of a coupled range, however, we can reverse the order of elements in a coupled range.

Let's call the above sorting mechanism *k-sorting*. A 5-sortable permutation is shown in Fig. 3.5.

Theorem 11.

k-separable permutations are exactly the class of permutations that are k-sortable.

Proof. It is obvious that any coupled range should be a set of consecutive numbers in 1 to n, for if not we won't be able to insert any element in a coupled range any more. That means there is a correspondence between k-sorting steps and nodes of a separation tree. Since we always pick at most k sub-ranges, the corresponding separation tree is a k-ary which is equivalent to k-separable permutations according to Theorem 9.

The other direction of the proof is simple. Given any k-separable permutation π, one can consider its separation tree and sort the permutation in a bottom-up fashion.

When we are at a node u, all its children are already sorted; since it has at most k children, we can swap the orderings of children to get a sorted permutation corresponding to the sub-tree rooted at u. □

3.8 Conclusions and Future Work

We considered the problem of finding a minimum distortion embedding of one fixed line metric into another fixed line metric. As a consequence, we studied properties of permutations under certain separability constraints, and discovered features of these permutations in various contexts: forbidden permutations, metric embedding, pattern matching, and stack sorting. The main open question that we would like to address is whether or not we can still find a parameterized solution for embedding two fixed line metrics. Notice that the problem is NP-hard [29] when the distortion is at least n^ϵ, for some constant ϵ, but is unresolved for smaller values of distortion. Although we proved that the idea of considering k-separable permutations does

not apply when the optimal distortion is greater than 13.602, there is still some hope. One may consider a different class of permutations that are algorithmically useful. Another important fact is that we are considering all permutations whereas only a few of them are a candidate to be an optimal embedding. It seems to us that excluding permutations that cannot be an optimal embedding from the set of k-separable permutations would be a major improvement to our work.

Another interesting problem is to look for parameterized approximate solutions. It appears that if we allow the optimal distortion to be approximated, we can easily avoid many permutations and only look for simple (possibly k-separable for small k) permutations.

Chapter 4

D-Width

4.1 Introduction

One of the most significant recent advances in the field of algorithmics comes from the Graph Minors project of Robertson and Seymour. In addition to being a major addition to the structure theory of graphs, the tools developed during their work imply algorithmic results such as every minor-closed graph property can be decided in polynomial time. The most far-reaching algorithmic contribution is the introduction of graph decompositions such as tree decompositions and measures such as tree-width, which have helped identify large classes of tractable instances of hard (e.g. NP-complete) graph problems.

The key to the algorithmic success of tree decompositions is that they are readily extendable to arbitrary relational structures. By considering tree decompositions of the background (or primal) graph, large classes of tractable instances of hard problems can be found for various structures including hypergraphs and directed graphs. The main drawback of this approach is that often information is lost when considering the background graph, and this may be crucial. For instance, the

background graph of a directed graph is the undirected graph obtained by forgetting edge orientations. Thus efficient solutions to problems like Hamiltonicity cannot be found by this technique.

In [34], Johnson et al. attempted to rectify this (and address problems in directed graph structure theory) by introducing directed tree-width, a generalization of tree-width to directed graphs. Although they managed to demonstrate the algorithmic benefits of arboreal decompositions by providing efficient algorithms for problems such as Hamiltonicity and disjoint paths, their measure was awkwardly defined and not as well behaved as tree-width, making it difficult to extend to further results. Consequently, other measures such as D-width [48], DAG-width [10, 40], and Kelly-width [31] have been introduced in an effort to find a more practical generalization of tree-width to directed graphs.

Also in [34], Johnson et al. introduced a graph searching game to partially characterize directed tree-width. The game, similar to one that Seymour and Thomas used to characterize tree-width [50], involves a robber who can run arbitrarily fast *in strongly connected components*, and a set of cops who attempt to capture the robber by blocking his escape routes and landing on him. Johnson et al. show that if G has directed tree-width $k-1$ then k cops can capture the robber in this game, and towards a converse, if k cops can capture the robber on G, then G has directed tree-width at most $3k-2$. In addition, they show that the number of cops required to capture a robber cop-monotonely (i.e. vacated vertices are never revisited by cops) is different from the number of cops required to capture a robber without this restriction, and if k cops have a winning strategy, then $3k-1$ cops have a robber-monotone (i.e. the set of vertices the robber can reach is non-increasing) winning strategy. Adler [1] further extended these results by showing

that robber-monotone and robber-non-monotone cop numbers do not coincide, and that the robber-monotone cop number and the directed tree-width also differ.

On undirected graphs, the equivalence of the cops and robber game and tree-width is critical to the importance of tree-width as a measure of graph complexity. On one hand, the game is a good indicator of structural properties of graphs. For example, acyclic graphs only require 2 cops to capture the robber, and the number of cops required does not increase under taking of minors. On the other hand, the equivalence of monotone and non-monotone strategies implies that the number of cops required can be efficiently computed. Without a clean correspondence with such games, it is difficult to establish similar results for directed tree-width, particularly results that can be used to efficiently compute the exact directed tree-width of a graph.

In this chapter, we further study D-width [48] and identify the class of digraphs with D-width one. We then study the game characterization of D-width and show that D-width is bounded above and below by the number of cops required in certain versions of the cop-monotone game. In particular, we obtain a non-trivial upper bound for D-width which is computable in polynomial time when that bound is constant.

We also compare various parameters and show that there exist arbitrarily big gaps between haven order, directed tree-width, and D-width.

The chapter is organized as follows. In Section 4.2 we formally define the cops and robber game and the concepts we use throughout the chapter. In Section 4.3 we prove the equivalence of D-width and directed tree-width when D-width is one and provide several algorithmic applications of directed one trees. Then, in Section 4.5 we compare D-width with other parameters such as haven order and directed tree-

width. In the final section, we obtain a non-trivial upper bound for D-width and propose an algorithm for computing that bound provided the bound is constant.

4.2 Definitions

In this chapter we assume all graphs are finite and directed unless otherwise stated. We use standard graph theory terminology, see for example [22].

4.2.1 D-width

We recall the definition of D-width as defined in [48].

Definition 3 (Strongly connected set). *A subset S of vertices of a digraph G is called a* strongly connected set *if $G[S]$, the subgraph induced by S on G, is strongly connected.*

Definition 4 (D-decompositions and D-width). *A* D-decomposition *of a directed graph G is a pair (T, W) where T is a tree and $W = \{W_t | t \in V(T)\}$ is a family of subsets of $V(G)$ such that for every strongly connected set $S \subseteq V(G)$:*

(D1) $T|_S := \{t \in T | W_t \cap S \neq \emptyset\} \neq \emptyset$, *and*

(D2) The subgraph of T with vertex set $T|_S$ and edges $\{e = (s,t) \in T | W_s \cap W_t \cap S \neq \emptyset\}$ forms a connected subtree of T. On the other hand, an edge is included only if both its end points contain same vertex u of S.

The width *of a D-decomposition (T, W) is the minimum k such that $|W_i| \leq k + 1$ for all $W_i \in W$. The* D-width *of a directed graph G is the minimum width over all D-decompositions of G.*

Figure 4.1 shows a digraph with D-width one, together with an optimal D-decomposition. One can verify the above condition for all strongly connected sets: $\{a\}$, $\{b\}$, $\{c\}$, $\{d\}$, $\{e\}$, $\{a,b,c\}$, $\{a,c,d,e\}$, $\{a,b,c,d\}$, and $\{a,b,c,d,e\}$.

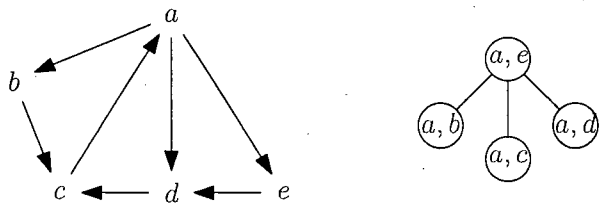

Figure 4.1: A digraph (left) with its D-decomposition of width one (right).

As D-decomposition is quite similar to tree-decomposition, it inherits all its structural properties that can be used for algorithmic purposes. For example, similar to the undirected variant [37], if a digraph has a D-decomposition of width w, then it has a *nice* D-decomposition of width w. A D-decomposition (T, W) is nice if every node $i \in V(T)$ has either one child or two. If it has one child j then $|W_i - W_j| = 1$. Otherwise, if it has two children j and k then $W_i = W_j = W_k$.

In addition a digraph G with D-width w has a related undirected chordal graph G' with tree-width w that captures the connectivity of G.

Lemma 9. *For every digraph G of D-width w, there exists an undirected chordal graph G' with treewidth w such that every strongly connected set in G is a connected set in G'.*

Proof. Let (T, W) be a D-decomposition of $G = (V, E)$ of width w. Let $G' = (V, E')$ where $E' = \{(u, v) | \exists t \text{ s.t. } u \in W_t \text{ and } v \in W_t\}$. In other words, every set of vertices W_t is a clique in G'. (T, W) is a tree-decomposition of G'. Moreover, every strongly connected set in G is a connected set in G'. □

D-width has also the balanced-separator property similar to tree-width.

Lemma 10. *For every digraph G of D-width w and any subset W, there exists a subset X of at most $w+1$ vertices such that every strongly connected component of $G\backslash X$ contains at most $\frac{|W|}{2}$ vertices from W.*

Proof. The proof is essentially similar to the undirected version. Given a D- decomposition T of G with width w, let i be the deepest node (pick an arbitrary node as the root) such that the sub-tree rooted at i has at least $\frac{|W|}{2}$ vertices from W. It's clear that every strongly connected component of $D\backslash W_i$ has then at most $\frac{|W|}{2}$ vertices from W. □

4.2.2 Directed tree-width and haven order

Here we introduce some relevant notation from [34]. Given two disjoint subsets Z and S of vertices of a digraph G, we say S is Z-normal if every directed path which starts and finishes in S is either wholly contained in S or contains a vertex in Z. Also, given a directed tree T with edges oriented away from a unique vertex $r \in V(T)$ (called the *root*), we write $t > e$ for $t \in V(T)$ and $e \in E(T)$ if e occurs on the unique directed path from r to t, and $e \sim t$ if e is incident with t. The following concepts were introduced in [34] and [35].

Definition 5 (Arboreal (pre-)decompositions and directed tree-width). *An* arboreal pre-decomposition *of a digraph G is a tuple (T, B, W) where T is a directed tree with a unique root, and $B = \{B_t | t \in V(T)\}$ and $W = \{W_e | e \in E(T)\}$ are sets of subsets of $V(G)$ that satisfy:*

(R1) B is a partition of $V(G)$ into (possibly empty) sets such that $B_r \neq \emptyset$ for the root r of T, and

(R2) If $e \in E(T)$, *then* $B_e^{\downarrow} := \bigcup\{B_t | t > e\}$ *is* W_e*-normal or empty.*

The width *of an arboreal pre-decomposition* (T, B, W) *is the minimum* k *such that for all* $t \in V(T)$, $|B_t \cup \bigcup_{e \sim t} W_e| \leq k + 1$. *An* arboreal decomposition *is a pre-decomposition in which all* B_t *are non-empty, and the* directed tree-width *of a digraph* G, $dtw(G)$, *is the minimal width of all its arboreal decompositions.*

If, in addition, an arboreal pre-decomposition satisfies:

(R3) For each $t \in V(T)$ we can order the outgoing edges $e_1, e_2, \ldots$ such that for $i < j$ there is no edge in G from $B_{e_j}^{\downarrow}$ to $B_{e_i}^{\downarrow}$

we call the decomposition *good.* In [35], Johnson et al. claim that an arboreal pre-decomposition can be transformed into a good one with the same width, but this does not follow from their results and remains an open problem. The importance of this problem, and indeed the motivation for [35], arises from the fact that the algorithmic results of [34] require a good arboreal decomposition. However, the decomposition constructed in the proof of Theorem 3.3 of [34] is good, implying that their algorithmic results do hold.

Our second definition is motivated by a similar definition in [50].

Definition 6 ((Pre-)haven and haven-width). *Let* G *be a digraph. A* pre-haven *of order* k *is a function* β *assigning to every set* $Z \subseteq V(G)$ *with* $|Z| < k$, *a union of strongly connected components of* $G \setminus Z$ *such that if* $X \subseteq Y \subseteq V(G)$ *and* $|Y| < k$ *then* $\beta(X)$ *is the union of all strongly connected components of* $G \setminus X$ *which intersect* $\beta(Y)$. *A* haven *is a pre-haven such that* $\beta(Z)$ *is a single strongly connected component of* $G \setminus Z$ *for all* $Z \subseteq V(G)$ *with* $|Z| < k$. *The* haven-width *of* G, $hw(G)$, *is the largest* k *such that* G *has a haven of order* k

In [50] it was shown that if an undirected graph G has a pre-haven of order k then it has a haven of order k. The analogous question for directed graphs remains an open problem.

4.2.3 Cops and robber game

We recall the definition of the cops and robber game defined in [34]. The game is played on a directed graph G, by two players: one controlling a set of k cops (k is a parameter of the game), the other controlling a visible robber. The cops and the robber occupy vertices in the graph. A move consists of the cop player announcing the next location of the cops and then proceeding to move the cops to this location. During this, the robber can run at great speed along directed paths which do not contain any cops not being moved *provided* there is also a cop-free directed path back to his original starting position. In other words, the robber may move to any vertex in the same strongly connected component of $G \setminus X$ where X is the set of locations occupied by stationary cops. If a cop lands on the position of the robber then the cop player wins, otherwise, if the robber is able to evade capture indefinitely, the robber player wins. More formally, the game consists of *positions* (X, R) where $X \subseteq V(G)$, $|X| \leq k$ and R is either empty, or a strongly connected component of $G \setminus X$. Initially the cop player chooses $X_0 \subseteq V(G)$ with $|X_0| \leq k$ and the robber player chooses a strongly connected component R_0 of $G \setminus X$ to give the initial position, (X_0, R_0). If $R \neq \emptyset$, a *move*, from position (X, R), consists of the cop player choosing $X' \subseteq V(G)$ with $|X'| \leq k$ and the robber player choosing R', a strongly connected component of $G \setminus X'$ such that R and R' are contained in the same strongly connected component of $G \setminus (X \cap X')$. If at any point the robber is unable to choose such a strongly connected component, then $R' = \emptyset$. This

gives the next position (X', R'). A *play* is a (possibly infinite) sequence of moves, and it is *winning for the cop player* if it is finite and has a final position $(X, \emptyset)$ for some X, otherwise it is *winning for the robber.* A play $(X_0, R_0), (X_1, R_1), \ldots$ is *cop-monotone* if the cops never revisit a vertex, that is for all h, i, j with $h < i < j$, $X_h \cap X_j \subseteq X_i$. The play is *robber-monotone* if $R_i \supseteq R_{i+1}$ for all i. For any digraph G, we denote the minimum number of cops that are required to capture the robber with a cop-monotone (resp. robber-monotone) strategy by *cop-monotone*(G) (resp. *robber-monotone*(G)).

As is usual for these types of games, we are primarily concerned with winning strategies. A *(k-cop) strategy* for the cop player is a tuple (Y_0, π) consisting of set $Y_0 \subseteq V(G)$ with $|Y_0| \leq k$ together with a function $\pi : \mathcal{P}(V(G)) \times \mathcal{P}(V(G)) \to \mathcal{P}(V(G))$, such that for $X \subseteq V(G)$ with $|X| \leq k$ if R is a strongly connected component of $G \setminus X$ then $|\pi(X, R)| \leq k$, and $\pi(X, \emptyset) = \emptyset$. A play $(X_0, R_0), (X_1, R_1), \ldots$ is *consistent* with a strategy (Y_0, π) if $X_0 = Y_0$ and $X_{i+1} = \pi(X_i, R_i)$ for all i, and a strategy is *winning* (*cop-monotone*, *robber-monotone*) if all consistent plays are winning for the cop player (cop-monotone, robber-monotone respectively).

Variants of the game where the robber moves first or only one cop can be moved at a time or the cops are lifted and placed in separate moves are all equivalent in that the existence of a winning strategy for a given number of cops does not depend on the variant.

For the results we present in the following sections, we introduce the idea of a *strategy forest.* Fix G and k, and consider the directed graph with nodes labeled by positions in the cops and robber game, and an edge from (X, R) to (X', R') if such a transition is possible under the rules specified above. That is, if $R \neq \emptyset$, and either $R' = \emptyset$ or R and R' are in the same strongly connected component of

$G \setminus (X \cap X')$. We call this the *positional graph defined by* G *and* k. A strategy $\sigma = (Y_0, \pi)$ will define a subgraph Π_σ of this graph, consisting of all roots of the form (Y_0, R), and edges $((X, R), (X', R'))$ if $X' = \pi(X, R)$. We also remove all edges of the form $(X, \emptyset)$. If σ is a winning robber-monotone strategy, Π_σ takes a very simple form.

Lemma 11. *If* $\sigma = (Y_0, \pi)$ *is a winning robber-monotone strategy, then* Π_σ *is a forest of rooted trees, with each root having a label of the form* $(Y_0, \_)$.

Proof. Since σ is a winning strategy, Π_σ is acyclic and all its roots are of the form $(Y_0, \_)$. To show that it is a forest, we need only show that no node has more than one predecessor. Suppose (X', R') has two predecessors. These two predecessors either have a common ancestor (X, R) with two distinct children $(\pi(X, R), R_1)$ and $(\pi(X, R), R_2)$ or are descended from two distinct roots (Y_0, R_1) and (Y_0, R_2) such that $R_1 \cap R_2 \neq \emptyset$. (By robber-monotonicity, the non-empty R' is a subset of $R_1 \cap R_2$.) But R_1 and R_2 are strongly connected components of $G \setminus \pi(X, R)$ (or $G \setminus Y_0$), so $R_1 = R_2$, contradicting the distinctness of the nodes. □

We call Π_σ the *strategy forest* associated with σ.

4.3 Directed One Trees

Currently there is no known polytime recognition algorithm for bounded D-width digraphs. For the special case that D-width is one, however, there is a fast recognition algorithm based on a structural characterization of such digraphs. Moreover, we prove that directed tree-width and D-width coincide in this case. This result is achieved by comparing both measures to the haven order.

First, we prove the following theorem relating haven order and D-width of directed one-trees.

Theorem 12. *A digraph G has D-width one if and only if it has haven order two.*

Proof. If G has a haven β of order at least 3 then the robber can win against two cops by staying at $\beta(X)$ where X is the position of cops. Hence, by Theorem 17, G must have D-width at least two. Since this is not the case, G has haven order at most two. But, the haven order of G cannot be one because G has a cycle (otherwise its D-width would be zero) and, thus, G has haven order at least two. (Simply set $\beta(\emptyset) = C$ and let $\beta(\{x\})$ be a strongly connected component the contains a vertex of $C - \{x\}$, where C is a cycle in G.) Consequently G has haven order exactly two.

Next, we show if G has haven order two then its D-width is one. It suffices to prove this for strongly connected G since if G has haven order two and D-width d, it contains a strongly connected component with haven order two and D-width d.

The proof is by induction on the number of vertices of G. By Lemma 12, G contains a vertex u with out-degree (or in-degree) one. Suppose u has out-degree one (the in-degree one case is handled similarly) with edge (u, v) being its only outgoing edge. Contract the edge (u, v), by removing u and connecting all u's incoming edges to v (and ignoring loops if created), to obtain a new digraph G'. By Lemma 13, G' has haven order at most two and, according to the induction hypothesis, has D-width at most one[1]. Let T' be a D-decomposition of G' with width at most one. Add a new node r to T' with $W_r = \{u, v\}$ and attach it to a node of T' that contains v. It is easy to verify that the resulting D-decomposition is a proper one for G and has D-width one because if S is a strongly connected set in G and $u \in S$ then $S - \{u\}$

[1]If G' has haven order one then it is acyclic and trivially has D-width zero.

is a strongly connected set in G' and $v \in S$. □

Lemma 12. *If G is strongly connected and has haven order two then G contains a vertex with in-degree or out-degree one.*

Proof. For any vertex u, a strongly connected component C of $G \setminus \{u\}$ is called a *u-root* if there is no edge from a vertex in another component of $G \setminus \{u\}$ to a vertex in C. Similarly we say a component C is a *u-leaf* if there is no edge from C to any other component. Let *rootleaf(u)* be the minimum size over all u-root and u-leaf components of $G \setminus \{u\}$.

If rootleaf$(u) = 1$ for some vertex u, then there is a single vertex v with either out-degree or in-degree one (to or from u). Otherwise, rootleaf(u) is at least two, for all u.

In this case, we show that G has haven order at least three, a contradiction. Let u be the vertex with minimum rootleaf(u) and C_u be the component that minimizes rootleaf(u), i.e. $|C_u| =$ rootleaf(u). Assume, without loss of generality, that C_u is a u-root component. Notice that such components do exist as the graph whose vertices are strongly connected components of $G\setminus\{u\}$ and whose edges are $\{(A, B) | \exists u \in A, \exists v \in B \text{ s.t. } (u, v) \in G\}$ is acyclic. It's roots are u-roots and its leaves are u-leaves.

Let $\beta(x)$[2], for any single vertex x, be the strongly connected component of $G\setminus\{x\}$ that contains C_u, if $x \notin C_u$, and the strongly connected component of $G\setminus\{x\}$ that contains u, otherwise. Let $\beta(\{x, y\})$ be the strongly connected component of $G \setminus \{x, y\}$ that contains $\beta(x) \cap \beta(y)$. We argue that β is a haven of order three.

It is sufficient to show that $\beta(x) \cap \beta(y) \neq \emptyset$ for all x and y. If x and y are both in C_u, then both $\beta(x)$ and $\beta(y)$ have vertex u in common. Similarly, if both

[2]In what follows we use $\beta(x)$ for $\beta(\{x\})$.

are in $G \setminus C_u$, then both $\beta(x)$ and $\beta(y)$ share C_u.

For the final case, $x \in C_u$ and $y \in G \setminus C_u$, it suffices to show that $\beta(x)$ contains at least one vertex from C_u. Let $S_1, S_2, \ldots S_k$ be the strongly connected components of $G \setminus \{x\}$ that contain at least one vertex from C_u, in topological order. (At least one such component must exist because $|C_u| \geq 2$.) If any S_i contains u then we're done. Otherwise, each S_i contains only vertices from C_u because every path from $v \in G \setminus C_u$ to a vertex in C_u contains u (a consequence of C_u being a u-root). Thus, $|S_i| < |C_u|$. We show that some S_i is an x-root or x-leaf component. This is a contradiction since C_u is supposedly the smallest such component.

For all $y \in C_u \setminus \{x\}$, there exists a path from u to y that contains only vertices in $C_u \setminus \{x\}$ (in particular, that doesn't contain x). If not, then the first component S_i (smallest i) that contains such a y is an x-root, a contradiction.

Since C_u is a u-root and $x \in C_u$, for all $z \in G \setminus C_u$, there exists a path from z to u that doesn't contain x. Hence S_k cannot have an outgoing edge (y, z) to a vertex $z \in G \setminus C_u$, otherwise y, z, and u would be strongly connected via paths that don't contain x. This implies that S_k is an x-leaf, a contradiction. □

Lemma 13. *If G' is a digraph obtained by contracting an edge (u, v), with u having out-degree one or v having in-degree one, in a digraph G then $H(G') \leq H(G)$, where $H(G)$ is the haven order of G.*

Note: The same statement regarding directed tree-width of G and G' was noted by Johnson et al. [34].

Proof. Let β' be a haven of order w for G'. We construct a haven, β, of order w for G. First, assume u has out-degree one. For any subset Z of vertices in G, if $u \in Z$, let $U(Z) = (Z - \{u\}) \cup \{v\}$, otherwise let $U(Z) = Z$. For Z with $|Z| < w$,

let $\beta(Z)$ be the strongly connected component of G that contains $\beta'(U(Z))$. (Note: $|U(Z)| \leq |Z|$ so $\beta'(U(Z))$ is defined.) If C is a strongly connected component of $G' \setminus Z$ for some Z and $u \notin Z$ then either C or $C \cup \{u\}$ is a strongly connected component of $G \setminus Z$. Thus, $\beta(Z)$ equals either $\beta'(U(Z))$ or $\beta'(U(Z)) \cup \{u\}$. Therefore, for any two subsets $Z_1 \subset Z_2$ of less than w vertices of G, $U(Z_1) \subseteq U(Z_2)$, so $\beta(Z_1) \cap \beta(Z_2) \supseteq \beta'(U(Z_1)) \cap \beta'(U(Z_2)) = \beta'(U(Z_2)) \neq \emptyset$. Thus, $\beta(Z_2) \subseteq \beta(Z_1)$. Notice that if C_1 and C_2 are two strongly connected componets of $G \setminus Z_1$ and $G \setminus Z_2$, respectively, then either $C_2 \subseteq C_1$ or $C_1 \cap C_2 = \emptyset$. The case when v has in-degree one is similar. □

Corollary 3. *The three statements "G has D-width one", "G has directed tree-width one", and "G has haven order two" are equivalent.*

Proof. This follows from 12 and the following two theorems from [34] and [47].

Theorem 13 (Johnson et al. [34]). $H(G) - 1 \leq$ *tree-width*$(G) \leq 3H(G) + 1$ *for digraphs* G*, where* $H(G)$ *is the haven order of* G*.*

Theorem 14 (Safari [47]). *For any digraph* G*, tree-width*$(G) \leq$ *D-width*(G)*.*

□

4.3.1 Algorithmic results

The nice property of directed one-trees is that they have a contractible edge as per Lemma 12. We can use this property to design recursive algorithms for certain problems on directed one-trees.

In the following algorithms, we assume that the contractible edge is (u, v) (with u having out-degree one). We contract the edge by removing u and con-

necting all u's incoming edges to v. The case when u has in-degree one is handled analogously.

Recognition To find a D-decomposition with width one, if it exists, in a digraph G:

0. If G is a single vertex return a single node containing the vertex.

1. Find a contractible edge (u, v), contract it, and obtain a new digraph G'. If no contractible edge exists then FAIL.

2. Recursively find a D-decomposition T' for G'.

3. Look for a node of T' that contains v, and add a new node r to it with $W_r = \{u, v\}$

If we keep the list of vertices ordered by in-degree and also by out-degree, we can perform steps 1, 2, and 3 in $O(n)$ time. Thus, the total running time is $O(n^2)$.

Hamiltonian cycle To find a Hamiltonian cycle, if it exists, in a directed one-tree G in $O(n^2)$ time:

0. If G is a single vertex return the vertex. If G is acyclic then FAIL.

1. Find a contractible edge (u, v), contract it, and obtain a new digraph G'. Also remove all edges (x, v) in G' where (x, v) is an edge in G. If no contractible edge exists then FAIL.

2. Recursively find a Hamiltonian cycle C in G'.

3. Replace the edge (x, v) in C with $(x, u), (u, v)$ to obtain a Hamiltonian cycle for G'.

4.4 Comparing D-width and directed tree-width

It is conjectured in [48] that D-width and directed tree-width are equal. We disprove this conjecture in this section and prove that there is an arbitrarily gap between D-width and directed tree-width, though it is still unknown whether the two are within a constant factor of each other. We will also provide several inequivalence results for other relevant parameters such as haven order.

To this aim, we consider game characterizations of D-width and directed tree-width.

Theorem 15. *For every digraph G,*

$$\textit{tree-width}(G) \leq \textit{robber-monotone}(G) \leq \textit{cop-monotone}(G) \leq \textit{D-width}(G)$$

Proof. $\textbf{\textit{tree-width}}(G) \leq \textbf{\textit{robber-monotone}}(G)$

It is proven in [34].

$\textbf{\textit{robber-monotone}}(G) \leq \textbf{\textit{cop-monotone}}(G)$

Let $\sigma = (Y_0, \pi)$ be a cop-monotone winning strategy, then we claim σ is a robber-monotone winning strategy. Assume not; we show the robber can defeat σ by moving to a vacated vertex, contradicting the assumption that σ is winning. Let $p = (X_0, R_0), (X_1, R_1), \ldots$ be a play consistent with σ such that $R_i \not\supseteq R_{i+1}$ for some i. From the definition of a play, it follows there exists $r \in R_{i+1}$ such that $r \in X_i \setminus X_{i+1}$. Let $p' = (X'_0, R'_0), (X'_1, R'_1), \ldots$ be a play consistent with σ that agrees with p up to (X_{i+1}, R_{i+1}), such that $r \in R'_j$ for all $j > i$. Note that since $r \in R_{i+1}$ and σ is cop-monotone, such a play exists as there will always be a strongly connected component containing r. But then this play is not winning for the cop player.

cop-monotone$(G) \leq$ ***D-width***(G)

Assume a D-decomposition (T, W) of G of width k is given. Let T be rooted at a node r. The cops can start at $X_0 = W_r$. Let $T_1, T_2, \cdots, T_m$ be subtrees of T with roots $r_1, r_2, \cdots, r_m$, children of r. Let U_i be the union of the sets W_j over all nodes j in T_i. According to conditions of D-decompositions, the robber can only be at vertices in one of the sets U_i. The cops can move to W_{r_i} and continue the strategy until they trap the robber in one of the leaves. The connectivity condition of D-decompositions ensures that this strategy will be strongly cop-monotone. □

4.4.1 Arbitrary gap between different games

We first observe that there can be an arbitrarily big gap between the number of cops required to win by using different strategies in the cop/robber game.

Theorem 16. *For any $m \in \mathbb{N}$ there exists:*

1. *A digraph on which $4m$ cops can capture a robber, but $5m$ cops are required to capture it with a robber-monotone strategy.*
2. *A digraph on which $4m$ cops can capture a robber with a robber-monotone strategy, but $5m$ cops are required to capture it with a cop-monotone strategy.*

Before we prove Theorem 16, we need to introduce the notion of lexicographic product.

Definition 7. *Let G and H be digraphs. The* lexicographic product, *$G \bullet H$, is the graph with $V(G \bullet H) = V(G) \times V(H)$ and*

$$E(G \bullet H) = \{((x, y), (x', y')) | (x, x') \in E(G), \text{ or } x = x' \text{ and } (y, y') \in E(H)\}.$$

The proof relies on the following result, similar to one presented in [32].

Lemma 14. *Let G be a digraph, K_m the complete digraph on m vertices. At least k cops have a (cop-monotone, robber-monotone) winning strategy on G, if and only if at least mk cops have a (cop-monotone, robber-monotone respectively) winning strategy on $G \bullet K_m$.*

Proof. If k cops have a winning strategy on G, then a winning strategy for mk cops on $G \bullet K_m$ is obtained by simulating the game on G. If the robber's position is $(r, s) \in V(G \bullet K_m)$ then we position a robber on $r \in V(G)$. We then consider the cops' play on G and play on $G \bullet K_m$ by placing n cops on $\{(x, y)|y \in V(K_m)\}$ whenever a cop would be placed on $x \in V(G)$. It's easy to verify that the cop-monotonicity and robber-monotonicity of the strategies do not change.

For the converse we show that if the robber can defeat $k-1$ cops on G then he can defeat $mk-1$ cops on $G \bullet K_m$. Again we simulate the game for $G \bullet K_m$ on G, but this time from the robber's perspective. We place a cop on $x \in V(G)$ only if all vertices in $V(G \bullet K_m)$ of the form (x, y), $y \in V(K_m)$ are occupied. By the pigeon-hole principle, this requires at most $k-1$ cops on G. The robber's current position is projected as before. The robber's response r' on G is lifted to $G \bullet K_m$ by playing to a non-occupied vertex of the form (r', y). As r' is unoccupied in the simulated game, at least one such vertex exists. As the robber can defeat $k-1$ cops on G, the strategy is winning. To complete the proof we need to show that if a strategy is not (cop-monotone, robber-monotone) on G then its corresponding strategy on $G \bullet K_m$ is not (cop-monotone, robber-monotone respectively) either. The idea is that if the robbers play according to the above strategy then the cops either need to occupy all vertices of type (x, r), for any x, in $G \bullet K_m$ or none of them. Partially filling these vertices doesn't impose any constraint on the robber's movement and, hence, is not useful. It's now very easy to verify that if such a

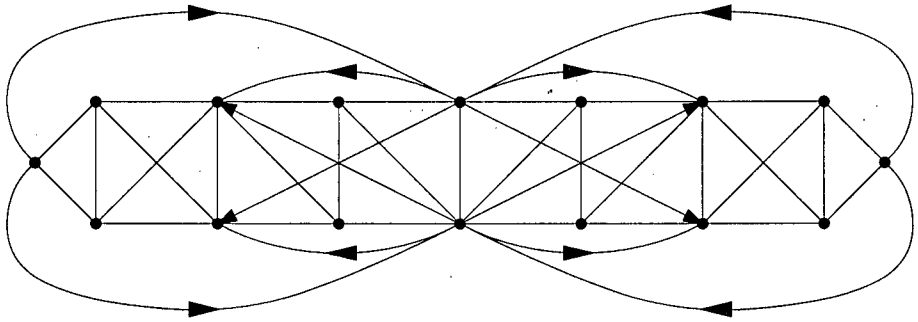

Figure 4.2: Graph on which 4 cops have a winning strategy but 5 cops are required for robber-monotone strategy.

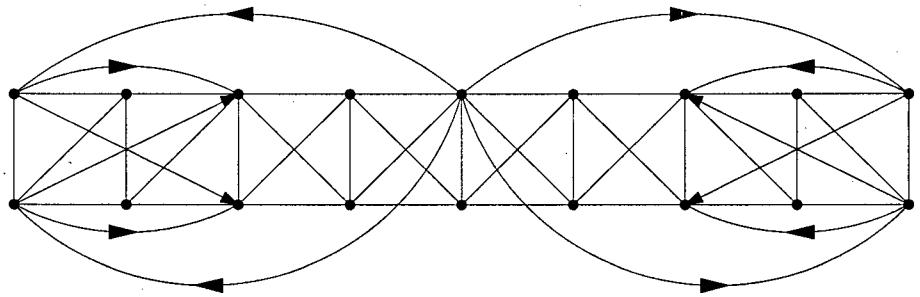

Figure 4.3: Graph on which 4 cops have a robber-monotone winning strategy but 5 cops are required for cop-monotone strategy.

strategy is (cop-monotone, robber-monotone) on $G \bullet K_m$ then it's corresponding strategy on G is (cop-monotone, robber-monotone respectively). □

We now turn to the proof of Theorem 16.

Proof. In [1] it was shown that 4 cops have a winning strategy on the graph in Figure 4.2, but 5 cops are required to capture the robber with a robber-monotone strategy. In [34] it was observed that 4 cops have a robber-monotone winning strategy on the graph in Figure 4.3, but 5 cops are required to capture the robber with a cop-monotone strategy. The results then follow by taking the lexicographic product of these graphs with K_m, the complete graph on m vertices. □

4.4.2 Arbitrary gap between D-width, directed tree-width, and haven order

Theorems 15 and 16 immediately yield an arbitrary big gap between directed tree-width and D-width. In this section we study the behavior of D-width, directed tree-width and haven order under lexicographic product and independently prove the existence of an arbitrary big gap for the above three parameters.

We first prove that D-width behaves well under lexicographic product.

Lemma 15. *If G is a digraph, and K_m the complete graph on m vertices, then* $1 + \textit{D-width}(G \bullet K_m) = m \cdot (1 + \textit{D-width}(G))$.

Proof. One can view $G \bullet K_m$ as making a clique $\{u_1, u_2, \cdots, u_m\}$ out of every vertex u of G and connecting u_i to v_j if and only if $(u, v) \in E(G)$. Let (T, W) be a D-decomposition of width w of G. We construct a D-decomposition (T, W') of $G \bullet K_m$ as follows. W' is a family of subsets of vertices of $G \bullet K_m$ such that for any j and i, $u_j \in W'_i$ if and only if $u \in W_i$. It can be easily proven that (T, W') is a proper D-decomposition of $G \bullet K_m$ and has width $m(w+1) - 1$. For the reverse direction, let (T', W') be a D-decomposition of $G \bullet K_m$ of width w' such that $\sum_{i \in T'} |W'_i|$ is minimized. We first make the following observation.

Claim 1. *For every $u \in G$ and $i \in T$, either $U \subset W'_i$ or $U \cap W'_i = \emptyset$, where* $U = \{u_1, u_2, \cdots, u_m\}$.

Proof. As U is a clique in $G \bullet K_m$, there must be a node j with $U \subset W'_j$. Let i be the furthest node from j that violates the above condition, i.e. there are $u_p, u_q \in U$ with $u_p \in W'_i$ and $u_q \notin W'_i$. Let k be the node before i in the unique path from j to i in T'. We claim that dropping u_p from W'_i still leaves a proper D-decomposition contradicting the assumption that $\sum_{i \in T'} |W'_i|$ is minimum. If not, there must be a

strongly connected set S such that vertices of S do not make a connected subtree in the new D-decomposition (obtained by dropping u_p from W_i'). This is possible only if $S \cap W_i' \cap W_k' = \{u_p\}$ and there are some vertices of S other than u_p in W_i'. But, then, the strongly connected set $(S \cup \{u_p\})\backslash\{u_q\}$ does not make a connected subtree in the original D-decomposition. As i is the furthest node from j with $0 < |U \cap W_i'| < m$, i is a leaf of the subtree that contains u_p (if a further node l contained u_p then $|W_l' \cap U| = m$ and thus $u_q \in W_l'$ so $T'|_{u_q}$ is not connected) and, hence, dropping u_p from W_i' does not violate conditions $(D1)$ and $(D2)$ for $T'|_{u_p}$. □

Now, given a D-decomposition with the above property we can replace every node that contains all vertices of U by u and obtain a D-decomposition of G with width $\frac{w'+1}{m} - 1$. □

A similar result holds for haven-width:

Lemma 16. *If G is a digraph, and K_m the complete graph on m vertices, then $hw(G \bullet K_m) = m \cdot hw(G)$.*

Proof. For this proof we define a function $f : \mathcal{P}(V(G \bullet K_m)) \to \mathcal{P}(V(G))$ by $f(X') = \{v | (v,k) \in X' \text{ for all } k \in V(K_m)\}$. First we show that if $G \bullet K_m$ has a haven, β, of order mk then G has a haven, β', of order k. We define β' as $\beta'(X) = f(\beta(X \times V(K_m)))$. Now as $\beta(X \times V(K_m)) \cap (X \times V(K_m)) = \emptyset$, every vertex $(x,y) \in \beta(X \times V(K_m))$ has $x \notin X$. But then, since $\beta(X \times V(K_m))$ is a strongly connected component (maximal strongly connected set), $\{x\} \times V(K_m) \subseteq \beta(X \times V(K_m))$ for each such x, so $\beta'(X)$ is non-empty and strongly connected. By observing that if $X \subseteq Y$ then $f(X) \subseteq f(Y)$, we see that $\beta'(X) \supseteq \beta'(Y)$ whenever $X \subseteq Y$.

For the converse, we show that if G has a haven, β, of order k then $G \bullet K_m$ has a haven, β', of order mk. For this we define $\beta'(X) = (\beta(f(X)) \times V(K_m)) \setminus X$. By the pigeon-hole principle, if $|X| < mk$ then $|f(X)| < k$, so β' is well-defined on sets X with $|X'| < mk$. Since β is a haven, $\beta(f(X))$ is non-empty and disjoint from $f(X)$. Thus $\beta(f(X)) \times V(K_m)$ has elements (x, y) such if $(x, y) \in X$, there exists $z \in V(K_m)$ such that $(x, z) \notin X$. Thus $\beta'(X)$ is non-empty and strongly connected. Again, the monotonicity of f implies $\beta'(X) \supseteq \beta'(Y)$ whenever $X \subseteq Y$. □

Unfortunately, directed tree-width is not obviously so well behaved. However, by replacing vertices in an arboreal decomposition by cliques, we obtain one direction of the analogous result.

Lemma 17. *If G is a digraph, and K_m the complete graph on m vertices, then* $1 + dtw(G \bullet K_m) \leq m \cdot (1 + dtw(G))$.

We observe that the graph in Figure 4.3 has directed tree-width 3, D-width 4, and haven-width 4, giving us an arbitrary gap between D-width, directed tree-width and haven-width.

Corollary 4. *For any $m \in \mathbb{N}$ there exists:*

1. *A graph with D-width $5m - 1$ and haven-width $4m$, and*
2. *A graph with D-width $5m - 1$ and directed tree-width $\leq 4m - 1$*

4.5 Upper Bounds for D-width

So far, we know some lower bounds for D-width, namely, directed tree-width, haven order, cop-monotonicity, and bramble number[3]. In this section we obtain a non-

[3]The bramble number result appears in [48].

trivial upper bound for D-width which is computable in polynomial time when D-width is constant. Unfortunately there doesn't exist any algorithm for computing optimal or nearly optimal D-decompositions. However, using the results of this section along with those of the previous section, we can compute non-trivial upper and lower bounds for D-width.

We prove that D-width is at most the number of cops required for a restricted cop-monotone winning strategy that we call *strongly cop-monotone.*

Definition 8. *A* strongly cop-monotone *strategy π is a cop-monotone strategy with the additional constraint that $\pi(X, R) \subseteq X \cup R$.*

Theorem 17. *Let G be a digraph. Then, if $k+1$ cops have a strongly cop-monotone winning strategy on G then the D-width of G is at most k.*

Proof. Let $\sigma = (Y_0, \pi)$ be a winning strongly cop-monotone strategy for $k+1$ cops. From Theorem 15, σ must also be robber-monotone. Let Π_σ be the strategy forest associated with σ. We define a D-decomposition (T, W) as follows:

1. $V(T) = V(\Pi_\sigma) \cup \{r\}$;
2. $E(T) = E(\Pi_\sigma) \cup \{(r, t) | t \text{ is a root of } \Pi_\sigma\}$;[4]
3. $W_r = Y_0$; and $W_t = \pi(X, R)$ for $t = (X, R) \in V(\Pi_\sigma)$.

It is clear that (T, W) has width at most $(k+1) - 1 = k$. Because σ is a winning strategy, every vertex must be occupied by a cop at some point, so for every strongly connected set S, $T|_S = \{t | W_t \cap S \neq \emptyset\} \neq \emptyset$. For condition (D2), let S be a strongly connected set. From the construction of Π_σ and the strong cop-monotonicity of σ, for any situation (Y_n, R_n) such that $S \cap \pi(Y_n, R_n) \neq \emptyset$, there is a unique path

[4]For the decomposition to be undirected we ignore the directions on the edges in Π_σ

$(Y_0, R_0), (Y_1, R_1), \ldots, (Y_n, R_n)$ such that $S \cap \pi(Y_i, R_i) \neq \emptyset$, for $0 \leq i \leq n$ and $S \subseteq R_0$. Moreover, (Y_0, R_0) is common in all such paths regardless of (Y_n, R_n). Hence, it suffices to show that S remains connected along paths of Π_σ. But this follows immediately from the cop-monotonicity of σ: if the cops occupy some of S, leave all vertices in S, and then occupy some of S, either they revisit a vertex (contradicting cop-monotonicity), or the robber can defeat σ on S (contradicting the fact that σ is winning).

□

4.5.1 Computing the strongly cop-monotonicity

Theorem 18. *Given a digraph G and an integer k, determining if k cops have a strongly cop-monotone winning strategy on G can be decided in time $O(n^{k+3})$, where $n = |V(G)|$. Furthermore, the algorithm will find such a strategy if one exists.*

Proof. The algorithm we present in Figure 4.4 recursively computes a k-cop strongly cop-monotone strategy π from position (X, R) by iterating through all possible values for X' which preserve monotonicity, and then checking that there is a winning strategy from (X', R') for all R' with a non-empty intersection with R. The correctness and running time of this algorithm follow in the next two lemmas. □

Lemma 18 (Correctness). *Given a digraph G and an integer k, $\pi =$* **strategy** *($\emptyset$, G, G, k) is a k-cop strongly cop-monotone winning strategy if, and only if, such a strategy exists.*

Proof. To show that the algorithm computes a strongly cop-monotone winning strategy, we first show that the computed strategy is strongly cop-monotone and then prove that it is a winning strategy. For every (X, R), $\pi(X, R) \subseteq X \cup R$, so π is

Algorithm strategy(X, R, G, k)
foreach $X' \subseteq X \cup R$ *with* $X' \neq X$, $|X'| = k$ **do**
  Let $R_1, R_2, \ldots, R_m$ be all strongly connected components of $G \backslash (X' \cap X)$ that have nonempty intersection with R
  $\forall i$, let $\sigma_i = \text{strategy}(X', R_i, G, k)$
  if $\sigma_i \neq \emptyset, \forall i$, **then** return $\sigma = (X, \pi)$ where $\pi(X, R) = X'$ and σ follows σ_i if the robber moves to R_i.
end
return $\emptyset$

Figure 4.4: Finding a strongly cop-monotone winning strategy

strongly cop-monotone. To show that the strategy is winning, we show that it is winning from each position (X, R). This is easily established by induction on the size of R, as $\pi(X, R)$ is defined as a set X' such that the strategy is winning from (X', R') for all reachable positions (X', R'). As we observed above, $R' \subseteq R$, and as $X \neq X' \subseteq X \cup R$, X' will include vertices from R, so R' will be strictly smaller than R.

For the converse, we need to show that if there is a k-cop strongly cop-monotone strategy $\sigma' = (Y_0, \pi')$ then $\pi'(X, R)$ is a possible return value for $\pi(X, R)$. Without loss of generality, we can assume $\pi'(X, R) \subseteq X \cup R$ and $|\pi'(X, R)| = k$ as we can always transform σ' into a strongly cop-monotone strategy which does satisfy these. From Theorem 15, σ' is also a robber-monotone strategy, so R is a strongly connected component of $G \setminus (X \cap \pi'(X, R))$. Finally, it is clear that if R' is a non-empty strongly connected component of $G \setminus \pi'(X, R)$, then $\pi'(\pi'(X, R), R') \neq \emptyset$. □

Lemma 19 (Running time). *Given a digraph G and an integer k,* **strategy**$(\emptyset, G, G, k)$ *runs in time $O(n^{k+2})$, where $n = |V(G)|$.*

Proof. We implement the algorithm using dynamic programming. There is an entry (X, R) in the dynamic programming table π for each subset of k vertices X and each

strongly connected component R of $G\backslash X$. The table is filled in increasing order of the size of R.

As there are at most n strongly connected components in $G\backslash X$, there exist at most $O(n^{k+1})$ possibility for any (X, R) pair. In computing $\pi(X, R)$ we try all possible $O(n^k)$ subsets X' of $X \cup R$ and, for each one, it takes $O(n^2)$ time to check if C is a strongly connected component of $G\backslash(X \cap X')$ (using depth-first search). Since each R_i is smaller than R, we can use dynamic programming (or memoization) so that the check of $\pi(X', R_i)$ takes constant time (after its initial calculation). In total, the running time of the algorithm is $O(n^{k+2})$.

□

4.6 Conclusion and Future Work

In this chapter we further study D-width and identify the class of digraphs with D-width one. We also obtain non-trivial upper bounds for D-width in terms of the number of cops that are required to capture the robber in (strongly) cop-monotone cops/robber games.

As D-width is an upper bound for directed tree-width, not only does D-width inherit all the algorithmic advantages of directed tree-width, such as an efficient algorithm for Hamiltonian cycle on bounded D-width graphs, but the simplicity of D-decompositions may also be used to establish other algorithmic results for digraphs with bounded D-width. Finding an algorithm for computing optimal or nearly optimal D-decompositions would have a direct effect on the efficiency of these solutions. Exploring the class of problems that are efficiently solvable on bounded D-width graphs is also a very interesting direction of future research.

Chapter 5

Hyper-D-width

5.1 Introduction

In this chapter, we introduce *hyper-D-width* and *hyper-T-width* as the first stable (see definition 9) measures of connectivity for hypergraphs. After studying some of their properties and, in particular, proposing an algorithm for computing nearly optimal hyper-T-decomposition when hyper-T-width is constant, we introduce some applications of hyper-D-width and hyper-T-width in solving hard problems such as the minimum vertex cover.

5.2 Background

In this section, we review the definitions that we use in this chapter. A hypergraph $H = (V, E)$ consists of a set of vertices V and a set of edges E where every edge $e \in E$ is a subset of V. A *path* in H is a sequence of vertices $u_1, u_2, \cdots, u_m$ such that u_i and u_{i+1} are both in some edge in H for $i = 1, 2, \cdots, m-1$. We say u is *connected* to v if there is some path from u to v. A set S of vertices is connected if

every two vertices in it are connected. A connected component of H is any maximal connected set of H. For a subset X of V, the hypergraph induced by X, denoted by $H[X]$, is (X, E'), where E' is the set of edges in E all of whose vertices are in X. Finally, the hypergraph $H \backslash X$ is $H[V \backslash X]$. Notice our different interpretation of connectivity in this Chapter. Given an edge $e = \{v_1, v_2, \cdots, v_k\}$, we consider it as a single connected unit meaning that e collapses by removing any of v_i's. This is in contrast with those definitions that define connectivity based on the primal graph.

Three graphs are often associated with any hypergraph H: The *primal graph*, the *dual graph*, and the *incidence graph*. The primal graph is obtained by making a clique out of the vertices in every edge in H. The dual graph is obtained by representing every edge by a vertex and connecting two vertices if their corresponding edges intersect. The incidence graph is a bipartite graph whose first part corresponds to vertices in H and whose second part corresponds to edges in H. A vertex u in the first part is connected to a vertex e in the second part if $u \in e$ in H. The tree-widths of the primal, dual, and incidence graphs are often referred to as *primal*, *dual*, and the *incidence* tree-width, respectively.

A famous example of using hypergraphs is using them to model inputs to the SAT problem. A boolean formula in conjunctive normal form with clause sets $C_1, C_2, \cdots, C_m$ of variables $X = \{x_1, x_2, \cdots, x_n\}$ and their negations $\overline{x_1}, \overline{x_2}, \cdots, \overline{x_n}$ is modeled by a hypergraph $H = (X, E)$ where $E = \{e_1, e_2, \cdots, e_m\}$ and, for all k, $x_k \in e_j$ iff either $x_k \in C_j$ or $\overline{x_k} \in C_j$.

For example, for $\varphi = (a \vee b \vee c) \wedge (a \vee \bar{c}) \wedge (\bar{b} \vee c) \wedge b$, the corresponding hypergraph is $H = (\{a, b, c\}, \{\{a, b, c\}, \{a, c\}, \{b, c\}, \{b\}\})$.

For these formulas, the tree-width of the incidence graph seems to be the most general parameter for which the SAT problem is fixed parameter tractable.

Theorem 19. *[51] Satisfiability of clause-sets with bounded incidence tree-width is fixed-parameter tractable.*

A problem is *fixed parameter tractable*, if it admits an algorithm with running time $O(f(k)n^{\alpha})$ where k is some parameter independent of n, f is any function of k, and α is a constant independent of k and n. As k doesn't appear in the exponent of n, instances of large size n can be solved efficiently.

Such a fact was already known for primal tree-width [27], but the above is stronger as the incidence tree-width is always smaller than the primal tree-width plus one [51].

One parameter that is used many times in the literature is *hypertree-width* and similar variants which was first introduced by Gottlob et al. [26]. One parameter that is used many times in the literature is *hypertree-width* and similar variants which was first introduced by Gottlob et al. [26]. A *generalized hypertree decomposition* of a hypergraph $H = (V, E)$ is a triple (T, W, λ), where (T, W) is a tree-decomposition of the primal graph of H and λ is a function that assigns to every vertex t of T a set of edges in E such that $W_t \subseteq \bigcup \lambda(t)$. The width of (T, W, λ) is the maximum of $|\lambda(t)|$ over all nodes t of T. (Generalized) Hypertree-width is different from tree-width of the primal graph only in the way we measure the width of a tree-decomposition: Instead of counting the number of vertices in a node, we count the number of edges that cover these nodes. The *generalized hypertree-width* of H is the minimum width over all its generalized hypertree-decompositions. A *hypertree-decomposition* (T, W, λ) is a generalized hypertree decomposition that satisfies one special condition: $(\bigcup \lambda(t)) \cap X(T_t) \subseteq W_t$, where T_t is the subtree rooted at t and $X(T_t)$ is $\bigcup_{s \in T_t} W_s$. This condition is added for technical reasons to make the hypertree decomposition computable when it is constant. It is not known whether

generalized hypertree-width is computable in polynomial time when it is constant. The *hypertree-width* of H is defined accordingly.

Gottlob et al. (See [26] or [25] for a survey) show how an optimal hypertree-decomposition can be computed for a hypergraph with bounded hypertree-width by associating it with certain cops and robber games. They also show that the constraint satisfaction problem is solvable in polynomial time for constant hypertree-width hypergraphs. A constraint satisfaction problem is a set of constraints (S_i, R_i) where S_i is a tuple of variables from a set of variables X and R_i is a list of tuples of values from some domain D. A solution to CSP is a valuation such that all constraints are satisfied. A valuation $V : X \rightarrow D$ satisfies constraint $((x_1, x_2, \cdots, x_k), R)$ if $(v(x_1), v(x_2), \cdots, v(x_k)) \in R$. In this specific model, all possible valuations of the tuple S_i are explicitly given, i.e. are part of the input, and, hence, the number of possible valuations is upper bounded by the input size. For example, in the SAT input $\varphi = (a \vee b \vee c) \wedge (a \vee \overline{c}) \wedge (\overline{b} \vee c) \wedge b$, the second clause is represented as $((a, c), \{(T, F), (T, T), (F, F)\})$ in this model. This is in contrast to a typical input to SAT (and other problems), which represents the set of values that satisfy a constraint via a formula (e.g. $(a \vee \overline{c})$). In this model, if we add a big constraint (i.e. S_i is big) we can make the problem easier to solve. For example, if a constraint contains all the variables, then we can simply try all valuations given for that constraint and check if one works for the other constraints as well. That's basically why hypertree-width is related to the time required to solve CSP.

Adler et al. [2] prove that hypertree-width is within a factor $3 + \epsilon$ of several other hypergraph measures: generalized hypertree-width, (monotone) marshal-width , hyperbramble number, hypertangle number, hyperbranch-width, and hyper-linkedness.

5.3 Hyper-D-width

5.3.1 Definition

Let $H = (V, E)$ be a hypergraph. A *hyper-D-decomposition* of H is a pair (T, W) where T is a tree and $W = \{W_t | t \in V(T)\}$ is a family of subsets of $V(H)$ such that for every connected set S:

(H1) $T|_S := \{t | W_t \cap S \neq \emptyset\} \neq \emptyset$, and

(H2) The subgraph of T with vertex set $T|_S$ and edges $\{(s,t) | W_s \cap W_t \cap S \neq \emptyset\}$ forms a connected subtree of T.

The width of a hyper-D-decomposition (T, W) is the maximum of $|W_t| - 1$ over all nodes $t \in V(T)$. The *hyper-D-width* of a hypergraph is the minimum width over all its hyper-D-decompositions. For example, a hyper-D-decomposition with width two for the hypergraph $H = (\{1,2,3,4\}, \{\{1,2,3\}, \{1,4\}, \{2,4\}, \{3,4\}\})$ is depicted in Fig. 5.1. It's not hard to prove that it's, in fact, a minimum width hyper-D-decomposition.

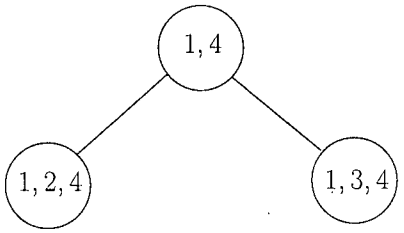

Figure 5.1: A hyper-D-decomposition with width two.

5.3.2 Basic Properties

Hyper-D-width is a generalization of tree-width. On regular graphs, where every edge has two vertices, hyper-D-decomposition wants the two vertices in every edge to share a node. This is exactly what any tree-decomposition wants.

Theorem 20. *For every undirected graph G, tree-width(G) = hyper-D-width(G).*

Let (T, W) be a hyper-D-decomposition for a hypergraph H. If we make a regular graph on the vertices of H by connecting two vertices iff they share a node in T, then the result would be a chordal graph with the same tree-width as the width of T.

Theorem 21. *For every hypergraph H with hyper-D-width w, there exists an undirected chordal graph G with tree-width w such that for any edge e of H with vertex set S, $G[S]$ is connected.*

Hyper-D-width is inspired by D-width on directed graphs. In fact, every minimal strongly connected set in digraphs is treated as an hyperedge in the definition of D-width meaning that hyper-D-width is, in some sense, a generalization of D-width.

5.3.3 Stability

Almost all existing connectivity measures for hypergraphs are very sensitive to big edges, i.e. edges that contains many vertices. (Generalized) Hypertree-width, hyperlinkedness, hyperbramble number, (monotone) marshal-width are all constant when we have an edge that contains all the vertices no matter what the rest of the hypergraph structure is. On the other hand, the tree-width of the primal graph is always $n - 1$ for the above example, where n is the number of vertices.

We would like a connectivity (cyclicity) measure on hypergraphs to behave in a stable way (as tree-width does for regular graphs): adding a constant number of vertices or edges shouldn't substantially change the connectivity. All the aforementioned measures violate this *stability condition*; so do the tree-width of the dual graph and the tree-width of the incidence graph.

Unlike all the above-mentioned connectivity measures for hypergraphs, hyper-D-width is stable. Let's formally define stability first and then prove the stability of hyper-D-width.

Definition 9. *A measure defined on hypergraphs is* stable *if after removing a constant number of vertices (with all edges containing those vertices) or a constant number of edges (defined on existing vertices) the measure decreases by only a constant.*

Theorem 22. *Hyper-D-width is stable.*

Proof. It's sufficient to show that hyper-D-width changes by a constant when adding one new vertex or one new edge. Assume we add a new vertex u and an arbitrary number of edges containing u to a hypergraph H. Let (T, W) be an optimal hyper-D-decomposition of H with width w. Obviously, (T, W'), where $W'_t = W_t \cup \{u\}$ for every $t \in V(T)$, is a proper hyper-D-decomposition of the new hypergraph with width $w + 1$. In case we add a new edge $e = \{v_1, v_2, \cdots, v_k\}$ to the hypergraph H, (T, W'), where $W'_t = W_t \cup \{v_1\}$ for every $t \in V(T)$, is a proper hyper-D-decomposition of the new hypergraph with width at most $w + 1$. □

In contrast, all the existing alternative measures are unstable.

Theorem 23. *(Generalized) Hypertree-width, hyperlinkedness, hyperbramble number, (monotone) marshal-width, (dual, incidence, or primal) tree width are all un-*

stable.

Proof. It's proven in [2] that the first four parameters are within a constant factor of each other. Hence it suffices to prove hypertree-width and (dual, incidence, primal) tree width are unstable. Let H be a hypergraph with big hypertree-width. Adding one edge that contains all the vertices makes the hypertree-width equal to one. Thus, hypertree-width is unstable. Similarly, the primal tree width is unstable. Now, let H be a hypergraph with small dual tree-width. Let H' be obtained from H by adding a new vertex u and all possible edges that contain u (i.e. 2^n edges where n is the number of vertices). The dual graph of H' has a clique of size 2^n (all edges that contain u) which means it has dual tree-width at least $2^n - 1$, which shows dual tree-width is unstable (under removal of vertex u). As for the incidence graph suppose H is a hypergraph with small incidence tree-width. We show that H', as constructed above, has large incidence tree-width. Let I be the first $\frac{n}{2}$ vertices of H. The number of edges in H' that contain all vertices in I is at least $2^{\frac{n}{2}}$ which means the incidence graph of H' contains a $K_{\frac{n}{2},\frac{n}{2}}$ (actually $K_{\frac{n}{2},2^{\frac{n}{2}}}$ as well) subgraph. Hence, its tree-width is at least $\frac{n}{2}$. □

5.3.4 Comparison

In this section we compare hyper-D-width with other existing parameters defined on hypergraphs, namely, hypertree-width and the tree-width of the primal, dual, and incidence graph.

Theorem 24. *For any hypergraph H,*

- *hyper-D-width*$(H) \leq$ *primal tree-width*(H).
- *hyper-D-width*$(H) \leq$ *incidence tree-width*(H).

- *hyper-D-width*$(H) \leq$ *dual tree-width*$(H) + 1$.

Proof. The first inequality follows from the fact that every tree-decomposition of the primal graph is a hyper-D-decomposition. On the other hand, there exist hypergraphs with primal tree-width $n-1$ and hyper-D-width one (a hypergraph with one edge containing all the vertices).

As for the incidence tree width, let (T, W) be a tree-decomposition of the incidence graph. For each edge $e = \{v_1, v_2, \cdots, v_k\}$, replace every occurance of e in W_t for $t \in V(T)$ with v_1. Since e and v_1 share some node of T, i.e. $\{e, v_1\} \subseteq W_t$ for some $t \in V(T)$, the nodes that contain v_1 still make a connected subtree. Moreover, since e shares some node with every v_i, $1 \leq i \leq k$, the vertices $v_1, v_2, \cdots, v_k$ make a connected subtree in the resulting tree-decomposition. Again, there are hypergraphs with small hyper-D-width, but large incidence tree-width. Assume H has $2n$ vertices $\{1, 2, \cdots, 2n\}$ and n edges of the form $e_i = \{1, 2, \cdots, n, n+i\}$ for $1 \leq i \leq n$. The incidence graph has a $K_{n,n}$ subgraph (every i is connected to e_j for $1 \leq i, j \leq n$) and, hence, has tree-width at least n. However, its hyper-D-width is one. Its minimum width hyper-D-decomposition is a star with root r containing $W_r = \{1\}$ and i^{th} leaf containing $W_i = \{1, i\}$ for $2 \leq i \leq 2n$.

Almost the same statement holds when comparing the dual tree width and hyper-D-width. Given a tree-decomposition (T, W) with width w of the dual graph of hypergraph H, we show how a hyper-D-decomposition of H with width at most $w+1$ can be constructed [1]. For any vertex $u \in V(H)$, all edges that contain u make a clique in the dual graph. Hence, there is some node in T, say $\lambda(u)$, that contains all such edges. In the first step, for all $u \in V(H)$ we add a leaf $l(u)$ with $W_{l(u)} = W_{\lambda(u)} \cup \{u\}$ and attach it to the node $\lambda(u)$ in T . Next, for every edge

[1] On regular graphs we can remove the additive constant one in the inequality.

$e \in E(H)$, we pick an arbitrary vertex $v \in e$ and replace e with v in every W_t for $t \in V(T)$. Call the resulting decomposition (T', W'). Now, (T', W') contains only vertices of H. We claim that (T', W') is a hyper-D-decomposition of H. First, the new leaves insure that every vertex $u \in H$ is contained in some W_t (in particular, $W_{l(u)}$) in T'. Second, for any vertex $u \in H$, all nodes $t \in T'$ such that $u \in W'_t$ make a connected subtree. Every edge e replaced by u in creating T', forms a connected subtree in T that contains the node $\lambda(u)$. Hence, these subtrees are connected in T' and they connect to the new leaf node $l(u)$. Third, suppose $e = \{x_1, x_2, \cdots, x_k\}$ is replaced by x_1, then $T'|_{x_1} \cap T'|_{x_j} \neq \emptyset$. In fact, $\lambda(x_j) \in T'|_{x_1} \cap T'|_{x_j}$. So the subtrees corresponding to vertices of e form a connected subtree. A star graph of size n has tree-width one and (by Theorem 20) hyper-D-width one whereas its dual graph is a clique and, hence, has tree-width $n - 1$. □

Corollary 5. *The class of bounded hyper-D-width hypergraphs contains the class of bounded (primal, dual, or incidence) hypergraphs.*

5.3.5 cops and robber game

In chapter 4 we obtained lower and upper bounds for D-width in terms of the number of cops that are required to win certain cops/robber game. An identical definition of cops and robber games under the new definition of connected components enable us to obtain similar results on hypergraphs.

Theorem 25. *let h be a hypergraph.*

1. *If $k+1$ cops have a strongly cop-monotone winning strategy on H then H has hyper-D-width at most k.*

2. *If H has hyper-D-width at most k then $k+1$ cops have a cop-monotone winning strategy on H.*

In addition, there exist an algorithm for computing the strongly cop-monotonicity of H in time $O(n^{k+2})$.

5.3.6 Applications

In this section we show that there exist polynomial-time approximation schemes (PTAS) for many problems including vertex cover, dominating set, and multicut problems on hypergraphs when the hyper-D-width of the input hypergraph is constant.

Let a generic problem P be as follows: Find a minimum number of vertices in a hypergraph that satisfy some constraint C.

We are especially interested in problems P with the following property.

(**Decomposable Property**)Problem P is decomposable if it satisfies the following condition. Let H be a hypergraph. For any subset X of vertices of H, let $C_1, C_2, \cdots, C_m$ be the connected components of $H \backslash X$. Let D_i be a solution for P on $H[C_i]$. Then, $X \cup (D_1 \cup D_2 \cup \cdots \cup D_m)$ is a solution of P on H.

The decomposable property lets us choose any suitable X, put it in the solution, break the input hypergraph into smaller parts, and solve the problem on each part independently. It's easy to verify that multi-cut, dominating set, and vertex cover are examples of such problems. For example, for the vertex cover problem, if we choose all vertices in X then any edge that intersects both C_i and C_j $(i \neq j)$ is covered. The rest is, then, solving the vertex cover problem on each C_i separately.

Now, we show how such problems have a PTAS on bounded hyper-T-width

hypergraphs. The idea is similar to the technique that Calinescu et al. [15] use to find a PTAS for multicut on bounded tree-width graphs and digraphs. Let (T, B, W) be a hyper-T-decomposition with width w for the input hypergraph. Let $t \in V(T)$ be the bottom-most node such that there exists an optimal global solution that contains at least w/ϵ vertices in the subtree T_t rooted at t. If there is no such t then the optimal global solution has fewer than w/ϵ vertices. Such a solution can be computed in time $O(n^{w/\epsilon})$. Otherwise, let opt_t be the number of vertices of an optimal solution in T_t. Choosing and removing all vertices in $\lambda_t = B_t \cup \bigcup_{e \sim t} W_e$ breaks T_t into several connected components each having less than w/ϵ vertices of the optimal solution. Hence, the problem can be solved by brute force on all these components in time $O(n^{w/\epsilon})$. Hence, the number of vertices that we pick is at most $opt_t + w \leq opt_t(1+\epsilon)$. Assume that the rest of the hypergraph has o vertices in the optimal solution. According to the induction hypothesis, we can find a solution with at most $o(1+\epsilon)$ vertices. Hence, we can solve the problem on H by choosing at most $(1+\epsilon)(opt_t + o)$ vertices, yielding a $(1+\epsilon)$-factor approximation for P. The details of the algorithm are shown in Fig. 5.2.

To complete this section we prove that all aforementioned problems are hard even on bounded hyper-D-width hypergraphs.

Theorem 26. *The vertex cover problem, the dominating set problem, and the multicut problem are NP-Complete on bounded hyper-D-width hypergraphs.*

Proof. Let $C_1, C_2, \cdots, C_m$ be a SAT problem instance over variables $x_1, x_2, \cdots, x_n$. We make a hypergraph $H = (V, E)$ with vertex set $V = \cup_{1 \leq i \leq n} \{x_i, \overline{x_i}, z_i\} \bigcup \{u\}$ and edge set $e_i = C_i \cup \{u\}$, for $1 \leq i \leq m$ and $e_{m+i} = \{x_i, \overline{x_i}, z_i\}$, for $1 \leq i \leq n$. We claim that the SAT problem instance is satisfiable iff the hypergraph H has a vertex cover of size exactly n. Assume the SAT instance is satisfiable with setting all x_i's

Input: A hypergraph H together with an optimal hyper-D-decomposition (T, W)
Output: Minimum number of vertices that satisfy P
/* Let Let $X_t = \cup_{t' \in T_t} W_{t'}$. */
foreach $t \in T$ **do**
  Let t_i, $i = 1, 2, \cdots, m$, be children of t.
  foreach i **do**
    find an optimal solution o_i to P of size at most w/ϵ for $H[X_{t_i} - W_t]$.
    If no such solution exist then try the next t.
  end
  Let $o = \cup_i o_i$.
  Recursively find a $(1+\epsilon)$-factor approximation solution o' for $H \backslash X_t$.
  Let $opt_t = o \cup o' \cup W_t$.
end
return opt_t with minimum size.

Figure 5.2: Solving decomposable problem P on a bounded hyper-D-width hypergraph

in a set X to be true and the rest to be false. In the hypergraph let the vertex cover be $X \cup \{\overline{x_i} | x_i \notin X\}$. Obviously, every edge of type $C_i \cup \{u\}$ and every edge of type $\{x_i, \overline{x_i}, z_i\}$ is covered and the size of the vertex cover is n. On the other hand, let X be a vertex cover of size at most n. Obviously, it must have picked exactly one vertex from every triple $\{x_i, \overline{x_i}, z_i\}$, $1 \leq i \leq n$ which is at least n vertices. Moreover, every edge of type $C_i \cup \{u\}$ must be covered by some x_i or $\overline{x_i}$ in C_i, which makes a satisfiable solution. Finally, we mention that the hypergraph constructed above has hyper-D-width at most three. Make a star with root u and every leaf containing $\{u, x_i, \overline{x_i}, z_i\}$, for $1 \leq i \leq n$.

It's easy to prove that the above reduction works for the dominating set problem as well. The NP-Completeness of the multicut problem follows from its NP-Completeness on bounded tree-width graphs proven by Calinescu et al. [15]. □

5.4 Introducing hyper-T-width

Although hyper-D-width has many nice properties and resembles undirected tree-width in a natural way, it has one big disadvantage that we haven't resolved yet: We don't know a polynomial time algorithm for computing optimal or even approximately (within a constant factor) optimal hyper-D-decompositions for bounded hyper-D-width hypergraphs.

One option is to consider a generalization of directed tree-width [34, 35] instead. Recall the following definition from [35].

Definition 10. *Let T be a directed tree. For a vertex t and edge e we say $e \sim t$ if t is one of the end points of e. We also say $t > e$ if either t is the head of e or there is a directed path from the head of e to t in T.*

Definition 11 (Arboreal (pre-)decompositions and directed tree-width). *An* arboreal pre decomposition *of a digraph G is a tuple (T, B, W) where T is a directed tree with a unique root, and $B = \{B_t | t \in V(T)\}$ and $W = \{W_e | e \in E(T)\}$ are sets of subsets of $V(G)$ that satisfy:*

(R1) B is a partition of $V(G)$ into (possibly empty) sets such that $B_r \neq \emptyset$ for the root r of T, and

(R2) If $e \in E(T)$, then $B_e^{\downarrow} := \bigcup\{B_t | t > e\}$ is W_e-normal or empty.

The width *of an arboreal pre-decomposition (T, B, W) is the minimum k such that for all $t \in V(T)$, $|B_t \cup \bigcup_{e \sim t} W_e| \leq k + 1$. An* arboreal decomposition *is a pre-decomposition in which all B_t are non-empty, and the* directed tree-width *of a digraph G, $dtw(G)$, is the minimal width of all its arboreal decompositions.*

D-decomposition is, in fact, a restricted variant of arboreal pre- decomposition. Given a D-decomposition (T, W) of width w of a digraph G, the following is an arboreal pre-decomposition of width w for G. (T', B', W'), where T' is obtained from T by choosing a random root and directing all edges away from the root. For any edge $e = (u, v)$ in T', $B'_v = W_v - W_u$ and $X_e = W_v \cap W_u$.

According to Johnson et al. [34] a set S is Z-normal if every path from a vertex in S to another vertex in S that contains a vertex in $V(G) - S$ has a vertex in Z as well. On the other hand, there is no strongly connected component of $G \backslash Z$ that contains vertices from both S and $V(D) - S$.

Inspired by the above definition, we define hyper-T-width as follows.

Definition 12 (Hyper T-decomposition and hyper T-width). *A* hyper T-decomposition *of a hypergraph H is a tuple (T, B, W) where T is a directed tree with a unique root, and $B = \{B_t | t \in V(T)\}$ and $W = \{W_e | e \in E(T)\}$ are sets of subsets of $V(H)$ that satisfy:*

(R1) B is a partition of $V(H)$ into possibly empty sets such that $B_r \neq \emptyset$ for the root r of T, and

(R2) If $e \in E(T)$, then $B_e^{\downarrow} := \bigcup\{B_t | t > e\}$ is W_e-normal or empty. A set S is Z-normal if there is no connected component of $H \backslash Z$ that contains vertices from both S and $V(H) - S$.

The width *of a hyper T-decomposition (T, B, W) is the minimum k such that for all $t \in V(T)$, $|B_t \cup \bigcup_{e \sim t} W_e| \leq k + 1$. The* hyper T-width *of a hypergraph H is the minimal width of all its hyper T-decompositions.*

In the following section we show that hyper-T-width has all the nice properties of hyper-D-width. It's stable, has the balanced-separator property, and has

all currently known algorithmic advantages of hyper-D-width. In addition, we can compute an approximate hyper-T-decomposition when hyper-T-width is constant.

5.4.1 Properties

Theorem 27. *Hyper-T-width is stable.*

Proof. It's sufficient to show that hyper-T-width changes by a constant when adding one new vertex or one new edge. Assume we add a new vertex u and an arbitrary number of edges containing u to a hypergraph H. Let (T, B, W) be an optimal hyper-T-decomposition of H of width w. Obviously, (T, B', W'), where $W'_t = W_t \cup \{u\}$ for every $t \in V(T)$ and $B'_r = B_r$, when r is not the root and $B'_r = B_r \cup \{u\}$ for the root r, is a proper hyper-T-decomposition of the new hypergraph. Moreover, (T, B', W') had width $w + 1$. In case we add a new edge $e = \{v_1, v_2, \cdots, v_k\}$ to the hypergraph H, (T, W', B'), where $W'_t = W_t \cup \{v_1\}$ for every $t \in V(T)$, is a proper hyper-T-decomposition of the new hypergraph. □

As mentioned in ther earlier section, a hyper-D-decomposition is a restricted version of a hyper-T-decomposition. Hence,

Theorem 28. *For any hypergraph H, the hyper-T-width of H is less than or equal to its hyper-D-width.*

Therefore,

Theorem 29. *The class of bounded hyper-T-width hypergraphs contains the class of bounded (primal, dual, and incidence) hypergraphs.*

Proof. This is a direct consequence of Theorems 24 and 28. □

As for algorithmic applications of hyper-T-width we show that a problem P that satisfies the decomposable property admits a PTAS on bounded hyper-T-width hypergraphs. The idea is quite similar to hyper-D-width. We consider the deepest edge $e = (r, r')$ (i.e. r has maximum depth) such that the subgraph $B_e^{\downarrow}$ has more than $\frac{w}{\epsilon}$ vertices of some optimal solution, where w is the hyper-T-width of the input hypergraph and ϵ is any constant. Now adding all vertices X_e in the solution does not change the number of vertices in the solution by more than a multiplicative factor $1 + \epsilon$.

5.4.2 Computation

The big advantage of hyper-T-width over hyper-D-width is the fact that we can approximately compute it when it is constant. Johnson et al. [34] prove this for directed tree-width and their proof generalizes immediately to hypergraphs. In particular, they introduce the notion of haven order and prove that directed tree-width and haven order are within a constant factor of each other.

Theorem 30 (Johnson et al. [34]). *$H(G) - 1 \leq$ tree-width$(G) \leq 3H(G) + 1$ for digraphs G, where $H(G)$ is the haven order of G.*

Their proof for tree-width$(G) \leq 3H(G)+1$ is constructive. If the haven order of G is at most w then it builds an arboreal decomposition of width at most $3w+1$. In this section we show that the construction quickly transfers to hypergraphs without any major change.

We basically mimic the proof of Johnson et al. [34] here. As our definitions of connectivity and havens are different from theirs, the proof is completely illustrated below.

Let's start with defining haven order on hypergraphs.

Definition 13 (haven and haven-order). *Let H be a hypergraph. A* haven *of order k is a function β assigning to every set $Z \subseteq V(H)$ with $|Z| < k$, a connected component of $H \setminus Z$ such that if $X \subseteq Y \subseteq V(H)$ and $|Y| < k$ then $\beta(Y) \subseteq \beta(X)$. The* haven-order *of H is the largest k such that H has a haven of order k*

Theorem 31. *$H(H) - 1 \leq$ hyper-T-width$(H) \leq 3H(H) + 1$ for hypergraphs H, where $H(H)$ is the haven order of H.*

Proof. **(Left inequality)**

Let (T, W, B) be a hyper-T-decomposition of width w of H. For any node t let $\lambda_t = B_t \cup \bigcup_{e \sim t} W_e$. First observe that $w + 1$ cops can catch a robber in the cops/robber game. Assume a hyper-T-decomposition (T, W, B) of G of width w is given. The cops can start at $X_0 = \lambda_r$, where r is the root. Let $T_1, T_2, \cdots, T_m$ be subtrees of T with roots $r_1, r_2, \cdots, r_m$, children of r. Let $e_i = (r, r_i)$. According to conditions of hyper-T-decompositions, the robber can only be at vertices in one of the sets $B^{\downarrow}_{e_i}$. The cops can then move to λ_{r_i} and continue the strategy until they trap the robber in one of the leaves.

On the other hand if H has a haven of order h then the robber has a winning strategy against $h-1$ cops by staying at $\beta(Z)$, where Z is the set of vertices occupied by the cops. Consequently, $H(H) - 1 \leq$ hyper-T-width(H).

(Right inequality) Assume H has no haven of order w. First, let's prove the following crucial lemma.

Lemma 20. *If H has no haven of order w then for every set Y of vertices of H with $|Y| \leq 2w - 1$ there is a subset Z of vertices of H such that $|Z| < w$ and every connected component of $H \setminus Z$ has at most $w - 1$ vertices of Y.*

Proof. Assume not. Then for every set Z with $|Z| < w$ there is one connected

component of Z, say $\beta(Z)$, such that $|\beta(Z) \cap Y| \geq w$. But, this is a contradiction as β is a haven of order w. For any Z and Z' with $Z \subset Z'$ and $|Z'| < w$, both $\beta(Z)$ and $\beta(Z')$ contain at least w vertices from Y. As $|Y| \leq 2w - 1$, we conclude that $\beta(Z') \cap \beta(Z) \neq \emptyset$ which means $\beta(Z') \subseteq \beta(Z)$. □

Consider a hyper-T-decomposition (T, W, B) of H with the following restrictions.

1. For any node r, if r is not a leaf then $|B_r| \leq w$ and $|\lambda_r| \leq 3w - 1$.

2. For any edge e, $|W_e| \leq 2w - 1$.

3. Subject to the above conditions we take the hyper-T-decomposition that minimizes the maximum of $|B_r|$, for all r's.

As the obvious hyper-T-decomposition with one vertex r and $B_r = V(H)$ satisfies the first two conditions, we conclude that there exist a hyper-T-decomposition (T, W, B) satisfying the above three conditions. If there is no leaf with $|B_r| > r$ then we are done and (T, W, B) would have width at most $3w - 2$. Otherwise, take the leaf r with maximum $|B_r|$. Assume r' is the unique root of r, $e = (r', r)$ and $Y = W_e$. According to lemma 20 there exist a set Z_0 of at most $w - 1$ vertices of H such that every connected component of $H \backslash Z_0$ contains at most $w - 1$ vertices from Y. Let $Z = Z \cup \{u\}$ for some arbitrary vertex u in B_r. We build a new hyper-T-decomposition satisfying the first two conditions, but a smaller $|B_r|$ which is a contradiction.

Let $X_1, X_2, \cdots, X_m$ be the connected components of $H[B_r] \backslash Z$. We create m new leaves $r_1, r_2, \cdots, r_m$ and connect them to r (i.e. create edges from r to each r_i). Set $B_{r_i} = X_i$ and change B_r to $Z \cap B_r$. For the edge $e_i = (r, r_i)$ set

$W_{e_i} = Z \cup (Y \cap X_i)$. As $|Y \cap X_i| \leq 2w - 1$ this insures that $|W_{e_i}|$ satisfies condition 2. By the way, $\lambda_r = Z \cup Y$; thus, $|\lambda_r| \leq 3w - 1$. □

The proof that hyper-T-width$(H) \leq 3H(H) + 1$ is constructive and can be used to obtain an approximate hyper-T-decomposition. There is at most n optimization steps during which we find a set Z and add now leaves to the existing hyper-T-decomposition. Finding a set Z of size less than w and verifying that all connected components $H \backslash Z$ contain less than w vertices of Y can be done in time $O(n^{w+2})$. Hence the total running time for finding an approximate hyper-T-decomposition would be $O(n^{w+3})$.

Chapter 6

Conclusions and Research Directions

In this thesis we covered problems related to metric embedding and tree-width. We obtained a low distortion embedding of series-parallel graphs into ℓ_1, computed optimal embeddings between line metrics when the distortion was small enough, and proposed tree-width-like connectivity measures, D-width, hyper-D-width, and hyper-T-width , for digraphs and hypergraphs.

As series parallel graphs and k-outerplanar graphs have bounded tree-width and both are known to have constant distortion (for constant k) embedding into ℓ_1, bounded tree-width graphs are conjectured to have bounded distortion embedding into ℓ_1. Since a good distortion embedding into ℓ_1 implies good approximation for several fundamental problems, such as the sparsest cut and multicut problems, the study bounded tree-width graphs and their connection to ℓ_1 metrics becomes important.

Such a connection between tree-width and metric embedding and, also, the

fact that the study on tree-width has found numerous applications in practice, inspires people to extend it to similar class of objects: digraphs and hypergraphs.

Among many proposed measures for directed graphs, D-width seems to be the simplest. As for hypergraphs, hyper-D-width and hyper-T-width are the first stable connectivity measures for hypergraphs and are more general than primal, dual, and incidence tree-width. They also have application in minimum vertex cover, minimum dominating set, and multicut problems.

6.1 ℓ_1 metrics

We've proven that the algorithm given by Gupta et al. gives an embedding with distortion at most 6.0 for every series parallel graph, but gives distortion at least 3.0 even for some outerplanar series parallel graphs.

An interesting open problem is to close this gap. Some other relevant open problems are minimizing the number of used dimensions (which is exponential with Gupta et al.'s approach) or embedding higher tree-width graphs (tree-width 3 as the first step) into ℓ_1 with bounded distortion.

6.2 Line metrics

We currently know how to compute an optimal embedding between two line metrics when the optimal distortion is small (less than 13.602) and know it is hard to do so when the distortion is at least n^ϵ for some constant ϵ [29]. An open problem is to close this gap and, for example, study its hardness when the distortion is $O(\log^\epsilon n)$.

Another very interesting problem is to look for embeddings that have close to the optimal distortion. Although finding a constant-factor approximation when the

optimal distortion is at least n^{ϵ} is hard [29], finding such an approximate distortion seems to be a lot easier for smaller distortions.

6.3 D-width

A very challenging open topic is to study the connection between D-width and directed metrics or directed cut problems. Bounded D-width digraphs admit PTAS for multi-cut problems (when two vertices are considered separated if they belong to two different strongly connected components). Chuzhoy and Khanna [20] recently proved than the sparsest cut problem and the multicut problem are hard to approximate within a factor $2^{\Omega(\log^{1-\epsilon} n)}$ for any constant ϵ even on directed acyclic graphs. This is a big negative result that basically says that D-width and directed tree-width are irrelevant to cut problems and directed metrics, but still leaves the open problem of studying digraph classes that admit constant-factor approximate solutions for cut problems.

One other research direction is to explore other algorithmic aspects of D-width such as its computation with D-width is constant.

6.4 Hyper-D-width

Hyper-D-width and hyper-T-width are very new and there exist several open problems related to them. An efficient algorithm (showing it is fixed parameter tractable in particular) for computing optimal (or approximate) hyper-D-decompositions for small hyper-D-width would be very useful problem. Exploring algorithmic aspects of hyper-D-width and hyper-T-width is also a very nice research direction. There are some fundamental problems (such as solving CSP, SAT, and finding Nash Equilibria

of certain games) that seem to be relevant to hyper-D-width and hyper-T-width and finding those connections is an interesting problem.

Bibliography

[1] Isolde Adler. Directed tree-width examples. *Journal of Combinatorial Theory(Series B)*, 2007. To appear.

[2] Isolde Adler, Georg Gottlob, and Martin Grohe. Hypertree-width and related hypergraph invariants. In Stefan Felsner, editor, *2005 European Conference on Combinatorics, Graph Theory and Applications (EuroComb '05)*, volume AE of *DMTCS Proceedings*, pages 5–10. Discrete Mathematics and Theoretical Computer Science, 2005.

[3] Michael H. Albert and Mike D. Atkinson. Simple permutations and pattern restricted permutations. *Discrete Mathematics*, 300(1-3):1–15, 2005.

[4] A. Andoni, M. Deza, A. Gupta, P. Indyk, and S. Raskhodnikova. Lower bounds for embedding edit distance into normed spaces. In *Proceedings of Annual ACM-SIAM Symposium on Discrete Algorithm*, pages 523–526, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[5] Mike Atkinson and Derek Holton, editors. *Permutation patterns.* Electronic Journal of Combinatorics, Clemson, SC, 2003. Including selected papers from the conference held in Otago, February 10–14, 2003, Electron. J. Combin. **9** (2002/03), no. 2.

[6] Y. Aumann and Y. Rabani. An o(log k) approximate min-cut max-flow theorem and approximation algorithm. *SIAM Journal on Computing*, 27, 1998.

[7] D. Avis and M. Newborn. On pop-stacks in series. *Utilitas Mathematica*, 19:129–140, 1981.

[8] Mihai Badoiu, Julia Chuzhoy, Piotr Indyk, and Anastasios Sidiropoulos. Low-distortion embeddings of general metrics into the line. In *Proceedings of Annual ACM Symposium on Theory of Computing*, pages 225–233, New York, NY, USA, 2005. ACM Press.

[9] Mihai Badoiu, Kedar Dhamdhere, Anupam Gupta, Yuri Rabinovich, Harald Racke, R. Ravi, and Anastasios Sidiropoulos. Approximation algorithms for low-distortion embeddings into low-dimensional spaces. In *Proceedings of Annual ACM-SIAM Symposium on Discrete Algorithm*, pages 119–128, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[10] Dietmar Berwanger, Anuj Dawar, Paul Hunter, and Stephan Kreutzer. Dag-width and parity games. In *The 23rd International Symposium on Theoretical Aspects of Computer Science*, pages 524–536, 2006.

[11] Miklós Bóna. A survey of stack-sorting disciplines. *Electr. J. Comb.*, on(2), 2002.

[12] Prosenjit Bose, Jonathan F. Buss, and Anna Lubiw. Pattern matching for permutations. *Inf. Process. Lett.*, 65(5):277–283, 1998.

[13] J. Bourgain. On lipschitz embeddings of finite metric spaces in hilbert spaces. *Israel Journal of Mathematics*, 52:46–52, 1985.

[14] William John Brinkman. *Metric Space Emebeddings into l_1: An optimization approach.* PhD thesis, Princeton University, November 2004.

[15] Gruia Calinescu, Cristina G. Fernandes, and Bruce Reed. Multicuts in unweighted graphs and digraphs with bounded degree and bounded tree-width. *Journal of Algorithms*, 48(2):333–359, 2003.

[16] Douglas E. Carroll, Ashish Goel, and Adam Meyerson. Embedding bounded bandwidth graphs into l1. In *The 33rd International Colloquium on Automata, Languages and Programming*, pages 27–37, 2006.

[17] Nishanth Chandran, Ryan Moriarty, Rafail Ostrovsky, Omkant Pandey, and Amit Sahai. Improved algorithms for optimal embeddings. *Electronic Colloquium on Computational Complexity (ECCC)*, (TR06-110), 2006.

[18] Moses Charikar, Konstantin Makarychev, and Yury Makarychev. Directed metrics and directed graph partitioning problems. In *Proceedings of Annual ACM-SIAM Symposium on Discrete Algorithm*, pages 51–60, New York, NY, USA, 2006. ACM Press.

[19] Chandra Chekuri, Anupam Gupta, Ilan Newman, Yuri Rabinovich, and Alistair Sinclair. Embedding k-outerplanar graphs into ℓ_1. In *Proceedings of Annual ACM-SIAM Symposium on Discrete Algorithm*, pages 527–536, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[20] Julia Chuzhoy and Sanjeev Khanna. Polynomial flow-cut gaps and hardness of directed cut problems. In *Proceedings of Annual ACM Symposium on Theory of Computing*, New York, NY, USA, 2007. ACM Press.

[21] D.G. Corneil, H. Lerchs, and L. Stewart Burligham. Complement-reducible graphs. *Discrete Applied Math.*, 3:163–174, 1981.

[22] R. Diestel. *Graph Theory.* Springer, 3rd edition, 2005.

[23] S. Even and A. Itai. Queues, stacks, and graphs. In *Theory of Machines and Computations, Z. Kohavi and A. Paz, Eds.*, pages 71–86, New York, NY, USA, 1973. Academic Press.

[24] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

[25] Georg Gottlob, Martin Grohe, Nysret Musliu, Marko Samer, and Francesco Scarcello. Hypertree decompositions: Structure, algorithms, and applications. In Dieter Kratsch, editor, *Proceedings of the 31st International Workshop on Graph-Theoretic Concepts in Computer Science (WG05)*, volume 3787 of *Lecture Notes in Computer Science*, pages 1–15. Springer-Verlag Berlin Heidelberg, 2005.

[26] Georg Gottlob, Nicola Leone, and Francesco Scarcello. Hypertree decompositions and tractable queries. *Journal of Computer and System Sciences*, 209:1–45, 2002.

[27] Georg Gottlob, Francesco Scarcello, and Martha Sideri. Fixed-parameter complexity in ai and nonmonotonic reasoning. *Artif. Intell.*, 138(1-2):55–86, 2002.

[28] Anupam Gupta, Alistair Sinclair, Ilan Newman, and Yuri Rabinovich. Cuts, trees and ℓ_1-embeddings of graphs. *Combinatorica*, 24(2):233–269, April 2004.

[29] Alexander Hall and Christos Papadimitriou. Approximating the distortion. In *APPROX-RANDOM*, pages 111–122, 2005.

[30] Roger A. Horn and Charles R. Johnson. *Matrix analysis.* Cambridge University Press, New York, NY, USA, 1986.

[31] Paul Hunter and Stephan Kreutzer. Digraph measures: Kelly decompositions, games, and orderings. In *Proceedings of Annual ACM-SIAM Symposium on Discrete Algorithm*, New York, NY, USA, 2007. ACM Press.

[32] Paul Hunter and Stephan Kreutzer. Digraph measures: Kelly decompositions, games, and orderings. *Theoretical Computer Science*, 2007. Submitted.

[33] P. Indyk. Algorithmic applications of low-distortion geometric embeddings. In *Proceedings of IEEE Symposium on Foundations of Computer Science*, page 10, Washington, DC, USA, 2001. IEEE Computer Society.

[34] T. Johnson, N. Robertson, P. D. Seymour, and R. Thomas. Directed Tree-Width. *Journal of Combinatorial Theory(Series B)*, 82:128–154, 2001.

[35] T. Johnson, N. Robertson, P. D. Seymour, and R. Thomas. Addendum to "Directed Tree-Width". manuscript, 2002.

[36] Claire Kenyon, Yuval Rabani, and Alistair Sinclair. Low distortion maps between point sets. In *Proceedings of Annual ACM Symposium on Theory of Computing*, pages 272–280, New York, NY, USA, 2004. ACM Press.

[37] T. Kloks. *Treewidth, Computation and Approximation.* Berlin : Springer-Verlag, 1994.

[38] Donald E. Knuth. *Art of Computer Programming, Volume 1: Fundamental Algorithms.* Addison Wesley Professional, 1973.

[39] Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15:215–245, 1995.

[40] Jan Obdrlek. Dag-width: connectivity measure for directed graphs. In *Proceedings of Annual ACM-SIAM Symposium on Discrete Algorithm*, pages 814–821, New York, NY, USA, 2006. ACM Press.

[41] H. Okamura and P. D. Seymour. Multicommodity flows in planar graphs. *Journal of Combinatorial Theory (Series B)*, 31:75–81, 1981.

[42] Vaughan R. Pratt. Computing permutations with double-ended queues, parallel stacks and parallel queues. In *Proceedings of Annual ACM Symposium on Theory of Computing*, pages 268–277, New York, NY, USA, 1973. ACM Press.

[43] Satish Rao. Small distortion and volume preserving embeddings for planar and euclidean metrics. In *SCG '99: Proceedings of the fifteenth annual symposium on Computational geometry*, pages 300–306, New York, NY, USA, 1999. ACM Press.

[44] B. Reed. Tree width and tangles: A new connectivity measure and some applications. *Suervey in Combinatorics*, 241:87–158, 1997.

[45] B. Reed. Introducing directed tree width. In H.J. Broersma, U. Faigle, C. Hoede, and J.L. Hurink, editors, *Electronic Notes in Discrete Mathematics*, volume 3. Elsevier, 2000.

[46] B. Reed, N. Robertson, P. Seymour, and R. Thomas. On packing directed circuits. *Combinatorica*, 16:535–554, 1996.

[47] M.A. Safari. Directed tree-width. Master's thesis, School of Computer Science, University of Waterloo, 2003.

[48] Mohammad Ali Safari. D-width: A more natural measure for directed tree width. In Joanna Jedrzejowicz and Andrzej Szepietowsk, editors, *Proceedings 30th International Symposium on Mathematical Foundations of Computer Science, MFCS'05, Lecture Notes in Computer Science, volume 3618*, pages 745–756, Berlin, 2005. Springer-Verlag.

[49] James H. Schmerl and William T. Trotter. Critically indecomposable partially ordered sets, graphs, tournaments and other binary relational structures. *Discrete Math.*, 113(1-3):191–205, 1993.

[50] P. Seymour and R. Thomas. Graph searching, and a min-max theorem for treewidth. *Journal of Combinatorial Theory (Series B)*, 58:239–257, 1993.

[51] Stefan Szeider. On fixed-parameter tractable parameterizations of sat. In *SAT*, pages 188–202, 2003.

[52] Robert Tarjan. Sorting using networks of queues and stacks. *J. ACM*, 19(2):341–346, 1972.

[53] D. H. Younger. Graphs with interlinked directed circuits. In *Proceedings of the Mid-west Symposium on Circuit Theory*, volume 2, pages XVI 2.1 – XVI 2.7, 1973.