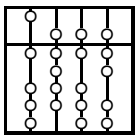
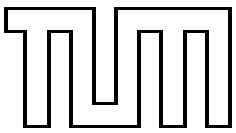


TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK



Lehrstuhl für Effiziente Algorithmen

Complexity Analysis of Tries and Spanning Tree Problems

Bernd Stefan Eckhardt

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr. J. Schlichter
Prüfer der Dissertation:	
1.	Univ.-Prof. Dr. E. W. Mayr
2.	Univ.-Prof. A. Kemper, Ph. D.

Die Dissertation wurde am 13.12.2007 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 23.11.2009 angenommen.

Abstract

Much of the research progress that is achieved nowadays in various scientific fields has its origin in the increasing computational power and the elaborated mathematical theory which are available for processing data. For example, efficient algorithms and data structures play an important role in modern biology: the research field that has only recently grown out of biology and informatics is called bioinformatics. String and prefix matching operations on DNA or protein sequences are among the most important kinds of operations in contemporary bioinformatics applications. The importance of those operations is based on the assumption that the function of a gene or protein and its sequence encoding are strongly related. Clearly, not all kinds of data that appear in bioinformatics can be modeled as textual data, but in many situations data are more appropriately modeled by networks, e.g., metabolic networks or phylogenetic networks. For such kinds of data algorithmic network analysis, i.e., applying graph algorithms to compute the relationship between the networks elements, is an indispensable tool in bioinformatics. In this thesis, we consider two fundamental computational problems which are important in this and other contexts.

In the first part of this thesis we try to give an answer to the following question: given a graph, how well can the distance metrics induced by the graph be approximated by the distance metrics induced by its spanning trees? More precisely, the combinatorial optimization problem which we study is the following: given a real-valued similarity measure rating the degree of correct approximation of the distance metrics of a graph by the distance metrics of a spanning tree, find for an input graph G a spanning tree T which is optimal with respect to the given similarity measure. We consider the standard matrix norms $\|.\|_{L,p}$ (for $1 \leq p < \infty$), $\|.\|_{L,\infty}$, $\|.\|_1$, and $\|.\|_\infty$ applied to the distance matrix of the tree and to the difference of the distance matrix of the tree and the graph as similarity measures. We also consider the vector norms $\|.\|_p$ applied to the difference of the closeness centrality vector of the graph and the tree as similarity measure. We prove that all versions of the problems which we consider are hard. For one version we give a polynomial-time 2-approximation algorithm. Distances and centralities are fundamental measures for *network analysis*. Besides this, approximating graph metrics by tree metrics has applications in *network design* and *combinatorial optimization*: we particularly consider an application from the area of bioinformatics, i.e., a version of the multiple sequence alignment problem.

In the second part of this thesis, we consider a fundamental data structure, i.e., the trie, which is frequently being used for text processing tasks, particularly for *string and prefix matching*. Tries (and trie-like data structures) are among the most basic and simple data structures for such tasks and nevertheless are very efficient in practice. Therefore, they are being used in many applications, ranging from IP-package classification in routers and fast string sorting over the inverted index in search engines to bioinformatics applications such as *homology searches in (distributed) DNA data bases*. Given the good practical performance of

tries even on non-random data, e.g., *DNA sequences* or *words in the dictionary of a natural language*, one is interested in a mathematically sound explanation for these findings. The most crucial parameter of a trie is its height which is according to experimental findings approximately logarithmic in the number of items stored in the trie although it is unbounded in the worst-case. Previous average-case analyses only can give such an explanation under the assumption that the inputs are generated by some random mechanism. Typically, an analysis which requires weaker assumptions is more sound and therefore desirable. Thus, the two questions that we try to answer in this context are: can we give an explanation for the practical findings without making any assumptions about the existence of a (stationary and ergodic) random source that approximates the inputs but instead of this under weaker assumptions? How well do tries perform on inputs that are near worst case? To answer these questions, we perform a smoothed analysis of trie height. The perturbation model subject to which the smoothed analysis is performed is based on (Mealy-type) probabilistic finite automata. The result of our smoothed analysis supports the practical findings and also yields that worst case inputs are isolated peaks in the input space: namely, we show that small random perturbations suffice to turn worst-case inputs into such inputs for which a trie has logarithmic expected height and we quantify the relation between the smoothing parameters and the trie height.

Acknowledgments

First and foremost, I thank my advisor Ernst W. Mayr for his support throughout the time of research and writing this thesis at the *Lehrstuhl für effiziente Algorithmen*. Furthermore, I am thankful to all my research colleagues and the current and former members of this facility. I appreciate the numberless discussions with Matthias Baumgart, Klaus Holzapfel, Riko Jacob, Moritz Maaß, Johannes Nowak, Sebastian Wernicke, and, particularly, with Sven Kosub and Hanjo Täubig. Also, I thank my father and all colleagues involved for proof reading. Finally, I am deeply grateful to my wife Sophie, my parents, and all my friends for their personal support during this time.

Contents

List of Figures

Chapter 1

Introduction

Μαθημάτων φρόντιζε μᾶλλον χρημάτων · τὰ γὰρ μαθήματ' ευπορεῖ τὰ χρηματα

Φιλεμον[1]

1.1 Analysis of tries and spanning tree problems

During the last three or four decades the amount of data that is being produced in the various scientific fields has tremendously increased. Much of the research progress that is achieved nowadays has its origin in the increasing computational power and the elaborated mathematical theory which are available for processing these data. For example, efficient algorithms and data structures play an important role in modern biology: the research field that has only recently grown out of biology and informatics is called bioinformatics. String (and prefix) matching operations on DNA or protein sequences are among the most important kinds of operations in many bioinformatics applications. The importance of those operations is based on the assumption that the function of a gene or protein and its sequence encoding are strongly related. Clearly, not all kinds of data that appear in bioinformatics can be modeled as textual data, but in many situations data are more appropriately modeled by networks. This holds particularly for relational data such as phylogenetic networks or metabolic networks. For such kinds of data algorithmic network analysis, i.e., applying graph algorithms to compute the relationship between the network elements, is an indispensable tool in bioinformatics. In this thesis, we consider two fundamental computational problems which are important in this and other contexts. The thesis is organized into two main parts. In the first part, we consider the complexity of finding spanning trees that approximate the distance metrics of a given graph. In the second, we consider a fundamental data structure, i.e., the trie, which is frequently being used for string matching operations, and perform a smoothed analysis of its most important parameter, i.e., its height.

The further parts of this introduction are organized as follows: in Section 1.1.1, we give a more detailed overview on some of the potential applications and our results on spanning trees. In Section 1.1.2, we describe some applications where string matching operations are important, argue why previous average-case analyses are often insufficient to explain the good practical performance, and describe the goals and results of our smoothed analysis.

[1] Philemon of Syracuse, around 300 B.C.

1.1.1 Approximating graph metrics by tree metrics

Networks appear in many different contexts in contemporary sciences. Informally, a network is a pair consisting of a set of elements (sites) together with a set of relations (links) between the elements (connecting the sites). It is natural to represent a network as a graph $G = (V, E)$ consisting of a set V of vertices, representing the elements, and a set E of edges connecting pairs of vertices, representing the relations between the elements. In the first part of this thesis we try to give an answer to the following question: given a graph, how well can the distance metrics induced by the graph be approximated by the distance metrics induced by its spanning trees? More precisely, the combinatorial optimization problem which we study is the following: given a real-valued similarity measure rating the degree of correct approximation of the distance metrics of a graph by the distance metrics of a spanning tree, find for an input graph G a spanning tree T which is optimal with respect to the given similarity measure. Particularly, we consider the standard matrix norms $\|.\|_{L,p}$ (for $1 \leq p < \infty$), $\|.\|_{L,\infty}$, $\|.\|_1$, and $\|.\|_\infty$ applied to the distance matrix of the tree and to the difference of the distance matrix of the tree and the graph as similarity measures. We also consider the vector norms $\|.\|_p$ applied to the difference of the closeness centrality vector of the graph and the tree as similarity measure. Distances and centralities are fundamental measures for *network analysis*. Besides this, approximating graph metrics by tree metrics has applications in *network design* and *combinatorial optimization*.

1.1.1.1 Motivation

Abstracting networks for network analysis Many complex biological relationships are modeled as *biological networks*: examples are phylogenetic networks or metabolic networks. For the latter kind of networks there are various kinds of elements (molecules, proteins) and relations (metabolic interactions) between these elements. An important contribution of informatics in order to understand the functionality of those networks is the *(algorithmic) network analysis* (see [BE05] for an introduction): such an approach aims at applying algorithms, usually graph algorithms, to compute the relationship between the network's elements on different levels of aggregation. Depending on the functionality of the network that one is interested in, various measures have been proposed relevant: an example for a (local) network measure is the centrality of a site, where there are several centrality concepts, e.g., betweenness centrality or closeness centrality. Knowing the centrality of an element helps in understanding how important the element is in the network. Also, many characteristics of networks are expressed in their distance matrix. Typical questions that are answered in the context of network analysis are: what is the functionality of a specific molecule in a metabolic network? What is the evolutionary distance between two species in a phylogenetic network? In many settings it is desirable to consider instead of the whole network a sub-network which captures the essential parts of the given network. Finding such *network backbones* is a non-trivial algorithmic problem, which can be stated as the following *combinatorial network abstraction problem* (cf. [EKM$^+$05a]): let $\mathcal{P}$ be a class of pattern graphs and let ϱ be a real-valued similarity measure rating the degree of correct approximation of a given graph G by a subgraph $H \subseteq G$. For a fixed pattern class $\mathcal{P}$ and a fixed similarity measure ϱ, the optimization problem is to find for an input graph G a subgraph H which belongs to $\mathcal{P}$ such that $\varrho(H, G)$ is minimal. Examples for the class $\mathcal{P}$ are trees, planar graphs, k-connected graphs or graphs not having a specific graph minor. Notably, the dual problem of fixing $\varrho(H, G)$ and finding a graph class

$\mathcal{P}$ that meets this constraint has been considered extensively in literature under the notion of so-called *graph spanner problems*. From an algorithmic point of view trees are particularly interesting as pattern class because many of the algorithmic problems are easier on trees than on arbitrary network topologies. Clearly, network models are important in many other contexts besides bioinformatics, particularly in social science: examples of social networks are *scientific collaboration networks* or *co-authorship networks*. Here, essentially the same kinds of questions have to be answered, e.g.: what is the distance between two scientists in a scientific collaboration network? Another important application for the problems which we consider is the *design of infrastructure networks*.

Network design Computer networks (see [Tan96] for an introduction) provide the infrastructure over which many different communication tasks can be accomplished. Most of these tasks can be abstracted as message exchanges or data flows between different sites. Basic questions that one is faced with in *network design* are: along which paths should messages be sent through the network (Routing)? How should the data flow be organized once the paths are fixed (Transmission)? How should the network topology be designed in order to meet certain requirements? In reality, it is very difficult – if not impossible – to completely re-organize an existing computer network, and therefore questions of the last kind are particularly interesting for the design of so-called *overlay networks*. One of the important characteristics of overlay networks is that the topology of the underlying network is invisible to the applications that run on top of them. Therefore, it is possible to organize overlay networks in an almost arbitrary manner, i.e., using an arbitrary topology. Peer-to-peer (P2P) file-sharing networks, distributed hash tables (DHTs), or virtual private networks (VPNs) are examples of overlay networks. Spanning subnets – also called spanners – are one important tool in this context: those constructions usually are evaluated along the three worst-case measures *space*, *approximation quality* and *construction time*. Besides this, the topology of a network is important for efficient communication. Spanning trees not only are the sparsest subnets and therefore extremely space efficient, but they have additional advantages over other graph structures, e.g., with respect to routing: for example, Thorup and Zwick [TZ01] have shown that it is possible to route in a tree on n vertices such that the routing information at the packages, i.e., their routing label size, is $(1 + o(1)) \log_2 n$, no additional routing information is required and a routing decision at an intermediate computer takes constant time. Now, since communication cost can be expressed in terms of path lengths, it is desirable to choose spanning trees which minimize path lengths. Such spanning trees have been considered under the notion of minimum route cost spanning trees (MRCT) and optimum communication spanning trees (OCT) [Hak64, Hu74] and it is known that both problems are hard, i.e., their decision versions are NP-complete [JLRK78]. There exists a polynomial-time approximation scheme for the MRCT problem which is due to Wu et al. [WLC+98]. For those problems, the quality of a spanning tree is measured by $\sum_{u,v \in V} r(u,v) \cdot d_T(u,v)$, where $r : V \times V \to \mathbb{R}_+$ is a requirement function and $d_T(u,v)$ measures the distance in T between u and v. A last field where the problems which we consider find applications is *combinatorial optimization*.

Combinatorial optimization and bioinformatics Graphs are frequently used within problem reductions in combinatorial optimization: representing mathematical optimization problems or their solution spaces as graphs in order to solve the original problems by solving the corresponding graph problems or by searching their solution spaces in a more efficient

manner is a standard approach. Again, it is often desirable to consider spanning trees which approximate the graph metrics: one example which fits into this framework is the $(2-\frac{2}{n})$-factor approximation algorithm for the multiple sequence alignment problem by Gusfield [Gus93], which extends upon an approach originally developed by Feng and Doolittle [FD87]: the graph which is considered there is the complete graph on all sequences to be aligned. The construction of the approximate alignment is guided by a center-star spanning tree on this graph. This was improved by Pevzner [Pev92] to $2-\frac{3}{n}$ and by Bafna, Lawler and Pevzner [BLP97] to $2-\frac{r}{n}$ for any fixed r, again using the connection between the MRCT problem and multiple sequence alignments.

1.1.1.2 Our contribution: approximating graph metrics by tree metrics

We give a short informal overview of the results which we present in Chapter 3: the norms which we use to measure the similarity between the distance matrix of the tree and the distance matrix of the graph are reviewed in Section 2.2.1. In the context of network analysis we contribute the following with respect to the above defined combinatorial network abstraction problem: we consider trees as pattern class, and we focus on distance-based similarity measures computed by the standard matrix norms applied to the difference of the distance matrix of the tree and the graph. The problems are hard with respect to all norms. We also consider the problem of finding a spanning tree which approximates the closeness centrality vector of a given graph. Note that the closeness centrality vector of a graph can be defined as a function of the distance matrix of the graph and therefore these problems fit into the introduced framework of similarity measures based on distance metrics. Unfortunately, we can not break the curse of NP-completeness and have not (yet) found an approximation algorithm for any version of those problems. Our contribution to network design and the mentioned combinatorial optimization problem is the following: we generalize the hardness results of the MRCT problem to spanning trees which minimize routing costs under the measures computed by the standard matrix norms applied to the distance matrix of the tree. All versions are hard, except for the version with respect to the $\|.\|_{L,\infty}$ matrix norm which has already been considered in the literature under the notion of minimum diameter spanning trees. We give an efficient, i.e., $O(n^2 \log n + nm)$ time, 2-approximation algorithm for the version with respect to the $\|.\|_{L,p}$ norms for arbitrary $1 \leq p < \infty$. Besides the fact that these versions of the problems are interesting from the network design point of view, they are particularly interesting for the mentioned combinatorial optimization problem from the area of bioinformatics: in Section 3.4, we show how a version of the multiple sequence alignment problem under a generalized sum-of-pairs objective can be modeled and solved using our framework of distance minimizing spanning trees with respect to the $\|.\|_{L,p}$ matrix norms.

1.1.2 Understanding the practical performance of tries

Relational data, which as we described can often be modeled by networks, constitutes an important class of data in bioinformatics and other sciences. Thanks to the increasing computational power and elaborated mathematical theory of networks many different tasks can be accomplished using this type of representation. Yet, there are many cases in which data are not of relational type but are given as or can be modeled as *textual data*. Therefore, understanding the efficiency of data structures which are used in such tasks is indispensable. Tries are among the most basic and simple data structures for string matching. The most

crucial parameter of such data structures is their height which is according to experimental and practical findings approximately logarithmic in the number of items stored in the data structure although it is unbounded in the worst-case. Previous average-case analyses only can give such an explanation under the assumption that the inputs are generated by some random mechanism. Typically, an analysis which requires fewer assumptions is more sound and therefore desirable. Thus, the two questions that we try to answer in this context are: can we give an explanation for the practical findings without making any assumptions about the existence of a (stationary and ergodic) random source that approximates the inputs but instead of this under weaker assumptions? How well do tries perform on inputs that are near worst case? To answer these questions, we perform a smoothed analysis of trie height. In the following, we give some motivations why these questions are interesting and shortly sketch the goals and results of our smoothed analysis.

1.1.2.1 Motivation

String matching in computer networks and bioinformatics Textual data are ubiquitous in modern computing: on the one hand, this clearly holds for the enormous amount of text documents, which are nowadays mainly accessed, distributed, processed, and organized on top of computer networks. On the other hand, this also holds for the equally large collection of *DNA or protein sequences* from various kinds of organisms that are stored in DNA or protein data bases. From an abstract point of view those documents are nothing but collections of strings over finite sets of symbols. Along the most fundamental of all text processing operations are *string and prefix matching*. For the first kind of matching, we are given a string t and a collection of strings S, and we want to determine whether t *exactly* matches one of the strings in S. For the second kind of matching, we are interested whether or not there is a string $s \in S$ such that t is a prefix of s. In computer networks, these matching operations are either user triggered or automatically performed by the class of algorithms that run the network: examples of user triggered string matching operations are queries that are sent to search engines like Google[2]; string or prefix matching tasks which are performed automatically include IP-package classifications at routers in the network layer of the Internet [NK99, Tan96, GT01], detection of malicious code in intrusion detection systems [Mar99, CSTV04], routing in *Content Addressable Networks* (CANs), where messages are not necessarily routed through the overlay network using IP-addressing, but where it is possible to address computers and perform routing by more or less arbitrary strings, and searching in peer-to-peer file sharing applications like Napster[3] or Gnutella[4]. In bioinformatics, such tasks are frequently found in homology searches in DNA or protein databases. See, e.g., [Gus97] for an introduction into string processing and its applications in bioinformatics. Note that nowadays these data bases are no longer located on one computer, but there are efforts being made to implement distributed versions (see, e.g., [SOUM05] for an example of such a distributed P2P-based architecture for homology searching).

Tries: practical findings and analytical investigations *Tries* and trie-like data structures such as b-tries, level-compressed-tries (LCP tries), PATRICIA trees and suffix trees are among the most basic and simple data structures for such tasks and nevertheless are very

[2]http://www.google.com
[3]http://www.napster.com
[4]http://www.gnutella.com

efficient in practice: clearly, trie-like data structures can be used to implement the inverted index of a search engine or a data base for homology searches. Notably, tries and LCP-tries even were used for an efficient implementation of IP-routing tables in [DBCP97, NK99]. Besides this, most fast string sorting algorithms are based on tries [BS97, SZ03, AN98]. Tries have the obvious advantage over hash tables that they allow for prefix matching and range queries, whereas the latter only support exact string matches because they are designed to distribute the keys equally over the whole hash table. Clearly, tries also suffer some drawbacks when compared to other data structures for text processing tasks. This particularly concerns their size: according to experimental investigations ordinary tries consume up to 100% more memory than, e.g., hash tables. Nevertheless, there are efficient implementations which do no longer suffer this drawback: Heinz, Zobel and William [HWZ02] have implemented b-tries and (experimentally) have shown that their version consumes only 25% more memory than a hash table. Another more serious obstacle with tries is that – theoretically – the worst-case performance of the retrieval operation is very poor.

In analytic models a trie is usually built over a set S of n *infinite* (identically distributed) random strings over a finite alphabet. See [Szp01] for an introduction into the analytic average case analysis of text processing algorithms and data structures. The parameters of a trie, e.g., its size, average depth, or height, can be viewed real-valued random variables over the probability space generated by finite prefixes of the random strings in the set S. For example, the height H_S of a trie built over the set S then equals the length of the longest common prefix of *any* two strings in S. Most analytic studies of random tries suggest that $H_S \overset{\text{w.h.p}}{\to} c \cdot \log_d n$ for two positive constants c and d which depend on the parameters of the random string model and the version of the trie being analyzed. We review some of these approaches when we perform our smoothed analysis of trie height in Chapter 4. Note that even better upper bounds of $\Theta(\log \log n)$ and $\Theta(\log^* n)$ are known for random LCP-tries [AN93, Dev01, DS05]. These findings are in contrast to a worst-case height of $\Omega(n)$.

Depending on the real world application, the previous analyses of trie height and other trie parameters do give a mathematically sound explanation for the good practical performance of these data structures: e.g., Markovian sources or hidden Markov models have been used to produce random time series which approximate the *statistical characteristics* of real-world time series such as natural language, DNA sequences, or the time series of various climatic parameters [BW90, MLMBdC05]. Thus, if some statistical information is known then one can model the respective real-world sequence using an appropriate random source and perform an average-case analysis with respect to that source. But it is not immediately clear that the mentioned analytic results can utterly explain the fact that in many practical settings where nothing is known about the statistics of the input sequences the height is still approximately logarithmic in the number of strings. Besides this, even for situations in which such models exist, the average-case analyses (and even the results that give sharper bounds on the higher moments of the distribution of H_S) cannot predict the performance of such data structures on an input that is very close to worst-case. An analysis of trie parameters which yields essentially the same results without making any assumptions about the existence of a random source that approximates the inputs but which is performed with respect to some weaker assumption seems desirable. This leads us to the already mentioned questions that are addressed in the second part of this thesis: can we give a sound explanation for the practical findings which requires fewer assumptions? How well do tries perform on inputs that are near worst case?

1.1.2.2 Our contribution: smoothed analysis of trie height

To answer to these questions, we perform a smoothed analysis of the most basic data-structure for text processing, i.e., the trie, where we focus on trie height as the most important parameter. The paradigm of smoothed analysis is relatively new and seems to be suitable to answer our questions. When performing a smoothed analysis of an algorithm one measures the maximum over all inputs of the expected performance of the algorithm under slight random perturbations of the respective input. The purpose of such an approach is to find out if worst-case inputs are isolated peaks in the input space or plateaus. A positive answer to this question then allows for conclusions about the performance on real-world inputs, particularly, if those inputs are subject to random influences. Also, it does not require any assumptions about the distribution function of the inputs, but instead of this only a reasonable perturbation model: a perturbation function should be such that it assigns higher probabilities to inputs which are in the proximity of the perturbed input, i.e., it should be *concentrated around this input*. This implicitly assumes the existence of some distance metrics defined over the input space.

For strings, a natural distance metrics is *edit distance*. The perturbation functions which we consider are based on, i.e., can be represented by, *Mealy-type star-like probabilistic finite automata (PFAs)*. On the one hand, they generalize a natural class of string perturbation functions, namely random edit operations. For a reasonable choice of the perturbation parameters the conditional distribution function generated by such perturbations is concentrated around the perturbed input under the edit distance metrics. On the other hand, the perturbation functions are general enough such that the random influences which strings in different real-world settings are subject to can be modeled. The main technical contribution of this chapter is a characterization of the smoothed trie height depending on the transition probabilities of the representing PFA: for a star-like perturbation automaton, it is logarithmic if and only if certain conditions for the automaton's transitions hold; if the conditions do not hold then the height is unbounded (see Theorem 4.1). The logarithmic-unbounded height dichotomy is certainly not surprising, but the conditions are very easy to check. Also, we give a more precise quantitative characterization. The theorem can be applied to rather complex perturbation models for which an ad-hoc analysis using existing random string models is either quite involved or even out of reach. A direct consequence of the theorem is a proof of the logarithmic smoothed trie height under convex combinations of random edit perturbations (i.e., insertions, deletions, substitutions) if and only if the convex combination does not collapse to deletions. The results of these findings substantiate that worst-case inputs, i.e., sets that force tries into worst-case behavior, are isolated peaks in the input space, and even small deviations suffice to yield logarithmic expected trie height.

1.2 Thesis outline

In Chapter 2, we first review some basic concepts of the analytic approach to the analysis of algorithms: worst-case and average-case complexity, NP-completeness, and the O-Notation. We also give a brief introduction into the relatively new paradigm of smoothed analysis. Thereafter, we introduce the necessary notation. Finally, we give a short introduction into the subject of rational generating functions.

In Chapter 3, we consider the complexity of finding spanning trees of a given graph that approximate the distance metrics induced by the graph. We show the NP-completeness of

our problems by reduction using two gadgets. Reductions using the first gadget are from X3C (Exact-3-Cover). The other reductions are from 2HS (2-Hitting Set, also know as Vertex Cover). Both problems are well-known to be NP-complete [GJ79].

In Chapter 4, we perform our smoothed analysis of trie height. We assume that all strings are perturbed independently by the same perturbation function. After the exposition of the perturbation model, we show that the semi-random string model that results from random perturbations of non-random inputs is not trivially included in previous *purely* random string models. Previous analyses heavily rely on the fact that the sources satisfy the (strong) mixing property [Bra05] which implies the existence of the *Rényi's Entropy of second order* for sub-strings. As this assumption does not hold in general for the semi-random model which we consider, we can not follow the vein of previous calculations, but we have to develop a new machinery: to do so, we use multivariate rational generating functions to express the computations of the perturbing PFA. This approach, which is called the *weighted words model* (cf. [FS07]) seems to fit best the requirements of our analysis. This proof certainly is the technically most challenging of all the proofs that we present in this thesis.

1.3 Publications

The results of Chapter 3 have been published under the title "Combinatorial network abstraction by trees and distances" as technical report [EKM+05b]. An extended abstract with the same title has been published in the conference proceedings of ISAAC [EKM+05a]. Additionally to the NP-completeness results which are presented in this thesis these publications include also some non-approximability results for a restricted version of the DAST problems, called *fixed-edges* DAST problems. Particularly, for those versions it is assumed that a part of the edges must be in the spanning tree. These results, which are achieved by a recursive application of the 2HS gadget, were mainly derived by my co-author Sebastian Wernicke. Also, the proof of Lemma 3.2 is also due to this co-author. A journal version has been submitted to Theoretical Computer Science and is subject to revision. First results on the smoothed analysis of trie height which we present in Chapter 4 have been published as technical report [EKN07].

Chapter 2

Preliminaries

People who analyze algorithms have double happiness. First of all they experience the sheer beauty of elegant mathematical patterns that surround elegant computational procedures. Then they receive a practical payoff when their theories make it possible to get other jobs done more quickly and more economically

Donald E. Knuth [Knu00]

2.1 Analysis and complexity of computational problems

In this section, we review the basic concepts of the complexity analysis of algorithms and computational problems.

2.1.1 A general setup

Computers are machines which solve computational problems by means of algorithms and data-structures. An *algorithm* is a step-by-step procedure to solve a specific computational problem where the individual steps are either *elementary (basic) computer operations* or invocations of other algorithms. To this end, an algorithm takes some input and produces some output. A *data-structure* is a systematic way of organizing data which supports a set of operations to access and change the data that is organized in the data structure. When analyzing algorithms and data-structures, one measures their resource consumptions. For an algorithm the measures are *computation time* and *memory consumption*. For a data structure the measures are *space requirement*, *initialization time*, and *running times of the supported operations*.

Usually, the resource consumption of an algorithm or a data-structure is given as a function of the size of its input. How exactly the size on an input is defined, depends on the computational problem which is analyzed. For problems involving graphs, as the one which we consider in Chapter 3, the input size is given by a pair of numbers – the *number of vertices* and the *number of edges*. For the problem which we consider in Chapter 4 – building a trie over a set of virtually infinite strings – the size of the input is given by the cardinality of the set of strings over which the trie is built.

Now, computational problems are of different complexities: the *complexity of a computational problem* is given by the resource consumption of the best possible algorithm or data-structure for this problem and upper bounded by the best known algorithm or data-structure for this problem. An algorithm is said to be *efficient* if the algorithm operates economically

with the computational resources. Analogous to that, a data structure is an *efficient data structure*, if the operations which it supports are efficient and the space requirements are appropriate.

In the *analytic approach to the analysis of algorithms and data-structures*, the efficiency of an algorithm or a data-structure is not measured in absolute values, i.e., micro-seconds, processor cycles or mega-bytes on a certain existing computer. Rather, one fixes a computer model which supports a set of *primitive operations* and "implements" the algorithm or the data-structure in a pseudo-code using the respective primitive operations. Each of the primitive operations should be such that it can be executed on a *real computer* in constant time. The advantages of such an approach are obvious: first, it makes the analysis more robust because it is independent from the operating system, the processor speed, the word-size or other technological characteristics and environmental facts; second, it makes the analysis easier.

Here, we adopt the *uniform cost model*, where we assume that all memory locations are of the same size and all numbers and other items which are involved in the computations fit into a constant number of memory locations. Then, a primitive operation is an operation that involves a constant number of operands each of which fits one memory location. Another cost model is the *logarithmic cost model*, in which the cost of a primitive operation depends on the number of bits of the involved operands. This makes particularly sense when numbers can get arbitrarily large which is not the case in the computational problems considered in this work.

2.1.2 Asymptotic analysis

In most settings, one in not interested in the exact number of primitive operations but rather in the *asymptotic behavior* of the function measuring this number as a function of the input size. From an analytic point of view this can be justified by the fact that in almost every analysis upper (or lower) bounds are used. Besides this, constant factors do not depend on the input instance, but rather on the exact computational model which is used, the representation in which the instance is given, or the actual implementation in pseudo-code. Thus, one makes use of the O-notation. A formal definition of the terms involved in the O-notation is given below. Let $f, g : \mathbb{N}_+ \to \mathbb{R}_+$ be two functions.

- $f(n)$ is said to be in $O(g(n))$, if there exists constants $c \in \mathbb{R}_+$ and $n_0 \in \mathbb{N}_+$ such that for all $n \geq n_0$ it holds that $f(n) \leq c \cdot g(n)$. $g(n)$ is said to be in $\Omega(f(n))$ in this case.
- $f(n)$ is said to be in $o(g(n))$ if for every constant $c \in \mathbb{R}_+$ satisfying $c > 0$ there exists a constant $n_0 \in \mathbb{N}_+$ such that for all $n \geq n_0$ it holds that $f(n) \leq c \cdot g(n)$. Equivalently, $f(n)$ is in $o(g(n))$ if $\lim_{n\to\infty} f(n)/g(n) = 0$. In this case $g(n)$ is said to be in $\omega(f(n))$.
- If both $f(n)$ in $O(g(n))$ and $g(n)$ in $O(f(n))$ then $f(n)$ is said to be in $\Theta(g(n))$.

The *asymptotic cost of an algorithm* is then given by a function f such that its cost function is in $\mathcal{X}(f(n))$ for $\mathcal{X} \in \{o, O, \omega, \Omega, \Theta\}$.

2.1.3 Analysis of algorithms

Worst-case and average-case analysis To ease the presentation, we assume in this section that the measure we are interested in is the asymptotic running time of an algorithm. For the other cost measure, i.e., space requirement, similar considerations apply. Clearly, the running

times of an algorithm on two different inputs of the same size may vary tremendously. In order to analyze its behavior, there are two main approaches: *worst-case analysis* and *average-case analysis*. The two corresponding measures are called *worst-case running time* and *average-case running time*, respectively. In order to explain precisely the difference between these two measures, we need to introduce the following notation: let I be the set of all inputs, also called the *input space*, of an algorithm A and let $I^{(n)}$ be the set of all inputs that have size n and let $C : I \to \mathbb{R}_+$ be the function measuring the running time of A. Then the worst-case running time of A is defined as

$$\mathrm{WC}_A(n) =_{\mathrm{def}} \max_{x \in I^{(n)}} C(x)$$

and the average-case running time of A is defined as

$$\mathrm{AC}_A(n) =_{\mathrm{def}} \mathbf{E}[C(x)]\,.$$

Here, the expectation is taken with respect to some probability measure $\mu : I \to [0, 1]$. To perform a worst-case analysis, one needs a good understanding of the algorithm and the structure of the computational problem. Nevertheless, this kind of analysis is generally much easier than average-case analysis which is often quite involved even under the oversimplified assumption that all instances are equiprobable.

A new paradigm: smoothed analysis The meaning of worst-case analysis is clear. For a meaningful average-case analysis a knowledge of what constitutes a realistic instance is indispensable. Since this knowledge is hard to acquire and the unknown distribution over the input space has to be approximated, average-case analysis often suffers from the drawback that it is dominated by valid instances that are practically irrelevant. One approach to overcome these kinds of problems, that has been pioneered in the works of Santha and Vazirani [SV86], Blum and Spencer [BS95], and Feige [FK01a] on *semi-random input models*, has been introduced by Spielman and Teng in their seminal paper [ST01] (see also the journal version of the paper [ST04]): the paradigm of *smoothed analysis* was introduced, in order to explain the good practical performance of the shadow-vertex simplex algorithm, which is not inferable from results about its average complexity and which is opposed to its poor worst-case complexity. The worst-case running time of the simplex method is exponential, but nevertheless it performs well on many instances. Before, a number of researchers had shown that the average-case running time is polynomial for various probability distributions over the input space. Some of these authors had not only shown that the average-running time is polynomial, but they had derived even stronger results by giving sharper bounds on the higher moments of the distribution of the running time.

Nevertheless, all these analyses do not give a sound mathematical explanation for the practical findings. There are two drawbacks: first, with instances being purely random, there is a high proportion of unrealistic instances. Second, even knowing the tails of the distribution of the running time does not allow to draw conclusion on the "shape of the input space". Thus, even if almost all possible inputs are such that the algorithm behaves well on them, this does not give a clue about how the algorithm behaves on a typical input. As we already mentioned, it is very difficult to give a probability distribution which prefers typical inputs. Therefore, in smoothed analysis one aims at answering the following sightly different question: are instances that cause an algorithm to perform poorly plateaus in the input space, or are they isolated

peaks? To answer this question, one defines the smoothed running time on an algorithm A as

$$\mathrm{SC}_A(n) = \max_{x \in I^{(n)}} \mathbf{E}[C(P(x))]\,,$$

where $P : I \to I$ is a random mapping from the input space to the input space and for each $x \in I^{(n)}$ the expectation is taken with respect to the probability measure on I that is generated by the random mapping P on input x. In this context, P is called a *perturbation function*. In the context of the simplex algorithm Spielman and Teng have chosen P such that it produces small Gaussian perturbations of the input matrix. An informal definition of the smoothed complexity of an algorithm is the following.

Definition 2.1 (Smoothed complexity)**.** *The* smoothed complexity *of an algorithm is given by the maximum over all inputs of the expected running time of the algorithm under slight random perturbations of the respective input, where the value of the smoothed complexity is then measured as a function of both the input size and a perturbation parameter, which gives a quantitative measure of the term "slight".*

Coming back to the original task of giving a mathematical framework to better understand the behavior of an algorithm on a typical input, we must now argue that smoothed analysis provides such a framework. Clearly, performing a smoothed analysis as described above we cannot claim to produce a more relevant probability distribution, i.e., one that prefers typical instances to atypical ones. But knowing that the maximum over all inputs of the expected running time of the algorithm under slight random perturbations of the respective input is asymptotically much smaller than the worst-case running time does indeed allow for the following conclusion: the running time on a typical input is likely to be asymptotically near its smoothed complexity and thus much smaller that its worst-case running time. This holds particularly under the assumption that typical inputs are subject to perturbations and are not designed such that they force the algorithm into the worst-case scenario.

By performing the smoothed analysis under Gaussian perturbations of the inputs, Spielman and Teng gave such a mathematical framework to better understand the good practical performance of the simplex method because they showed that the shadow-vertex simplex algorithm has polynomial smoothed complexity, i.e, its running time is polynomial in the size of the input and the standard deviation of the (Gaussian) perturbation. In this particular case, the smoothed complexity is claimed to explain the good practical performance fairly well, because numerical inputs are likely to be subject to errors of different types.

Clearly, the paradigm of smoothed analysis is not limited to numerical computational problems, but it can also be applied to purely combinatorial problems. The *only* ingredients necessary to perform a meaningful smoothed analysis are: an adequate perturbation model and a thorough analysis of the expected running times under slight random perturbations using the respective perturbation model. What exactly constitutes an adequate perturbation function depends on the problem considered and is being subject to discussions [ST03a]. For the case of numerical data small Gaussian perturbations, which account for the random noise that real world inputs are subject to, could be a reasonable model of perturbations: nevertheless, the reader should also note that it has been criticized that the effects of the Gaussian perturbations which yield the good analytic results do often smooth the input in such a way that the problem specific characteristics of many LP-formulations are severely destroyed. In other settings, particularly in combinatorial settings, perturbation functions

should be preferred that preserve the most-significant properties of the input. In all cases, the random perturbations should resemble those random influences which real world inputs are typically subject to. A number of different works concerning the smoothed analysis of linear programming problems, numerical problems, and variants of the simplex algorithm followed the original work of Spielman and Teng. (see, e.g., [BD02, ST03b, SST06, AV06, Ver06] and the references therein). The smoothed analysis paradigm has also been applied to purely discrete optimization problems, particularly to ILPs (see, e.g., [BV06, RV07]). Additionally, the smoothed complexity of various more specific algorithmic problems, e.g., the height of binary search trees [MR05], ordering problems, such as left-to-right maxima counting, shortest path, and quicksort [BBM03], online algorithms [BLMS$^+$03, SS05], computational geometry [DadHR$^+$03, BadHS04, DS04], and bi-criteria optimization problems [ANRV07] has been investigated.

From an analytic point of view, there are two more points that should be added: first, that smoothed analysis interpolates between average-case and worst-case analysis because choosing as the perturbation function the identity gives the latter complexity measure and choosing as the perturbation function any probability distribution on $I^{(n)}$ that is independent from the respective input $x \in I^{(n)}$ gives the former complexity measure; second, that smoothed complexity can also be explained very well in the following *adversary model*: assume we are given an *oblivious adversary* that has full information about the algorithm and the perturbation function, but no control over the random perturbations; then the smoothed complexity is the expected complexity under exactly this adversary model.

2.1.4 Complexity of computational problems

Knowing about the complexity of an algorithm for a specific computational problem may give some insight into the structure of the problem or even into the complexity of a number of closely related algorithms for the same or very similar computational problems. But, from this kind of "local analysis" one can only infer that the problem can be solved *at least as efficiently* as the most efficient known algorithm can solve it. Nevertheless, there might either be an unknown but much more efficient algorithm that solves the problem, or an optimal algorithm is known, but it is not known to be an optimal algorithm. A more "global analysis" is given by the *complexity of a computational problem*, i.e., by the two functions that are an upper bound on the complexity of the best known algorithm and a lower bound on the complexity of the best possible algorithm that solve the problem.

A "(computational) problem" is formally expressed in terms of a relation $P \subseteq I \times S$, where the set I is called the *a set of problem instances or inputs* and the set S of *problem solutions*. For $x \in I$ and $y \in S$, $(x, y) \in P$ may then be interpreted as y is a solution to the instance x. In the most simple setting, one only wants to decided on input $x \in I$ whether or not there exists a solution $y \in S$ such that $(x, y) \in P$. Those kinds of problems are called *decision problems*. The answer is either 'Yes' or 'No'. In a *search problem*, the task is only little more complicated: one is not only interested in the answer, but also in a respective solution if the answer is 'Yes'. The third scenario is that of an *optimization problem*: here, the task is to find on input $x \in I$ the *best solution* $y^* \in S$ of all solutions y such that $(x, y) \in P$, where the quality is with respect to some measure.

Complexity of decision problems One approach to get a better understanding of the complexity of decision problems is to establish relations between the different problems. To

this end, reductions, complexity classes, and completeness are the basic tools. We review the basic concepts in this paragraph. For the case of decision problems, one makes use of the theory of *formal languages*[1]: let Σ be a finite alphabet. Given a decision problem $P \subseteq I \times S$, we may interpret Σ^* as the set of all possible instances, i.e., $I = \Sigma^*$. Let $\mathcal{L} \subseteq \Sigma^*$ be a language with the following characteristic function: for $x \in \Sigma^*$,

$$\chi_{\mathcal{L}}(x) = \begin{cases} 1 & \text{if } (\exists y \in S)\ (x,y) \in P \\ 0 & \text{if } (\forall y \in S)\ (x,y) \notin P \end{cases}$$

The language $\mathcal{L}$ completely characterizes P. Let $f : \mathbb{N}_+ \to \mathbb{N}_+$ and let $\mathcal{L} \subseteq \Sigma^*$ be some language. $\mathcal{L}$ is said to be *decided* in time $O(f(n))$ by an algorithm A, if for every $x \in \Sigma^*$ it holds that A stops in time $O(f(|x|))$ and outputs $A(x) = 1$ if $x \in \mathcal{L}$ and $A(x) = 0$, otherwise.

Definition 2.2 (Complexity class)**.** *Let $f : \mathbb{N}_+ \to \mathbb{N}_+$. The class* $\textsc{Time}(f(n))$ *is the collection of all languages for decision problems which are decided by an algorithm in time $O(f(n))$.*

The problems in the following class are considered to be solvable efficiently.

Definition 2.3 (P)**.** *Let P' be a decision problem with corresponding language $\mathcal{L}$. $\mathcal{L}$ is is said to be in* P *, if there exists an algorithm which decides $\mathcal{L}$ in time $O(n^k)$ for some fixed $k \in \mathbb{N}$. This is,* $\mathrm{P} = \cup_{k=0}^{\infty} \textsc{Time}(n^k)$.

Certificates are strings which testifies that a word is in a language. As an example, let $\mathcal{L}$ be the language of all boolean formulas in CNF. A certificate for an element $x \in \mathcal{L}$, i.e., a boolean formula x, is a assignment y for the variables which makes x true. A *verification algorithm* is an algorithm which takes as input two strings, one ordinary input string x and one certificate y. Such an algorithm is said to verify an input string x if there exists some certificate y such that $A(x,y) = 1$. The language that is verified by a verification algorithm is

$$\mathcal{L} = \{x \in \Sigma^* \ :\ \text{there exists } y \in \Sigma^* \text{ such that } A(x,y) = 1.\}$$

Note that the algorithm must only return $A(x,y) = 1$ if y is a certificate for x.

Definition 2.4 (NP)**.** *Let P' be a decision problem with corresponding language $\mathcal{L}$. $\mathcal{L}$ is said to be in the class* NP *if there exist a polynomial-time verification algorithm A and a constant c such that*

$$\mathcal{L} = \{x \in \Sigma^* \ :\ \textit{there exists a } y \textit{ satisfying } |y| \in O(|x|^c) \textit{ such that } A(x,y) = 1\}$$

A *reduction* from a language $\mathcal{L}_1$ to a language $\mathcal{L}_2$ enables us to establish a relation between the corresponding decision problems. A total function $f : \Sigma^* \to \Sigma^*$ is said to be *polynomial-time computable*, if there is an algorithm that, given $x \in \Sigma^*$, produces the output $f(x)$ such that the asymptotic running time of the algorithm is polynomial in $|x|$ and $|f(x)|$.

Definition 2.5 (Polynomial-time reduction)**.** *Let $\mathcal{L}_1 \in \Sigma^*$ and $\mathcal{L}_2 \in \Sigma^*$ be two languages. $\mathcal{L}_1$ is said to be* polynomial-time reducible *to $\mathcal{L}_2$ if there exists an* polynomial-time computable *function $f : \Sigma^* \to \Sigma^*$ such that for all $x \in \Sigma^*$ it holds that*

$$x \in \mathcal{L}_1 \Leftrightarrow f(x) \in \mathcal{L}_2.$$

[1] see Section 2.2.3 for a formal definition of the terms language and characteristic function

Clearly, if $\mathcal{L}_1$ is polynomial-time reducible to $\mathcal{L}_2$, then $\mathcal{L}_2 \in \mathrm{P}$ implies that $\mathcal{L}_1 \in \mathrm{P}$. The same applies for two problems in NP.

Definition 2.6 (NP-Completeness)**.** *Let P' be a problem with corresponding language $\mathcal{L}'$. $\mathcal{L}'$ is said to be* NP-complete *if $\mathcal{L}'$ is in* NP *and for every problem P'' with corresponding language $\mathcal{L}''$ such that $\mathcal{L}''$ is in* NP *there exists a polynomial-time reduction from $\mathcal{L}''$ to $\mathcal{L}'$.*

The expositions given in this paragraph are based on the expositions given in [Sch94].

Approximations of optimization problems For the case of optimization problems we assume that there exists some function $f : S \to \mathbb{R}_+$ measuring the cost or the quality of a solution: for example, in the graph theoretic decision problem Vertex Cover (see Section 3.2.1 for a formal definition), the inputs are of type (G, k) and one is asked whether or not the given graph G has a vertex cover of size at most k. The corresponding optimization problem is: given a graph G, find the smallest $k \in \{1, \ldots, n\}$ such that G has a vertex cover of size k. This is a *minimization problem*, i.e., we seek to find a solution $y^* \in S$ such that $f(y^*) = \min_{y \in S} f(y)$. Clearly, there are also *maximization problems* where we seek to find a solution $y^* \in S$ such that $f(y^*) = \max_{y \in S} f(y)$. Given an optimization problem $P \subseteq I \times S$ equipped with some measure f, we say that an algorithm A is a *polynomial time c-approximation algorithm for the problem P* if A has running time polynomial in the size of the input $x \in I$ and returns a feasible solution $y \in S$ such that

$$\max\left(\frac{f(y)}{f(y^*)}; \frac{f(y^*)}{f(y)}\right) \leq c,$$

where $y^* \in S$ is the optimal solution.

2.2 Notation and elementary concepts

2.2.1 Mathematical preliminaries

Throughout this work, let $\mathbb{N} = \{0, 1, 2 \ldots\}$ be the set of natural numbers and denote by $\mathbb{N}_+ = \mathbb{N} \setminus \{0\}$ the set of positive natural numbers. Also, let $\mathbb{R}$ be the set of reals and denote the set of all positive reals by $\mathbb{R}_+$. For a finite set A denote its cardinality by $\|A\|$ and let $\mathcal{P}^A$ be the set of all subsets of A. For two reals x, y satisfying $x < y$, the closed interval between x and y is denoted by $[x, y]$, the open interval between x and y is denoted by (x, y) and the two half-open intervals between x and y are denoted by $(x, y]$ and $[x, y)$, respectively.

In Chapter 3, we measure the similarity between a graph and its spanning trees by evaluating different kinds of matrix norms of distance matrices. To this end, we consider the following classical *matrix and vector norms*: for an n-dimensional real vector $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$, the L_p-norm for $1 \leq p < +\infty$ is defined as

$$\|x\|_p =_{\text{def}} \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}.$$

For an $n \times n$ matrix $A = (a_{ij})_{\substack{1 \leq i \leq n \\ 1 \leq j \leq m}} \in \mathbb{R}^{n \times m}$, the L_p norm for $1 \leq p < +\infty$ is defined as

$$\|A\|_{L,p} =_{\text{def}} \left(\sum_{i=1}^{n} \sum_{j=1}^{m} |a_{ij}|^p\right)^{1/p}.$$

Letting $p \to +\infty$, we arrive at the L_∞ norm which is defined as

$$\|A\|_{L,\infty} =_{\text{def}} \max_{\substack{1 \le i \le n \\ 1 \le j \le m}} |a_{ij}|.$$

Finally, we consider the maximum-column-sum norm and the maximum-row-sum norm which are defined as

$$\|A\|_1 =_{\text{def}} \max_{1 \le j \le m} \left(\sum_{i=1}^{n} |a_{ij}| \right)$$

and

$$\|A\|_\infty =_{\text{def}} \max_{1 \le i \le n} \left(\sum_{j=1}^{m} |a_{ij}| \right),$$

respectively. A number of well-known mathematical facts which we use in our calculations is given in the Appendix A.

2.2.2 Graphs

The basic notation concerning graphs can be found almost any text-book which deals with graphs and graph algorithms (see, e.g., [GT01, CLR01, Die05]).

Graphs An *undirected unweighted graph* $G = (V, E)$ is a pair such that V is a finite set of elements, called the *vertex set*, and E is a subset of all 2-elementary subsets of V, called the *edge set*. An *undirected (positive) weighted graph* $G = (V, E, w)$ is an undirected unweighted graph $G = (V, E)$ together with a function $w : E \to \mathbb{R}_+$, called the *weight function*. In this thesis, we only consider positive weighted graphs. A labeled graph $G = (V, E, l)$ is given by an unweighted graph $G = (V, E)$ together with some function $l : E \to A$ mapping edges to some countable set A, called the labeling function. Labeled graphs appear most often in the context of trees.

For an unweighted graph $G = (V, E)$ and an edge $\{u, v\} \in E$ the vertices u and v are called the endpoints. For a vertex $v \in V$, the number of incident edges is called the *degree of* v in G. Two vertices $u, v \in V$ are said to be *adjacent* if $\{u, v\} \in E$. For two vertices $v_1, v_k \in V$ an *undirected path* from v_1 to v_k in G is a sequence $(v_1, \ldots, v_k)$ such that $\{v_2, \ldots, v_{k-1}\}$ are distinct vertices and such that for all $i \in \{1, \ldots, k-1\}$ it holds that $\{v_i, v_{i+1}\} \in E$. The vertex v_1 is called the *starting point* of the path and the vertex v_k is called the *end point*. The *length of a path* in an unweighted graph is defined as the number of edges which constitute the path. The length of a path in a weighted graph is defined as the sum of the weights of its constituent edges. If the starting point of a path equals its end point, but all other vertices appear at most once in the sequence, then the path is called a *(simple) cycle*.

A graph $G = (V, E)$ is *connected* if for all pairs $u, v \in V$, there exists a path from u to v in G. For a connected graph $G = (V, E)$, either weighted or unweighted, and two vertices $u, v \in V$ the *distance* between u and v in G is defined as the minimum over all the lengths of all paths from u to v in G and is denoted by $d_G(u, v)$. If $u = v$ then $d_G(u, v) = 0$. Let $V = \{v_1, \ldots v_n\}$. For a graph G there exists a corresponding square matrix $D_G = (d_G(v_i, v_j))_{1 \le i \le j \le n}$, called the *distance matrix* of G. For undirected graphs, this matrix is symmetric with all entries being non-negative. The problem of computing the distance matrix of a given undirected positive weighted graph is called the *All-Pairs-Shortest-Path problem* is is well-known to be solvable

in time $O(n^2 \log n + nm)$ (cf. [CLR01]). Yet, there are some faster algorithms but for our purposes the given running time suffices.

A graph $G' = (V', E')$ is a *subgraph* of a graph $G = (V, E)$, denoted by $G' \subseteq G$, if both subset relations $V' \subseteq V$ and $E' \subseteq E$ hold. A graph $G' = (V', E')$ is said to be a *spanning subgraph* of a connected graph $G = (V, E)$ if G' is a subgraph of G such that $V' = V$ and G' is connected. For a undirected graph $G = (V, E)$ and a spanning subgraph $G' = (V, E')$ it clearly holds that

$$(\forall u, v \in V) \;\; d_G(u, v) \leq d_{G'}(u, v).$$

Trees A graph is a *tree*, if it contains no cycles. A cycle-free spanning subgraph is called a *spanning tree*. We denote the set of all spanning trees of a graph G by $SP(G)$. It is a well-known fact that a tree on n vertices has exactly $n - 1$ edges. Also, in a tree $T = (V, E)$ there exists for each pair of vertices $u, v \in V$ *exactly one path in* T. A *rooted tree* $T = (V, A)$ consists of a vertex set V and a subset A of all ordered pairs of elements from V, called *arcs*, and a designated vertex $r \in V$ such that for every vertex $u \in V \setminus \{r\}$ there is a directed path in T from r to v: a *directed path* from $r = v_1$ to $u = v_k$ is a sequence $(v_1, \ldots, v_k)$ of distinct vertices such that for all $i \in \{1, \ldots, k-1\}$ it holds that $(v_i, v_{i+1}) \in A$. For an arc (u, v), u is called the *parent* of v and u is called a *child* of v. A vertex that has no children is called a *leaf* of *external vertex* and vertices that have at least one child are called *internal vertices*. The *depth* of a vertex $v \in V$ is defined as the number of edges on the directed path between the root and v in T. The *height* of a rooted tree T is defined as the maximum depth of any external vertex in T.

2.2.3 Strings and regular languages

Strings In this work $\mathcal{A}$ denotes a finite set called the *alphabet*. The elements of $\mathcal{A}$ are called the *symbols* of the alphabet. For $m \in \mathbb{N}$ and $1 \leq i \leq m$, the finite *sequence* $s = (a_1, \ldots, a_m)$ of symbols $a_i \in \mathcal{A}$ is called a *finite string* over $\mathcal{A}$ of *length* m, denoted by $|s|$. If $m = 0$ then the string is called the *empty string* and is denoted by ϵ. An infinite sequence $s = (a_1, a_2, \ldots)$ of symbols such that for $i \in \mathbb{N}_+$ it holds that $a_i \in \mathcal{A}$ is called and *infinite string*. In this case, we set $|s| = +\infty$. We note here that $|.|$ is used in two different meanings within this work: if $x \in \mathbb{R}$, then $|x|$ denotes the modulus-function and if s is a string, then $|s|$ denotes the length of s. This "operator-overloading" will never cause any uncertainty, because it will be clear from the context which is the correct meaning.

A string $s = (a)$ of length one will be abbreviated by a. For a finite string s of length m and $i \in \{1, \ldots, m\}$ we access the i-th element a_i by $s[i]$, where for every string s it holds the $s[0] = \epsilon$. For a finite string s and $i, j \in \{1, \ldots, |s|\}$ satisfying $i \leq j$, the subsequence $(a_i, \ldots, a_j)$ is called a substring of s and is accessed by $s[i \ldots j]$. Here, for $i, j \in \{1, \ldots, m\}$ satisfying $i > j$ we define $s[i \ldots j] = \epsilon$ as the access to the empty string. If s is infinite, we access the infinite substring starting at the i-th position of s by $s[i \ldots]$.

For a symbol $a \in \mathcal{A}$ and a string s over the same alphabet we denote by $|s|_a$ the number of occurrences of the symbol a in s. For a finite string $s \in \mathcal{A}$, it clearly holds that $|s| = \sum_{a \in \mathcal{A}} |s|_a$. For a natural number m we denote by $\mathcal{A}^m$ the set of all strings over $\mathcal{A}$ that have length exactly m and by $\mathcal{A}^{\leq m}$ the set of all strings that have length at most m. Further, $\mathcal{A}^\infty$ denotes the set of all infinite strings over $\mathcal{A}$, $\mathcal{A}^{<\infty}$ denotes the set of all finite strings over $\mathcal{A}$, and $\mathcal{A}^{\leq\infty}$ denotes the set of all strings over $\mathcal{A}$.

Two non-empty strings s, t over the same alphabet are *identical*, denoted by $s = t$, if $|s| = |t|$ and for all indices $i \in \{1, \ldots, |s|\}$ it holds that $s[i] = t[i]$. Note that this includes the identity of infinite strings. A string s is a *prefix* of a string t, if $|s| \leq |t|$ and for all indices $i \in \{1, \ldots, |s|\}$ it holds that $s[i] = t[i]$. We write $s \sqsubseteq t$ in this case. A prefix s of t is a *proper prefix*, if $|s| < |t|$. We write $s \sqsubset t$ in this case. Note that for the empty string ϵ, $\epsilon \sqsubset t$ for every string t with $|t| > 0$.

Given two strings s, t over an alphabet $\mathcal{A}$, the *longest common prefix* of s and t is the longest string r over $\mathcal{A}$ that is a prefix of both s and t. We define the following function $\mathrm{lcp} : \mathcal{A}^{\leq\infty} \times \mathcal{A}^{\leq\infty} \to \mathbb{N}$, where lcp measures the length of the longest common prefix of two strings:

$$\mathrm{lcp}(s, t) = \max(j \in \mathbb{N} \ : \ s[0 \ldots j] \sqsubseteq t)$$

This function is particularly important in the setting of tries and trie-like data structures. For two strings s, t such that s is finite, the sequence $r = (s_1, \ldots, s_{|s|}, t_1, \ldots)$ is called the *concatenation of s and t* and is denoted by $r = s\ t$.

Regular languages Let $\mathcal{A}$ be an alphabet. A *language* $\mathcal{L} \subseteq \mathcal{A}^{\leq\infty}$ is a collection of strings over $\mathcal{A}$. Given a language $\mathcal{L} \subseteq \mathcal{A}^{\leq\infty}$, the function $\chi_{\mathcal{L}} : \mathcal{A}^{\leq\infty} \to \{0, 1\}$ such that for every $w \in \mathcal{A}^{\leq\infty}$ it holds that

$$\chi_{\mathcal{L}}(w) = \begin{cases} 1 & \text{if } w \in \mathcal{L} \\ 0 & \text{if } w \notin \mathcal{L} \end{cases}$$

is called the *characteristic function of $\mathcal{L}$*. A language $\mathcal{L} \subseteq \mathcal{A}^{\leq\infty}$ is called a *regular language* over $\mathcal{A}$ if all elements – in this particular case called *words* – of $\mathcal{L}$ are described by regular expressions over $\mathcal{A}$: formally, given an alphabet $\mathcal{A}$ the following are *regular expressions over $\mathcal{A}$*:

- For all $a \in \mathcal{A}$, 'a' is a regular expression. The language corresponding to this regular expression is $\{a\}$.
- For two regular expressions R and Q with corresponding languages R and Q, '$R \mid Q$' is a regular expression with the corresponding language $R \cup Q$.
- For two regular expressions R and Q with corresponding languages R and Q, '$R\ Q$' is a regular expression with the corresponding language

$$\{w \ : \ w = u\ v \text{ for } u \in R \text{ and } v \in Q\}.$$

- For a regular expression R with corresponding language R, 'R^+' is a regular expression with language

$$\{w_1\ w_2\ \ldots\ w_m \ : \ m \in \mathbb{N}_+ \text{ and for all } i \in \{1, \ldots, m\},\ w_i \in R\}.$$

- For a regular expression R with corresponding language R, 'R^*' is a regular expression with language

$$\{\epsilon\} \cup \{w_1\ w_2\ \ldots\ w_m \ : \ m \in \mathbb{N}_+ \text{ and for all } i \in \{1, \ldots, m\},\ w_i \in R\}.$$

2.2.4 Tries and alike data structures

A trie is a fundamental data structure for storing and retrieving data. Let $\mathcal{A} = \{a_1, \ldots, a_N\}$ be an alphabet and let $S \subseteq \mathcal{A}^{\leq\infty}$ be a *non-empty set of n distinct strings* over $\mathcal{A}$ such that *no string in S is a prefix of another string in S*.

Definition A *N-ary retrieval tree* over S (for short an N-ary trie) is an arc-labeled rooted N-ary tree $T = (V, E, l)$ which can be defined via the following recursive procedure: if $\|S\| \leq 1$ then $T = (\{v\}, \{\}, \{\})$. If $\|S\| > 1$ then the set S is partitioned in r subsets $S_1, \ldots, S_r$ such that for $i \in \{1, \ldots, r\}$,

$$S_i = \{s \in S \ : \ a_i \sqsubseteq s\}$$

For each such set S_i which is non-empty, let S_i' be the set of all strings in S_i, where for each string the first symbol is deleted, i.e.,

$$S_i' = \{s[2\ldots] \ : \ s \in S_i\}.$$

Let $T_i = (V_i, E_i, l_i)$ be the trie over S_i' and let v_i be the root of T_i. Then $T = (V, E, l)$ is defined as

$$V = \bigcup_{\substack{i=1 \\ S_i \neq \emptyset}}^{N} V_i \cup \{v\}$$

and

$$E = \bigcup_{\substack{i=1 \\ S_i \neq \emptyset}}^{N} (E_i \cup \{(v_i, v)\})$$

with the labeling function $l : E \to \mathcal{A}$ that assigns each arc $(v, v_i) \in E$ the label 'a_i' and each arc $e \in \cup_{\substack{i=1 \\ S_i \neq \emptyset}}^{N} E_i$ the label $l_i(e)$. We may omit the term N-ary because it corresponds with the cardinality of the alphabet.

Retrieval Now, in order to retrieve a given string s, one follows the directed path of arcs labeled $s[1], s[2], s[3], \ldots$ starting from the root as long as either an external vertex is reached, or such a path can no longer be followed in T because an internal vertex has no child with an appropriately labeled arc. In the first case the respective string is found and in the latter case it is not present in T, at all.

Efficiency issues We are not interested in the details of constructing tries, but we are more concerned with the efficiency of the retrieval operation. It is easy to see a retrieval operation takes time $O(d)$, where d denotes the depth of the first external vertex on the retrieval path for a string. Thus the retrieval operation takes time $O(h)$, where h denotes the height of the trie. For the sake of completeness it should be mentioned that a trie can be constructed incrementally by inserting one string in S at a time.

Enhancements A PATRICIA tree is a trie, where paths of vertices of degree one are contracted into one arc and labeled by the string that results by concatenating the labels on the path in the direction starting with the label that is nearest to the root of the trie. PATRICA is the acronym for "Practical Algorithm to Retrieve Information Coded in Alphanumeric".

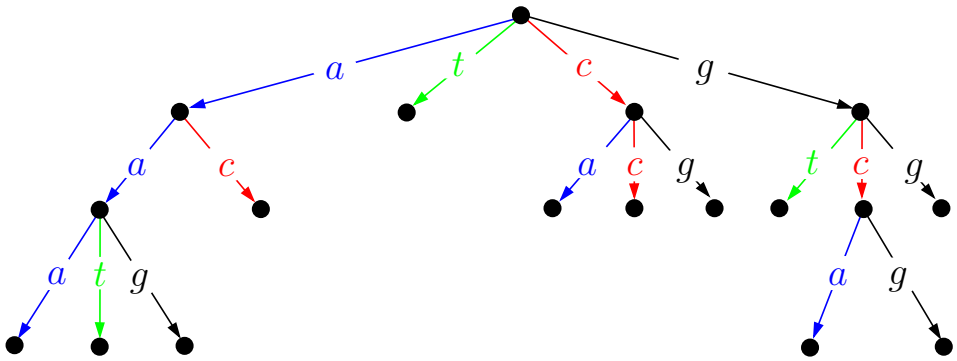


Figure 2.1: An example of a 4-ary trie built over the set $\{aaaa, aata, aagc, acct, atcg, cagt, cctg, cgaa, gtaa, gcaa, gcgg, ggac\}$. The symbols and their respective edges are drawn in different colors: a, t, c, g.

This enhancement was introduced by Morrison [Mor68]. *Level- and path-compressed (LCP) tries* are an enhancement of ordinary (binary) tries which were introduced by Andersson and Nilsson [AN93, AN94]: given a rooted binary tree T, we denote the set of vertices at depth i by L_i. The level- and path-compressed version of T is then constructed by the following recursive procedure: let i be maximum complete level in T, i.e., i maximum such that $\|L_i\| = 2^i$ and $\|L_{i+1}\| < 2^{i+1}$. The subtree of T induced by the vertices in $\cup_{j=0}^{i} L_j$ is then a complete binary tree. In the LCP trie corresponding to T, this subtree is replaced by a root having 2^i child vertices, where the arc between the root and the j-th child is labeled by the i-bit binary representation of the number j. This compression is then applied recursively to the subtrees rooted at the 2^i child vertices. Additionally to the arc-labelings, at each vertex a value *bit* is stored such that the degree of a vertex equals $2^{\,bit}$. In order to retrieve a string s that is stored in the LCP trie T, one again follows a labeled path in T starting from the root. Only this time, the labels are $s[1 \dots k_1], s[k_1 + 1 \dots k_2], \dots$, where $k_i - k_{i-1} + 1$ (with $k_0 = 1$) equals the size of the *bit*-value of the i-th vertex on this path. Finally, a *b-trie* over an alphabet $\mathcal{A}$ is a rooted $\|\mathcal{A}\|$-ary arc-labeled tree such that every leaf is capable of storing up to b strings. All strings that are stored in a leaf have the same common prefix, which equals the concatenation of the labelings on the path from the root to the respective leaf in the b-trie.

2.3 Generating functions of regular specifications

In our analyses, particularly in Chapter 4, we make use of the theory of generating functions in order to evaluate sums of probabilities. The theory is much too involved to give a complete introduction within the preliminaries of a PhD thesis. We nevertheless review those parts of the theory that are crucial for the understanding of the calculations performed in this thesis. For an introduction, the reader is referred to the book of Graham, Knuth and Patashnik [GKP05] or the book of Wilff [Wil94]. For a very thorough reading, the book of Flajolet and Sedgewick [FS07] is recommended. The expositions in this overview follow the expositions given in [FS07].

Given a finite or denumerable set of elements $\mathcal{A}$, called a *combinatorial class*, such that the size $|.| : \mathcal{A} \to \mathbb{N}_+$ is defined and such that the number of elements of a given size is finite, the sequence $(A_i)_{i \geq 0}$, where $A_i = |\{a \in \mathcal{A} \ : \ |a| = i\}|$, is called the *counting sequence* of

the class $\mathcal{A}$. The *ordinary generating function (OGF)* $A(z)$ corresponding to the sequence $(A_i)_{i\geq 1}$ is the following *formal power series.*

$$A(z) = \sum_{i=0}^{\infty} A_i \cdot z^i,$$

where we implicitly assume that z is such that the power series converges and we can consider them as algebraic objects (with some care!). Given a formal power series, we are interested in the coefficient at z^m because it equals the number A_m. To extract the coefficient, we use the operator

$$[z^m]\, A(z) = A_m.$$

Regular specifications A class of well-understood combinatorial objects are those sets for which there exists a *regular specification.* The simplest combinatorial classes that are given by a regular specification are the empty set $\mathcal{A} = \{\}$ and the singleton set $\mathcal{A}' = \{\alpha\}$. The corresponding generating functions are $A(z) = 1$ and $A'(z) = z$. Let $\mathcal{A}$ and $\mathcal{B}$ be a two combinatorial classes with corresponding OGFs $A(z)$ and $B(z)$.

- The *Cartesian product* is the combinatorial class

  $$\mathcal{A} \times \mathcal{B} =_{\text{def}} \{(\alpha, \beta) \ : \ \alpha \in \mathcal{A} \text{ and } \beta \in \mathcal{B}\},$$

  where the size $|(\alpha, \beta)| = |\alpha| + |\beta|$. The OGF corresponding to the combinatorial product is $A(z) \cdot B(z)$.

- Let $\heartsuit$ and $\spadesuit$ be two distinct markers. The *combinatorial sum* is defined as

  $$\mathcal{A} + \mathcal{B} =_{\text{def}} \{(\heartsuit, \alpha) \ : \ \alpha \in \mathcal{A}\} \cup \{(\spadesuit, \beta) \ : \ \beta \in \mathcal{B}\},$$

  where $|(\heartsuit, \alpha)| = |\alpha|$ and $|(\spadesuit, \beta)| = |\beta|$. The corresponding OGF is $A(z) + B(z)$.

- Finally, let $\mathcal{A}$ be non-empty. The *sequence construction* is the combinatorial class

  $$\mathcal{A}^* =_{\text{def}} \{\} + \mathcal{A} + (\mathcal{A} \times \mathcal{A}) + (\mathcal{A} \times \mathcal{A} \times \mathcal{A}) + \ldots,$$

  where $|(\alpha_1, \ldots, \alpha_k)| = |\alpha_1| + \ldots + |\alpha_k|$. The corresponding OGF is

  $$(1 - A(z))^{-1} = \sum_{i=0}^{\infty} A(z)^i \cdot z^i.$$

  .

Definition 2.7 (Regular specification)**.** *A specification for a combinatorial class that involves only singleton sets, Cartesian products, combinatorial sums and sequence constructions is called a* regular specification.

Note that for every combinatorial class with regular specification which involves the singleton sets $\{a_1\}, \ldots, \{a_n\}$, there is a corresponding regular language over the alphabet $\{a_1, \ldots, a_n\}$.

Fact 2.1. *Let $\mathcal{A}$ be a combinatorial class which has a regular specification. Then the corresponding OGF $A(z)$ is a* rational function.

Note that the construction of the rational OGF corresponding to a combinatorial class with regular specification is straightforward: each symbol of the corresponding regular language is translated into the variable z and the operations are then transformed as follows:

- $a_i\ a_j$ (which corresponds to Cartesian product in the regular specification) becomes $z \cdot z$,
- $a_i \mid a_j$ (which corresponds to combinatorial sum in the regular specification) becomes $z + z$ and
- a_i^* (which corresponds to the sequence construction in the regular specification) becomes $1/(1-z)$.

Sometimes, it is more convenient to use a *multivariate generating function* (MGF). Let $\mathcal{A}$ be a combinatorial class with regular specification which involves the singleton sets $\{a_1\}, \ldots, \{a_n\}$. The multivariate generating function corresponding to the class $\mathcal{A}$ is given by the multivariate function $f(\mathbf{z}, u_1, \ldots, u_n)$ which results from the following construction: consider the corresponding regular language over the alphabet $\{a_1, \ldots, a_n\}$.

- $a_i\ a_j$ becomes $z^2 \cdot u_i \cdot u_j$,
- $a_i \mid a_j$ becomes $u_i \cdot z + u_j \cdot z$ and
- a_i^* becomes $1/(1 - u_i \cdot z)$.

Here, the variable z is said to *mark size*. Assume that a weight $\lambda_i \in \mathbb{R}_+$ is attach to every atomic element a_i, $1 \leq i \leq n$, in the regular specification and let an element in the combinatorial class have the following weight:

- for an atomic element a_i, $1 \leq i \leq n$, in the regular specification, $w(a_i) =_{\text{def}} \lambda_i$,
- for an element which is formed by a Cartesian product, $w((\alpha, \beta)) = w(\alpha) \cdot w(\beta)$,
- for an element which is formed by a combinatorial sum, $w((\heartsuit, \alpha)) = w(\alpha)$ and
- for an element which is formed by a sequence construction, $w((\alpha_1, \ldots, \alpha_k)) = \prod_{i=1}^{k} w(\alpha_i)$.

The weight of a set equals the sum over all weights of its elements. Let $\mathcal{A}$ be a combinatorial class which has a regular specification involving n atomic elements and assume that a weight is attached to each element as described above. Let $A(\mathbf{z}, u_1, \ldots, u_n)$ be the corresponding MGF. The weight of the set $\mathcal{A}^{(n)} = \{x \in \mathcal{A} \ : \ |x| = n\}$,

$$w(\mathcal{A}^{(n)}) = \sum_{x \in \mathcal{A}^{(n)}} w(x)$$

can be derived from the coefficient at z^n of the MGF $A'(z) = A(z, \lambda_1, \ldots, \lambda_n)$. This is,

$$w(\mathcal{A}^{(n)}) = [z^n]\, A(z, \lambda_1, \ldots, \lambda_n).$$

Coefficient extraction In order to extract the coefficient at z^n of a rational function, we make use of the following result on the expansion of rational functions.

Theorem 2.1 (Expansion of rational functions)**.** [Theorem IV in [FS07]] *If $f(z)$ is a rational function that is analytic at zero and has poles at $z_1 \leq z_2 \leq \ldots \leq z_k$ then its coefficients are a sum of* exponential polynomials*: there exist k polynomials $\Pi_1(z), \ldots, \Pi_k(z)$ such that for m larger than some fixed m_0,*

$$[z^m]\, f(z) = \sum_{j=1}^{k} \Pi_j(m) \cdot \left(\frac{1}{z_j}\right)^m.$$

Furthermore, the polynomial Π_j has degree equal to the order of the pole at z_j minus one.

For the sake of completeness, we give a proof for the above theorem. The proof follows the proof in [FS07].

Proof. First note that since $f(z)$ is a rational functions it admits a partial fraction decomposition. For $i \in \{1, \ldots, k\}$ let $\mathrm{mult}(i)$ be the multiplicity of the pole of f at z_i. The partial fraction decomposition is

$$f(z) = Q(z) + \sum_{i=1}^{k} \sum_{j=0}^{\mathrm{mult}(i)} \frac{c_{i,j}}{(z - z_i)^j},$$

where $Q(z)$ is a polynomial of degree $n_0 := \mathrm{N} - \mathrm{D}$ if $f = N/D$ and the $c_{i,j}$'s are constants. Now, according to Newton's expansion, for $i \in \{1, \ldots, k\}$ and $j \in \{1, \ldots, \mathrm{mult}(i)\}$,

$$[z^m]\, \frac{1}{(z - z_i)^j} = \frac{(-1)^j}{(z_i)^j}\, [z^m]\, \frac{1}{(1 - \frac{z}{z_i})^j} = \frac{(-1)^j}{z_i^j} \binom{m + j - 1}{j - 1} \left(\frac{1}{z_i}\right)^m,$$

where the binomial coefficient clearly is a polynomial of degree $j - 1$ in m. Now, collecting the terms associated with a given z_i and summing over all $i \in \{1, \ldots, k\}$ yields the statement of the theorem. □

Chapter 3

Approximating graph metrics by tree metrics

A fool sees not the same tree that a wise man sees.

Willliam Blake

3.1 Introduction

3.1.1 Motivation and problem statement

In this chapter, we study how well the distance metrics of a given undirected simple graph can be approximated by the distance metrics of its spanning trees. Clearly, the distance metrics of a graph is expressed in its *distance matrix*. If not explicitly otherwise stated, we assume for a graph $G = (V, E)$ that its vertex set is $V = \{v_1, \ldots, v_n\}$ and its edge set has size $\|E\| = m$. Also, we remind the reader that $D_G = \big(d_G(v_i, v_j)\big)_{1 \leq i,j \leq n}$ is the distance matrix corresponding to G and that D_G is symmetric with all entries being non-negative. Furthermore, for every spanning tree $T \subseteq G$ and all $v_i, v_j \in V$ it holds that $D_T[i, j] \geq D_G[i, j]$. We study different versions of the following combinatorial optimization problem: given a real-valued similarity measure rating the degree of correct approximation of the distance matrix of a graph by the distance matrix of a spanning tree, find for an input graph G a spanning tree T which is optimal with respect to the given similarity measure. Particularly, we consider as similarity measure $\varrho(G, T)$ the following functions:

- the standard matrix norms $\|.\|_{L,p}$ (for $1 \leq p < \infty$) $\|.\|_{L,\infty}$, $\|.\|_1$ and $\|.\|_\infty$ applied to the distance matrix D_T of the T.
- the standard matrix norms $\|.\|_{L,p}$ (for $1 \leq p < \infty$) $\|.\|_{L,\infty}$, $\|.\|_1$ and $\|.\|_\infty$ applied to the difference $(D_T - D_G)$ of distance matrix of T and G.
- the standard vector norms $\|.\|_p$ applied to the difference $(c_G - c_T)$ of the closeness centrality vector of G and T,

where the *closeness centrality vector* c_G of a graph G is defined as $c_G = (c_G(v_1), \ldots, c_G(v_n))^T$ and for $i \in \{1, \ldots, n\}$ the *closeness centrality* [Sab66, Bea65] of the vertex v_i is defined as

$$c_G(v_i) =_{\text{def}} \Big(\sum_{j=1}^{n} d_G(v_i, v_j)\Big)^{-1}. \tag{3.1}$$

Note that for any graph G and any subgraph $G' \subseteq G$ we have $c_G(v) \geq c_{G'}(v)$ for all vertices $v \in V$. We systematically study the impact of the matrix norm on the complexity of the following optimization problems: first, we consider *spanning trees that minimize distances.*

Problem:	Distance Minimizing Spanning Tree (DMST) with respect to $\|.\|_r$.
Input:	A connected graph G.
Task:	Find a spanning tree T of G such that $\|D_T\|_r$ is minimal.

In order to make clear the connection to the similarity measure which we introduced, note that in the above case the similarity measure $\varrho_r(G,T) = \|D_T\|_r$. Some variants of this problems have already been considered in the literature: for the L_1 norm the tree achieving the minimum is known as the minimum average distance tree (or, MAD-tree for short) [JLRK78, DDGS03]; for the L_∞ norm the tree realizing the minimum is known as the minimum diameter spanning tree [CGM80, HT95]. Also, these problems correspond to the MRCT problem, i.e., the special version of the OCT problem where $r(u,v) = 1$ for all pairs of vertices, which we mentioned in the introduction. Second, we consider *spanning trees that approximate distances.*

Problem:	Distance Approximating Spanning Tree (DAST) with respect to $\|.\|_r$.
Input:	A connected graph G.
Task:	Find a spanning tree T of G such that $\|D_T - D_G\|_r$ is minimal.

Here, the similarity measure $\varrho(G,T) = \|D_T - D_G\|_r$. It is mainly this version of the problem from which the term 'similarity measure' originally emerged: instead of merely optimizing the distance matrix of the tree as such, we here aim at finding a tree that approximates the distance matrix of a given graph. This constitutes the key ingredient of what we refer to as *combinatorial network abstraction* in the introduction. With respect to the L_1 norm, the DAST and the DMST problem are similar in the sense that there is an affine transformation between the measures of the different problems. Thus, DAST with respect to the L_1 norm again reduces to looking for MAD-trees. By contrast, for the L_∞ norm the two variants differ tremendously and here we are this time looking for a tree that, for all vertex pairs, does not exceed a certain amount of additive increase in distance. Such trees are know as additive tree spanners [KLM$^+$03]. As it was already mentioned, we do not only deal with matrix-norm-based similarity measures, but we also consider here a slight variation of the DAST problem. Namely, we consider the problem of *finding a spanning tree that approximates centralities.*

Problem:	Centrality Approximating Spanning Tree (CAST) with respect to $\|.\|_r$.
Input:	A connected graph G.
Task:	Find a spanning tree T of G such that $\|c_G - c_T\|_r$ is minimal.

Note that except for the L_1 matrix-norm, distance-minimizing spanning trees and distance-approximating spanning trees typically cannot be used to provide good approximate solutions for each other: the different underlying matrices drastically affect the cost of an optimal solution as well as the structure of the corresponding spanning tree. An example for this (with respect to the L_∞ norm) is given in Figure 3.2. Whilst the upper spanning tree T provides a minimum diameter spanning tree for G, it approximates an optimal distance-approximating spanning tree only by a factor of $\Theta(\ell) = \Theta(\|V\|)$. The lower spanning tree T' is an optimal distance-approximating spanning tree for G – whilst being suboptimal with

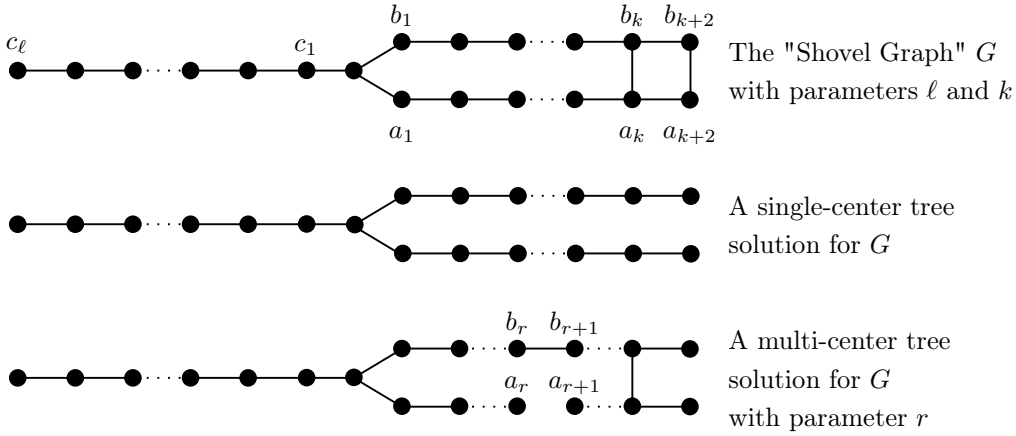


Figure 3.1: An illustration for the difference between distance minimization and distance approximation for L_p matrix-norms.

respect to minimizing $\|D_T\|_{L,\infty}$. Note that there is no spanning tree for G which provides an optimal solution to both problems. Figure 3.1 gives another example that separates optimal solutions for distance approximation and distance minimization for L_p. For arbitrary k and $\ell = 2^p(2k+4)+1$, the distance-minimizing spanning tree is the single center tree, which is not optimal with respect to distance approximation. Conversely, the best possible distance-approximating spanning tree is a multi-center tree for some r with $0 \leq r \leq k$ which, however, is not optimal for distance minimization. When increasing ℓ to $2(2k+3)^p+1$, the single center solution is optimal for both distance approximation and distance minimization. This shows that distance approximation, to a certain degree, prefers locally good solutions over globally good solutions. When p increases, the "shaft" of the "shovel" can be made even longer compared to the "blade". Hence, a large p amplifies the local influence. Throughout this chapter, the graphs which we consider are simple, unweighted, and undirected.

3.1.2 Our contribution

In order to study the impact of the norm on the complexity of the described combinatorial optimization problems, we consider the decision versions of the above optimization problems. For computing distance-minimizing spanning trees, it is known that it is NP-complete to decide on input (G,γ) whether there is a spanning tree T of G such that $\|D_T\|_{L,1} \leq \gamma$ [JLRK78]. Moreover, there exists a polynomial-time algorithm for computing a minimum diameter spanning tree [HSP78, KH79, CGM80, HT95]. However, for computing distance-approximating spanning trees in general graphs, even for L_1 and L_∞, no such complexity results are known to the best of our knowledge.[1] Research in this area has more focused on proving the (non-)existence of certain distance approximating trees for special graph classes (see, e.g., [Pri97, BCD99, FK01b, KLM+03]). These existence theorems are usually complemented with

[1]Note that in contrast to some claim in the literature the results in [LS93] do *not* provide a proof for the NP-completeness of deciding whether there is a spanning tree T with $\|D_T - D_G\|_{L,\infty}(G) \leq \gamma$, neither does an easily conceivable adaption.

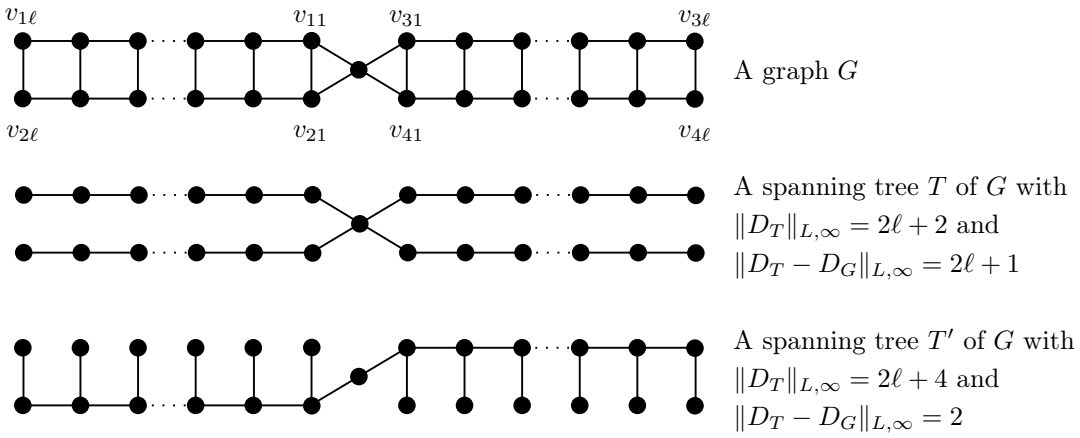


Figure 3.2: An illustration for the difference between distance minimization and distance approximation for the L_∞ matrix-norm.

polynomial-time algorithms for finding the guaranteed trees. Unfortunately, we cannot break the curse of NP-completeness for our optimization problems. We contribute the following:

- We prove that for any given instance (G, γ), deciding whether there is a spanning tree T such that $\|D_T\|_r \leq \gamma$ is NP-complete for all the matrix norms within our framework where complexity has been unknown so far.

- We prove that for any given instance (G, γ), deciding whether there is a spanning tree T of G such that $\|D_T - D_G\| \leq \gamma$ is NP-complete for all matrix norms within our framework, i.e., essentially for all standard norms with the exceptional case of the spectral norm which is left open. This is somewhat surprising, since at least in the case of L_∞, one might have hoped for a polynomial-time algorithm building up on the polynomial-time algorithms for computing minimum diameter spanning trees. Even worse, the polynomial-time algorithm for computing the minimum diameter spanning tree cannot be used for approximating $\min_T \|D_T - D_G\|_{L,\infty}$ within reasonable factors as already shown by the example in Figure 3.2.

- We prove that, with respect to closeness centrality, for any given instance (G, γ), deciding whether there is a spanning tree T such that $\|c_G - c_T\|_r \leq \gamma$ is NP-complete for the L_1 vector-norm.

- We present an $O(n^2 \log n + nm)$ time 2-approximation algorithm for the problem DMST with respect to the $\|.\|_{L,p}$ norm for $1 \leq p < \infty$.

- We show how the 2-approximation algorithm translates into a 2-approximation algorithm for a version of the multiple sequence alignment problem under a generalized sum-of-pairs objective.

3.1.3 Chapter outline

The detailed presentation of the results in this chapter is organized as follows: in Section 3.2.1, we introduce the gadgets, namely the X3C gadget and the 2HS gadget, which we use in the reductions to follow. In Section 3.2.2, we consider the DMST versions of our problems: the NP-completeness results for these versions are without exception derived by reductions using modifications of the X3C gadget. In Section 3.2.3, we consider the DAST versions of our problems: the problem DAST with respect to the $\|.\|_{L,\infty}$ matrix norm is shown to be NP-complete using a reduction which involves the 2HS gadget and a problem specific construction which we call *cycle assembly*. The hardness of the other DAST problems is again shown by reductions using "different versions" of the X3C gadget. In Section 3.2.4, we prove the hardness of the CAST problem with respect to the $\|.\|_1$ matrix norm by modifying the X3C gadget once more. Section 3.3 contains the presentation of the approximation algorithm and Section 3.4 deals with its application to the MSA problem. In Section 3.5, we give some bibliographic notes.

3.2 The results in detail

For all of the problems which we consider in this section, we will establish NP-completeness results. In order to establish these results, we need the decision versions of the above optimization problems. We will give the formal definitions of these problems at the beginning of the respective Sections 3.2.2, 3.2.3 and 3.2.4. We start the detailed exposition of our results by first defining the gadgets which are necessary in the reductions.

3.2.1 Gadgets

We will prove the NP-completeness of the considered spanning tree problems by reduction from two problems, i.e., X3C and 2HS, whose NP-completeness is well known. To this aim, we show that we can construct for each instance of the respective problems a graph such that there is a spanning tree for the graph having a prescribed similarity to the graph, if and only if the respective instance of the problem from which we reduce has a solution of a certain quality. Since a tree is a cycle-free graph, this means that in order to find a spanning tree, one needs to delete a certain amount of edges from the graph. Every such deletion results in an increase of the distance between some pairs of vertices (at least the distance between to two endpoints of the deleted edge increases). By carefully constructing the respective graphs, one can control these increases, i.e., one can assure that for some edges the penalty is larger and for some edges it is less than for others, and thus assure that only those spanning trees have small penalties which correspond to solutions of the corresponding problem.

3.2.1.1 Graph representation of X3C instances

The following decision problem is well-know to be NP-complete [GJ79].

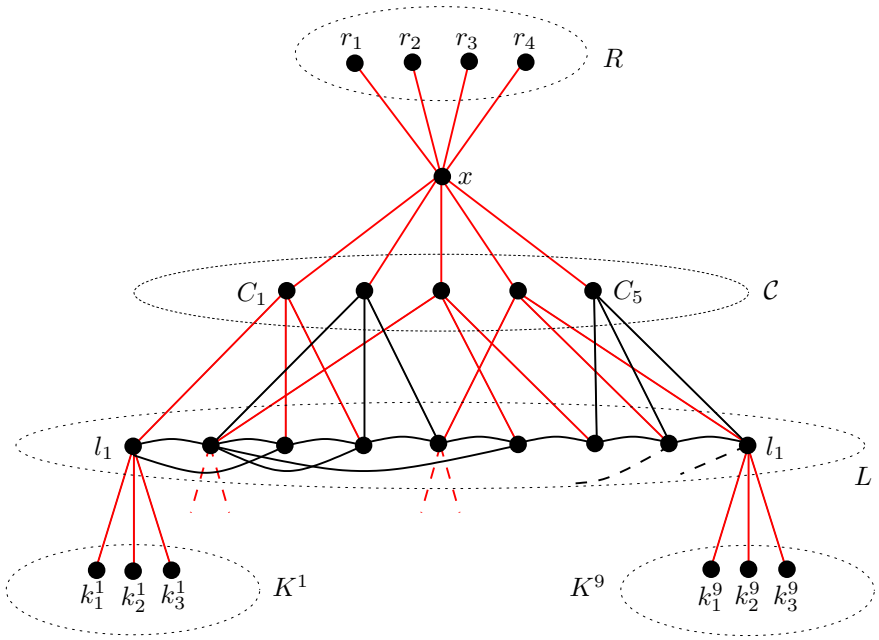


Figure 3.3: The graph $G_{4,3}(\mathcal{C}, L)$ for $\mathcal{C} = \{\{l_1, l_3, l_4\}, \{l_2, l_4, l_5\}, \{l_2, l_6, l_7\}, \{l_5, l_8, l_9\}, \{l_7, l_8, l_9\}\}$ and $L = \{l_1, l_2, l_3, l_4, l_5, l_6, l_7, l_8, l_9\}$. The edges of the solution tree $T_{\mathcal{V}}$ of $G_{4,3}(\mathcal{C}, L)$ corresponding to the solution $\mathcal{V} = \{C_1, C_3, C_5\}$ are drawn red.

Problem:	Exact-3-Cover (X3C).
Input:	A family $\mathcal{C} = \{C_1, \ldots, C_s\}$ of 3-element subsets of a set $L = \{l_1, \ldots, l_{3m}\}$.
Question:	Is there a subfamily $\mathcal{V} \subseteq \mathcal{C}$ of pairwise disjoint sets such that $\cup_{A \in \mathcal{V}} = L$?

A subfamily $\mathcal{V}$ satisfying the required properties is called an *admissible solution* to an instance $(\mathcal{C}, L)$. Let a and b be arbitrary natural numbers. Extending a construction from [JLRK78], we define the graph $G_{a,b}(\mathcal{C}, L)$ to consist of the vertex set

$$V =_{\text{def}} \mathcal{C} \cup L \cup R \cup X \cup K,$$

where $R =_{\text{def}} \{r_1, \ldots, r_a\}$, $X = \{x\}$, $K =_{\text{def}} \bigcup_{i=1}^{3m} K^i$ and for all $i \in \{1, \ldots, 3m\}$, $K^i =_{\text{def}} \{k_1^i, \ldots, k_b^i\}$. The edge set of $G_{a,b}(\mathcal{C}, L)$ is

$$\begin{aligned} E \ =_{\text{def}} \ & \{\ \{r_\mu, x\} \ : \ \mu \in \{1, \ldots, a\}\ \} \cup \{\ \{C_\mu, x\} \ : \ \mu \in \{1, \ldots, s\}\ \} \cup \\ & \cup \{\ \{l_\mu, C_\nu\} \ : \ l_\mu \in C_\nu\ \} \cup \{\ \{l_\mu, l_\nu\} \ : \ \mu, \nu \in \{1, \ldots, 3m\}\ \} \cup \\ & \cup \{\ \{k_\nu^\mu, l_\mu\} \ : \ \mu \in \{1, \ldots, 3m\} \text{ and } \nu \in \{1, \ldots, b\}\ \}. \end{aligned}$$

This construction is illustrated in Figure 3.3. The following two propositions capture some important relations between the graph $G_{a,b}(\mathcal{C}, L)$, a corresponding instance $(\mathcal{C}, L)$, an admissible solution $\mathcal{V}$ to $(\mathcal{C}, L)$ and the spanning trees of $G_{a,b}(\mathcal{C}, L)$.

Proposition 3.1. *Let $(\mathcal{C}, L)$ be an* X3C *instance and let a, b be natural numbers. Suppose T is a spanning tree of the graph $G_{a,b}(\mathcal{C}, L)$.*

1. *For all $\mu \in \{1, \ldots, a\}$, T contains the edge $\{r_\mu, x\}$.*
2. *For all $\mu \in \{1, \ldots, 3m\}$ and $\nu \in \{1, \ldots, b\}$, T contains the edge $\{k_\nu^\mu, l_\mu\}$.*
3. *If for some $\mu \in \{1, \ldots, s\}$, T does not contain the edge $\{C_\mu, x\}$, then we have $d_T(C_\mu, r_\nu) \geq 4$ and $d_T(C_\mu, r_\nu) \geq d_{G_{a,b}(\mathcal{C},L)}(C_\mu, r_\nu) + 2$ for all $\nu \in \{1, \ldots, a\}$.*
4. *If for some $\mu \in \{1, \ldots, 3m\}$, the vertex l_μ is not adjacent in T to any $C_\nu \in \mathcal{C}$, $\nu \in \{1, \ldots, s\}$, then for all $\kappa \in \{1, \ldots, a\}$ and $\lambda \in \{1, \ldots, b\}$ we have $d_T(l_\mu, r_\kappa) \geq 4$, $d_T(k_\lambda^\mu, r_\kappa) \geq 5$, $d_T(l_\mu, r_\kappa) \geq d_{G_{a,b}(\mathcal{C},L)}(l_\mu, r_\kappa) + 1$, and $d_T(k_\lambda^\mu, r_\kappa) \geq d_{G_{a,b}(\mathcal{C},L)}(k_\lambda^\mu, r_\kappa) + 1$.*

Given an admissible solution $\mathcal{S}$ to an X3C instance $(\mathcal{C}, L)$, we can identify a corresponding spanning subgraph $T_\mathcal{S}$ called *solution tree* in $G_{a,b}(\mathcal{C}, L)$ through the edge set

$$\begin{aligned} E(T_\mathcal{S}) \;=\; & \big\{\, \{r_\mu, x\} \;:\; \mu \in \{1, \ldots, a\} \,\big\} \cup \big\{\, \{C_\mu, x\} \;:\; \mu \in \{1, \ldots, s\} \,\big\} \cup \\ & \cup \big\{\, \{l_\mu, C_\nu\} \;:\; l_\mu \in C_\nu \text{ and } C_\nu \in \mathcal{S} \,\big\} \cup \\ & \cup \big\{\, \{k_\nu^\mu, l_\mu\} \;:\; \mu \in \{1, \ldots, 3m\} \text{ and } \nu \in \{1, \ldots, b\} \,\big\}. \end{aligned}$$

The edges of the solution tree corresponding to the unique solution of the graph $G_{4,3}(\mathcal{C}, L)$ are drawn as red lined in Figure 3.3.

Proposition 3.2. *Let $(\mathcal{C}, L)$ be an* X3C *instance having an admissible solution $\mathcal{S} \subseteq \mathcal{C}$, let a and b be natural numbers and let $T_\mathcal{S}$ be a solution tree in $G_{a,b}(\mathcal{C}, L)$ that corresponds to $\mathcal{S}$.*

1. *For all $\mu \in \{1, \ldots, s\}$: If $C_\mu \in \mathcal{S}$, then C_μ has four neighbors in $T_\mathcal{S}$, otherwise C_μ has only one neighbor in $T_\mathcal{S}$.*
2. *For all vertices $u \in R \cup X$ and $v \in V$, $d_{T_\mathcal{S}}(u, v) = d_{G_{a,b}(\mathcal{C},L)}(u, v)$.*
3. *For all $\mu, \nu \in \{1, \ldots, s\}$, $d_{T_\mathcal{S}}(C_\mu, C_\nu) = d_{G_{a,b}(\mathcal{C},L)}(C_\mu, C_\nu)$.*

The following lemma gives necessary and sufficient conditions for a spanning tree to be a solution tree. In the reductions to follow, we frequently make use of the characterizations given in this lemma. Moreover, all the NP-completeness proof that rely upon this gadget, follow the same vein and thus it is natural to capture the essential ingredients of these proofs into a Lemma, which we therewith do.

Lemma 3.1. *Let $(\mathcal{C}, L)$ be an* X3C *instance, $a, b \in \mathbb{N}$, and let T be any spanning tree of the graph $G_{a,b}(\mathcal{C}, L)$. There exists an admissible solution $\mathcal{S} \subseteq \mathcal{C}$ such that $T = T_\mathcal{S}$ if and only if the following conditions are satisfied:*

1. *For all $\mu \in \{1, \ldots, s\}$, the tree T contains the edge $\{C_\mu, x\}$.*
2. *For all $\mu \in \{1, \ldots, 3m\}$, there is a $\nu \in \{1, \ldots, s\}$ such that T contains the edge $\{l_\mu, C_\nu\}$.*
3. *For all $\mu \in \{1, \ldots, s\}$, the vertex C_μ has either four neighbors in T or one.*

Proof. Clearly, the three conditions are necessary for a tree $T_\mathcal{S}$ to correspond to an admissible solution $\mathcal{S}$. For the other direction, suppose the tree T satisfies all conditions. By the first

and second conditions, for $\mu, \nu \in \{1, \ldots, 3m\}$ (with $\mu \neq \nu$), there exist $\kappa, \lambda \in \{1, \ldots, s\}$ such that the path $(l_\mu, \mathcal{C}_\kappa, x, \mathcal{C}_\lambda, l_\nu)$ exists in T. Thus, the edges $\{l_\mu, l_\nu\}$ do not belong to T. Consequently, using the third condition, we obtain an admissible solution by defining $\mathcal{S}$ to consist of all $\mathcal{C}_\mu$ having exactly four neighbors in T. □

Before we expose our second gadget, the meaning of the two parameters a and b should be made (at least a little bit) clear. Loosely speaking, a and b have some *amplifying effects*: to this aim, let $G_{a,b}(\mathcal{C}, L)$ be a graph and assume for the moment that the norm considered is the L_p norm and the considered measure is $\varrho_r(T, G) = \|D_T\|_{L,p}$. Let T be some spanning tree of $G_{a,b}(\mathcal{C}, L)$ and assume that $d_T(l_\nu, x) = 3$ for some vertex $l_\nu \in L$. We are interested in how much the value of $\|D_T\|_{L,p}$ increases due to the fact that the distance between l_ν and x is equal to 3 in T (instead of being equal to 2 as in $G_{a,b}(\mathcal{C}, L)$). For $a = 0$, the increase (with respect to the value $\|D_{G_{a,b}(\mathcal{C},L)}\|_{L,p}$) is clearly $2(3^p + b \cdot 4^p) - 2(2^p + b \cdot 3^p)$ and for $a > 0$ the increase is $a \cdot 2(3^p + a \cdot 4^p + b(4^p + a \cdot 5^p)) - 2(2^p + a \cdot 3^p + b(3^p + a \cdot 4^p))$. Now, by choosing the appropriate parameter values, we can control the penalty a tree suffers if it does not satisfy some sought-after properties. This idea will be made more explicit within the detailed proofs where we specify the exact values for a and b.

3.2.1.2 Graph representation of 2HS

The problem 2-Hitting Set(2HS) is known to be NP-complete [GJ79]. Also, it is commonly better known as Vertex Cover. In order to avoid overusing the terms "vertices" and "edges", we have decided to use its less frequently used name.

Problem:	2-Hitting Set (2HS).
Input:	A family $\mathcal{E} = \{E_1, \ldots, E_m\}$ of 2-element subsets of a set $\mathcal{V} = \{v_1, \ldots, v_n\}$ and a number $k \in \{1, \ldots, n\}$.
Question:	Is there a subfamily $\mathcal{V}' \subseteq \mathcal{V}$ such that $\|\mathcal{V}'\| \leq k$ and for each $\mu \in \{1, \ldots, m\}$ the set $E_\mu \cap \mathcal{V}' \neq \emptyset$?

A subset $\mathcal{V}' \subseteq \mathcal{V}$ having the required properties is called an admissible solution to a 2HS instance $(\mathcal{E}, \mathcal{V}, k)$. For a given 2HS instance $(\mathcal{E}, \mathcal{V}, k)$, we define the graph $G(\mathcal{E}, \mathcal{V})$ as follows:

- For each $s_\mu \in \mathcal{V}$, we introduce a *literal gadget* G_μ consisting of the two *connection vertices* s_μ, t_μ and the vertices $e^\mu_1, \ldots, e^\mu_{m+1}$ and $l^\mu_1, \ldots, l^\mu_m$. Moreover, s_μ is connected to t_μ via two paths: the first path $(s_\mu, e^\mu_1, \ldots, e^\mu_{m+1}, t_\mu)$ is called *elongation path* and has length equal to $m + 2$, whereas the second path $(s_\mu, l^\mu_1, \ldots, l^\mu_m, t_\mu)$ of length $m + 1$ is called the *literal path*.

- We introduce the three vertices a', a'', b'.

- For each $i \in \{1, \ldots, n-1\}$, we connect the successive connection vertices t_i and s_{i+1} via an edge. Also, we introduce the edges $\{a', a''\}, \{a'', s_1\}$ and $\{t_n, b'\}$.

- For each $E_\mu = \{v_\nu, v_\kappa\}$, we introduce a *clause path* of length $2n(m + 2)$ that connects the vertices l^ν_μ and l^κ_μ and a *safety path* of length $2n(m + 2)$ that connects the vertices a' and l^ν_μ (Here, we assume w.l.o.g. that $\nu < \kappa$).

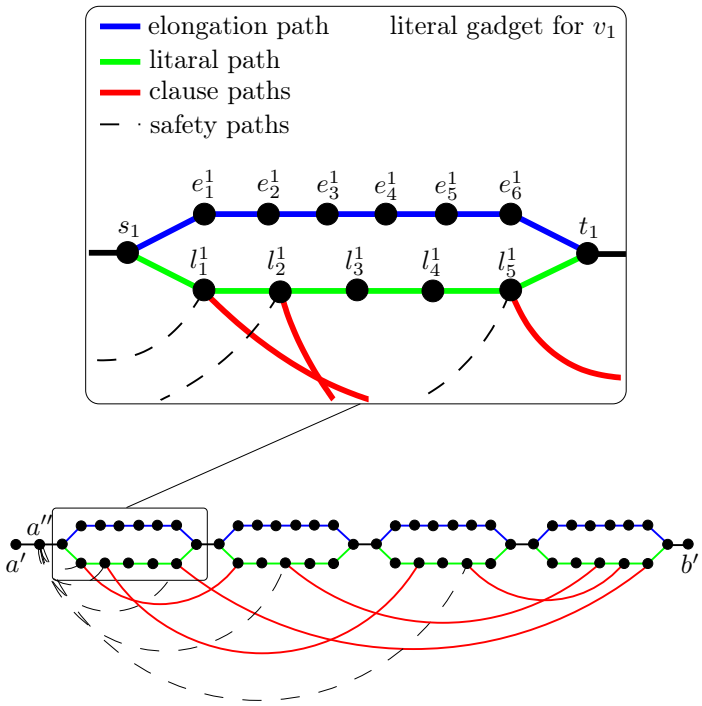


Figure 3.4: Graph representation $G(\mathcal{E}, \mathcal{V})$ of the 2HS instance $(\mathcal{E}, \mathcal{V}, 2) = (\{\{v_1, v_2\}, \{v_1, v_3\}, \{v_2, v_4\}, \{v_3, v_4\}, \{v_1, v_4\}\}, \{v_1, v_2, v_3, v_4\}, 2)$

The whole construction is exemplified in Figure 3.4. The main idea behind the gadget is that for each literal gadget, its elongation path is tuned such that it is exactly one edge longer than its literal path, and further that the path

$$(a', a'', s_1, l_1^1, \ldots, l_m^1, t_1, s_2, \ldots, s_n, l_n^1, \ldots, l_m^n, t_n, b')$$

is a shortest path between a' and b' in $G(\mathcal{E}, \mathcal{V})$ and all detours leading along clause paths or safety paths are very long. In any spanning tree the cycles induced by the clause paths can only be broken by destroying a certain amount of literal paths – ignoring for a moment the possibility of destroying clause paths. Thus, the distance between the two vertices a' and b' increases by one for every such broken literal path in the tree. This is captured in the following lemma.

Lemma 3.2. *Let $(\mathcal{E}, \mathcal{V}, k)$ be an instance of* 2HS. *Then*

$$d_{G(\mathcal{E},\mathcal{V})}(a', b') = n \cdot (m + 2) + 2.$$

Further, there exists an admissible solution $\mathcal{V}' \subseteq \mathcal{V}$ to $(\mathcal{E}, \mathcal{V}, k)$ if and only if there exists a spanning tree T of $G(\mathcal{E}, \mathcal{V})$ containing all edges in the clause paths such that $d_T(a', b') - d_{G(\mathcal{E},\mathcal{V})}(a', b') \leq k$.

The proof is given in the appendix. In our NP-completeness proofs, we use a slightly modified gadget, where we *twist* the gadget into a cycle. This enables us to show not only

the hardness of the decision version of the problem DAST with respect to the $\|.\|_{L,\infty}$ matrix norm, but also gives at hand an alternative proof for the hardness of a restricted version of the maximum-stretch spanning tree problem considered in [Cai94, CC95].

3.2.2 Distance-minimizing spanning trees

In this section, we consider the problem of computing spanning trees of given graphs that minimize distances among the vertices of the graph under certain matrix norms. It turns out that all versions of the problem considered here are NP-complete. In order to perform the reductions necessary of these proofs, we define the decision versions of the problem.

Problem:	DMST (with respect to $\|.\|_r$).
Input:	A connected graph G and an algebraic number γ.
Question:	Is there a spanning tree T of G with $\|D_T\|_r \leq \gamma$?

We begin with our study by proving that computing distance-minimizing spanning trees is computationally hard under the L_p matrix-norm for all $1 \leq p < \infty$. Note that the case $p = 1$ corresponds to the MAD-tree problem which was shown to be NP-complete in [JLRK78].

Theorem 3.1. DMST *with respect to* $\|\cdot\|_{L,p}$ *is* NP*-complete for all* $p \in \mathbb{N}_+$.

Proof. First note that the containment in NP is obvious. We prove the hardness by reduction from X3C using the graph representation $G_{a,0}(\mathcal{C}, L)$ for any instance $(\mathcal{C}, L)$. We fix our parameters in an appropriate manner in a moment, so that $(\mathcal{C}, L)$ has an admissible solution $\mathcal{S} \subseteq \mathcal{C}$ if and only if $G_{a,0}(\mathcal{C}, L)$ has a spanning tree T such that $\|D_T\|_{L,p}^p \leq \gamma^p$. Define

$$N =_{\text{def}} \sum_{\substack{u,v \in V \text{ and} \\ u \in R \cup X \text{ or } v \in R \cup X}} d_{G_{a,0}(\mathcal{C},L)}(u,v)^p.$$

Suppose $\mathcal{S} \subseteq \mathcal{C}$ is an admissible solution to instance $(\mathcal{C}, L)$. Let $T_{\mathcal{S}}$ be the corresponding spanning tree of $G_{a,0}(\mathcal{C}, L)$. By Proposition 3.2, Property 2, N remains unchanged if the distance in $G_{a,0}(\mathcal{C}, L)$ are replaced by the distances in $T_{\mathcal{S}}$. Next, define

$$M =_{\text{def}} \sum_{u,v \in C \cup L} d_{T_{\mathcal{S}}}(u,v)^p.$$

Clearly, $\|D_{T_{\mathcal{S}}}\|_{L,p} = \big(N+M\big)^{1/p}$. The definition of the parameter M as a function of $T_{\mathcal{S}}$ seems contradictory at first sight, because the reduction should *only* depend on the instance $(\mathcal{C}, L)$ and *not* on a specific solution $\mathcal{S}$. Nevertheless, there is no contradiction in this particular case: the value M is fully specified only through the size of the instance $(\mathcal{C}, L)$, i.e., its *independent* of a specific solution $\mathcal{S}$, and can be calculated in time polynomial in the size of the instance. Now we are ready to set the necessary parameters as follows:

$$\begin{aligned} a &\stackrel{\text{def}}{=} \left\lceil \frac{M}{4^p - 3^p} \right\rceil \\ \gamma &\stackrel{\text{def}}{=} (N+M)^{1/p} \end{aligned}$$

Note that the parameters are indeed well-defined, because a only depends on the instance $(\mathcal{C}, L)$ and the parameter p of the matrix norm, and once a is fixed, γ is well-defined, too. It is easy to check that $\|D_{T_\mathcal{S}}\|_{L,p}^p = N + M = \gamma^p$. Thus, $T_\mathcal{S}$ is a spanning tree of $G_{a,0}(\mathcal{C}, L)$ having the desired distance property.

Now, suppose T is a spanning tree of $G_{a,0}(\mathcal{C}, L)$ such that $\|D_T\|_{L,p}^p \leq \gamma^p$. We apply the characterization of a solution tree given in Lemma 3.1 and show that all conditions are satisfied and thus T is indeed a solution tree. Note that, by Proposition 3.2, N is a lower bound for the p-distance sum between vertices in $R \cup X$.

- Assume to the contrary that the first condition of Lemma 3.1 does not hold, i.e., for some $\mu \in \{1, \ldots, s\}$, the edge $\{C_\mu, x\}$ is not in T. Then, $d_T(C_\mu, x) \geq 3$ and for all $\nu \in \{1, \ldots, a\}$, $d_T(C_\mu, r_\nu) \geq 4$. Thus, $\|D_T\|_{L,p}^p \geq N - 1^p - 2^p a + 3^p + 4^p a$ and we conclude

$$\|D_T\|_{L,p}^p - \gamma^p \geq 3^p - 1 + a(4^p - 2^p) - M > 1 + M\frac{(4^p - 2^p)}{4^p - 3^p} - M > 1,$$

  a contradiction. We have just seen how the parameter a amplifies the effect of a small increase in the distance between C_μ and x.

- Assume to the contrary that the second condition of Lemma 3.1 does not hold, i.e., there is a vertex l_μ not adjacent to any $\mathcal{C}_\nu$ in T. Then, $\|D_T\|_{L,p}^p \geq N - 2^p - 3^p a + 3^p + 4^p a$ and we conclude

$$\|D_T\|_{L,p}^p - \gamma^p \geq 3^p - 2^p + a(4^p - 3^p) - M > 1 + M\frac{(4^p - 3^p)}{4^p - 3^p} - M = 1,$$

  a contradiction.

- Note that, if the first and second condition in Lemma 3.1 are both satisfied, then all edges but those between $\mathcal{C}$ and L are already fixed by now and the distances in T and $G_{a,0}(\mathcal{C}, L)$ are the same except for those between vertices in L (between L and $\mathcal{C}$, each l_μ has p-distance one to exactly one C_ν and 3^p, otherwise). Let g be the number of pairs (l_μ, l_ν) such that edges $\{l_\mu, C_\kappa\}$ and $\{l_\nu, C_\kappa\}$ exist in T. The total number of pairs is $9m^2 - 3m$. We obtain

$$\sum_{\mu=1}^{3m}\sum_{\nu=1}^{3m} d_T(l_\mu, l_\nu)^p = 2^p g + 4^p(9m^2 - 3m - g).$$

  The maximum possible value of g is $6m$ which corresponds to the case that the third condition in Lemma 3.1 is satisfied. Assume to the contrary $g < 6m$. Then we have

$$\|D_T\|_{L,p}^p - \gamma^p \geq -2^p 6m - 4^p(9m^2 - 9m) + 2^p g + 4^p(9m^2 - 3m - g) = \\ = (6m - g)(4^p - 2^p) > 1,$$

  a contradiction.

This proves the theorem by applying Lemma 3.1. □

Next we state that distance-minimizing spanning trees are hard to find under the maximum column-sum norm and thus, maximum row-sum norm as well.

Theorem 3.2. DMST *with respect to* $\|\cdot\|_1$ *is* NP*-complete.*

Proof. Containment in NP is obvious. Again, NP-hardness is proved by reduction from X3C using the graph representation $G_{a,0}(\mathcal{C}, L)$ for a given X3C instance $(\mathcal{C}, L)$ and an appropriate choice of the parameters a and γ. We will fix the parameter later, so that $(\mathcal{C}, L)$ has an admissible solution $\mathcal{S} \subseteq \mathcal{C}$ if and only if $G_{a,0}(\mathcal{C}, L)$ has a spanning tree T such that $\|D_T\|_1 \leq \gamma$.

Suppose $\mathcal{S} \subseteq \mathcal{C}$ is an admissible solution to $(\mathcal{C}, L)$. Let $T_{\mathcal{S}}$ be the corresponding spanning tree in the graph $G_{a,0}(\mathcal{C}, L)$. The vertices in the sets R, X, and L all have the same column sums. We calculate for $\mu \in \{1, \ldots, a\}$ and $\nu \in \{1, \ldots, s\}$:

$$\begin{aligned} \sum_{v \in V} d_{T_{\mathcal{S}}}(r_\mu, v) &= 2a + 2s + 9m - 1 \\ \sum_{v \in V} d_{T_{\mathcal{S}}}(x, v) &= a + s + 6m \\ \sum_{v \in V} d_{T_{\mathcal{S}}}(l_\nu, v) &= 3a + 3s + 12m - 8 \end{aligned}$$

For $\mathcal{C}$, we have to make a distinction between vertices with one neighbor in $T_{\mathcal{S}}$ or four:

$$\sum_{v \in V} d_{T_{\mathcal{S}}}(C_\mu, v) = \begin{cases} 2a + 2s + 9m - 1 & \text{if } C_\mu \text{ has one neighbor in } T_{\mathcal{S}} \\ 2a + 2s + 9m - 7 & \text{if } C_\mu \text{ has four neighbors in } T_{\mathcal{S}} \end{cases}$$

We define our parameters as follows:

$$\begin{aligned} a &=_{\text{def}} s + 12m + 8 \\ \gamma &=_{\text{def}} 6s + 48m + 16 \end{aligned}$$

Clearly, we have $\|D_{T_{\mathcal{S}}}\|_1 = 6s + 48m + 16 = \gamma$. Thus, $T_{\mathcal{S}}$ is a spanning tree in $G_{a,0}(\mathcal{C}, L)$ having a distance property as desired.

Suppose T is a spanning tree in $G_{a,0}(\mathcal{C}, L)$ satisfying $\|D_T\|_1 \leq \gamma$. We apply the characterization of a solution tree given in Lemma 3.1 and show that all conditions are satisfied.

- Assume to the contrary that the first condition in Lemma 3.1 does not hold, i.e., for some $\mu \in \{1, \ldots, s\}$, the edge $\{C_\mu, x\}$ does not belong to T. We obtain

  $$\sum_{v \in V} d_T(C_\mu, v) \geq 4a + 2s + 6m - 5 = 6s + 54m + 27 > \gamma,$$

  a contradiction.

- Assume to the contrary that the second condition in Lemma 3.1 does not hold, i.e., there is a vertex l_μ not adjacent to any vertex C_ν in T. Then

  $$\sum_{v \in V} d_T(l_\mu, v) \geq 4a + 2s + 3m + 2 = 6s + 51m + 34 > \gamma,$$

  a contradiction.

- Assume to the contrary that the third condition in Lemma 3.1 does not hold, i.e., there is a vertex C_μ having two or three neighbors in T, and remember that the first two conditions imply that there is no edge between any two vertices in L. Let l_ν be one of C_μ's neighbors in T. We calculate

$$\sum_{v \in V} d_T(l_\nu, v) \geq 3a + 3s + 12m - 5 = 6s + 48m + 19 > \gamma,$$

  a contradiction.

This proves the theorem by Lemma 3.1. □

Remark 3.1. *Both constructions in Theorem 3.1 and Theorem 3.2 do not rely on using the edges between the vertices in L of the graph representation of an* X3C *instance. Consequently, the constructed graphs are planar (if we assume that all clauses in the* X3C *instance are distinct). This means that computing distance-minimizing spanning trees for these norms is* NP*-hard already in planar graphs.*

3.2.3 Distance-approximating spanning trees

We turn to the problem of finding spanning trees approximating the distances in a graph reasonably well under a certain given matrix norm.

Problem:	DAST (with respect to $\|.\|_r$).
Input:	A connected graph G and an algebraic number γ.
Question:	Is there a spanning tree T of G with $\|D_T - D_G\|_r \leq \gamma$?

Independent of the norm, we show all problems to be NP-complete. In was already mention in the introductory section to this chapter that with respect to L_1 matrix-norm, computing distance-minimizing and optimal distance-approximating spanning trees is equivalent. An immediate consequence is NP-completeness under L_1 matrix-norm (from Theorem 3.1 or [JLRK78]).

3.2.3.1 DAST with respect to the $\|\cdot\|_{L,p}$ and $\|\cdot\|_1$ matrix norms

Theorem 3.3. DAST *with respect to* $\|\cdot\|_{L,p}$ *is* NP*-complete for all* $p \in \mathbb{N}_+$.

Proof. Containment in NP is obvious. NP-hardness is proved by reduction from X3C using the graph representation $G_{a,0}(\mathcal{C}, L)$ for a given X3C instance $(\mathcal{C}, L)$ and an appropriate choice of the parameters a and γ. We will fix the parameter later, so that $(\mathcal{C}, L)$ has an admissible solution $\mathcal{S} \subseteq \mathcal{C}$ if and only if $G_{a,0}(\mathcal{C}, L)$ has a spanning tree T such that $\|D_{G_{a,0}(\mathcal{C},L)} - D_T\|_{L,p}^p \leq \gamma^p$.

Suppose $\mathcal{S} \subseteq \mathcal{C}$ is an admissible solution to $(\mathcal{C}, L)$. Let $T_\mathcal{S}$ be the corresponding spanning tree in $G_{a,0}(\mathcal{C}, L)$. By Proposition 3.2, we only have $d_{T_\mathcal{S}}(l_\mu, l_\nu) - d_{G_{a,0}(\mathcal{C},L)}(l_\mu, l_\nu) > 0$ and

$d_{T_\mathcal{S}}(l_\mu, C_\nu) - d_{G_{a,0}(\mathcal{C},L)}(l_\mu, C_\nu) > 0$. Thus,

$$\|D_{G_{a,0}(\mathcal{C},L)} - D_{T_\mathcal{S}}\|_{L,p}^p = 2\left(\underbrace{2^p 3m(s-m) + 3m(m-1)}_{\text{C to L}} + \underbrace{3m + 3^p\frac{9m^2-9m}{2}}_{\text{L to L}}\right).$$

This can be seen as follows: there are $6ms$ ordered pairs of vertices, where one vertex is in C and the other in L. For $6m$ such pairs, the distance in T does not increase at all. For another $6m(s-m)$ of these pairs, the p-distance increases by 2^p for each pair and for the rest it increases by one. Also, there are $9m^2$ ordered pairs of vertices, such that both are in L. For $6m$ such pairs, the p-distance increases by one and for the rest of the pairs it increases by 3^p. We now set our parameters as follows:

$$\begin{aligned} a &=_{\text{def}} \gamma \\ \gamma &=_{\text{def}} \|D_{G_{a,0}(\mathcal{C},L)} - D_{T_\mathcal{S}}\|_{L,p} \end{aligned}$$

Note that computing γ (and thus a) in polynomial time is possible, since all information needed is already given in the input: as it was the case in the reduction used in Theorem 3.1, the difference $\|D_{G_{a,0}(\mathcal{C},L)} - D_{T_\mathcal{S}}\|_{L,p}$ does not depend on a particular solution $\mathcal{S}$, but only on the size of $(\mathcal{C}, L)$. Now, by definition, $T_\mathcal{S}$ is a spanning tree in $G_{a,0}(\mathcal{C}, L)$ having the desired distance property.

Suppose T is a spanning tree in $G_{a,0}(\mathcal{C}, L)$ satisfying $\|D_{G_{a,0}(\mathcal{C},L)} - D_T\|_{L,p}^p \leq \gamma^p$. We apply the characterization of a solution tree given in Lemma 3.1 and show that all conditions are satisfied.

- Assume to the contrary that the first condition in Lemma 3.1 does not hold, i.e., for some $\mu \in \{1, \ldots, s\}$, the edge $\{C_\mu, x\}$ does not belong to T. This implies $d_T(C_\mu, x) \geq d_{G_{a,0}(\mathcal{C},L)}(C_\mu, x) + 2$ and for all $\nu \in \{1, \ldots, a\}$, $d_T(C_\mu, r_\nu) \geq d_{G_{a,0}(\mathcal{C},L)}(C_\mu, r_\nu) + 2$. Thus,

  $$\|D_{G_{a,0}(\mathcal{C},L)} - D_T\|_{L,p}^p \geq (a+1)2^p > \gamma^p,$$

  a contradiction.

- Assume to the contrary that the second condition in Lemma 3.1 does not hold, i.e., there is a vertex l_μ not adjacent to any vertex C_ν in T. We obtain $d_T(l_\mu, x) \geq d_{G_{a,0}(\mathcal{C},L)}(l_\mu, x) + 1$ and for all $\nu \in \{1, \ldots, a\}$, $d_T(l_\mu, r_\nu) \geq d_{G_{a,0}(\mathcal{C},L)}(l_\mu, r_\nu) + 1$. This gives

  $$\|D_{G_{a,0}(\mathcal{C},L)} - D_T\|_{L,p}^p \geq a + 1 > \gamma^p,$$

  a contradiction.

- Note that, if the first and second condition in Lemma 3.1 are both satisfied, then all edges but those between $\mathcal{C}$ and L are already fixed by now and the distances in T and $G_{a,0}(\mathcal{C}, L)$ are the same. For the distances from vertices in $\mathcal{C}$ to vertices in L we have

  $$d_T(l_\mu, C_\nu) = \begin{cases} d_{G_{a,0}(\mathcal{C},L)}(l_\mu, C_\nu) & \text{if edge } \{l_\mu, c_\nu\} \text{ is in } T \\ d_{G_{a,0}(\mathcal{C},L)}(l_\mu, C_\nu) + 1 & \text{if edge } \{l_\mu, c_\nu\} \text{ is not in } T \text{ and } l_\mu \notin C_\nu \\ d_{G_{a,0}(\mathcal{C},L)}(l_\mu, C_\nu) + 2 & \text{if edge } \{l_\mu, c_\nu\} \text{ is not in } T \text{ and } l_\mu \in C_\nu \end{cases}$$

Let h_i be the number of vertices C_μ having exactly i neighbors in T. It clearly holds $h_1 + h_2 + h_3 + h_4 = s$ and $h_2 + 2h_3 + 3h_4 = 3m$. Moreover, we have

$$\sum_{\mu=1}^{3m}\sum_{\nu=1}^{s}\left(d_{G_{a,0}(\mathcal{C},L)}(l_\mu, C_\nu) - d_T(l_\mu, C_\nu)\right)^p = 2\left(3s(m-1) + 2^p(3h_1 + 2h_2 + h_3)\right).$$

For the distances between vertices in L, we obtain for $\mu, \nu \in \{1, \ldots, 3m\}$ and $\mu \neq \nu$,

$$d_T(l_\mu, l_\nu) = \begin{cases} d_{G_{a,0}(\mathcal{C},L)}(l_\mu, l_\nu) + 1 & \text{if edges } \{l_\mu, C_\kappa\} \text{ and } \{l_\nu, C_\kappa\} \text{ belong to } T \\ & \text{for some } \kappa \in \{1, \ldots, s\} \\ d_{G_{a,0}(\mathcal{C},L)}(l_\mu, l_\nu) + 3 & \text{otherwise} \end{cases}$$

Let g be the number of pairs (l_μ, l_ν) such that $\mu \neq \nu$ and for some $\kappa \in \{1, \ldots, s\}$, the edges $\{l_\mu, C_\kappa\}$ and $\{l_\mu, C_\kappa\}$ exist in T. We calculate

$$\sum_{\mu=1}^{3m}\sum_{\nu=1}^{3m}\left(d_{G_{a,0}(\mathcal{C},L)}(l_\mu, l_\nu) - d_T(l_\mu, l_\nu)\right)^p = g + 3^p(9m^2 - 3m - g).$$

The maximum possible value of g is $6m$ and that of h_4 is s both values simultaneously corresponding to the case that the third condition in Lemma 3.1 is satisfied. Assume to the contrary $g < 6m$ and $h_4 < s$. Then we have

$$\begin{aligned} &\|D_{G_{a,0}(\mathcal{C},L)} - D_T\|_{L,p}^p - \gamma^p \geq \\ &\quad - 6s(m-1) - 6m - 3^p(9m^2 - 9m) + 6s(m-1) + 2^{p+1} + g + 3^p(9m^2 - 3m - g) \\ &\qquad = 2^{p+1} + (3^p - 1)(6m - g) > 1, \end{aligned}$$

a contradiction.

This proves the theorem by Lemma 3.1. □

Theorem 3.4. DAST *with respect to* $\|\cdot\|_1$ *is* NP*-complete.*

Proof. Containment in NP is obvious. We prove NP-hardness by reduction from X3C using the graph representation $G_{a,b}(\mathcal{C}, L)$ for a given X3C instance $(\mathcal{C}, L)$ and an appropriate choice of the parameters a, b and γ. We will fix the parameter later, so that $(\mathcal{C}, L)$ has an admissible solution $\mathcal{S} \subseteq \mathcal{C}$ if and only if $G_{a,b}(\mathcal{C}, L)$ has a spanning tree T such that $\|D_{G_{a,b}(\mathcal{C},L)} - D_T\|_1 \leq \gamma$. We may suppose that $s \geq 1$ and thus $m \geq 1$. For each $\mu \in \{1, \ldots, 3m\}$, let $h(\mu)$ denote the number of sets of $\mathcal{C}$ in which l_μ appears, i.e., $h(\mu) = \|\{\nu \ : \ l_\mu \in C_\nu\}\|$. Define $h_{max} =_{\text{def}} \max_\mu h(\mu)$.

Suppose $\mathcal{S} \subseteq \mathcal{C}$ is an admissible solution to $(\mathcal{C}, L)$. Let $T_\mathcal{S}$ be the corresponding spanning tree in $G_{a,b}(\mathcal{C}, L)$. The vertices in the sets R, X, L, and K all have the same column sums. We calculate for $\mu \in \{1, \ldots, s\}$, $\nu \in \{1, \ldots, 3m\}$, and $\kappa \in \{1, \ldots, b\}$:

$$\begin{aligned} \sum_{v \in V} d_T(r_\mu, v) - d_{G_{a,b}(\mathcal{C},L)}(r_\mu, v) &= 0 \\ \sum_{v \in V} d_T(x, v) - d_{G_{a,b}(\mathcal{C},L)}(x, v) &= 0 \end{aligned}$$

$$\begin{aligned}\sum_{v\in V} d_T(l_\nu, v) - d_{G_{a,b}(\mathcal{C},L)}(l_\nu, v) &= s + h(\nu) + 9m - 9 + b(9m-7)\\ \sum_{v\in V} d_T(k_{\nu,\kappa}, v) - d_{G_{a,b}(\mathcal{C},L)}(k_{\nu,\kappa}, v) &= s + h(\nu) + 9m - 9 + b(9m-7)\end{aligned}$$

For vertices in $\mathcal{C}$ we have to make a distinction between vertices with one neighbor in T and vertices with four neighbors in T. We obtain for $\mu \in \{1,\ldots,s\}$:

$$\sum_{v\in V} d_T(C_\mu, v) - d_{G_{a,b}(\mathcal{C},L)}(C_\mu, v) = \begin{cases} (3m+3)(b+1) & \text{if } C_\mu \text{ has one neighbor in } T_{\mathcal{S}} \\ (3m-3)(b+1) & \text{if } C_\mu \text{ has four neighbors in } T_{\mathcal{S}} \end{cases}$$

We set our parameters in the following way:

$$\begin{aligned} a &=_{\text{def}} \gamma + 1 \\ b &=_{\text{def}} s + 1 \\ \gamma &=_{\text{def}} s + h_{\max} + 9m - 9 + b(9m-7) \end{aligned}$$

This gives $\|D_{G_{a,b}(\mathcal{C},L)} - D_T\|_1 = s + h_{\max} + 9m - 9 + b(9m-7) = \gamma$. Consequently, $T_{\mathcal{S}}$ is a spanning tree of $G_{a,b}(\mathcal{C}, L)$ having the desired distance property.

Suppose T is a spanning tree in $G_{a,b}(\mathcal{C}, L)$ satisfying $\|D_{G_{a,b}(\mathcal{C},L)} - D_T\|_1 \leq \gamma$. We apply the characterization of a solution tree given in Lemma 3.1 and show that all conditions are satisfied.

- Assume to the contrary that the first condition in Lemma 3.1 does not hold, i.e., for some $\mu \in \{1,\ldots,s\}$, the edge $\{C_\mu, x\}$ does not belong to T. Then

  $$\sum_{v\in V} d_T(C_\mu, v) - d_{G_{a,b}(\mathcal{C},L)}(C_\mu, v) \geq 4a > \gamma,$$

  a contradiction.

- Assume to the contrary that the second condition in Lemma 3.1 does not hold, i.e., there is a vertex l_μ not adjacent to any vertex C_ν in T. Then

  $$\sum_{v\in V} d_T(l_\mu, v) - d_{G_{a,b}(\mathcal{C},L)}(l_\mu, v) \geq a > \gamma,$$

  a contradiction.

- Note that, if the first and second condition in Lemma 3.1 are both satisfied, then all edges but those between $\mathcal{C}$ and L are already fixed by now and the distances in T and $G_{a,b}(\mathcal{C}, L)$ are the same. Assume to the contrary that the third condition in Lemma 3.1 does not hold, i.e., there is a vertex C_μ having two or three neighbors in T. Let l_ν be a neighbor of such a vertex C_μ and let $\deg_T(C_\mu)$ denote the number of neighbors of C_μ in T. Then, we conclude

  $$\begin{aligned}&\sum_{v\in V} d_T(l_\nu, v) - d_{G_{a,b}(\mathcal{C},L)}(l_\nu, v)\\ &\quad= s + h(l_\nu) - 2 + \deg_T(C_\mu) - 2 + 3(3m - \deg_T(C_\mu) + 1)(b+1)\\ &\quad= \gamma - s - h_{max} - 9m + 9 - b(9m-7)s + h(l_\nu) - 4 + \deg_T(C_\mu) +\end{aligned}$$

$$
\begin{aligned}
& & & + 3(3m - \deg_T(C_\mu) + 1)(b+1) \\
&= & \gamma + (h(l_\nu) - h_{max}) + 8 - 2\deg_T(C_\mu) + b(10 - 3\deg_T(C_\mu)) \\
&\geq & \gamma - s + b \quad > \quad \gamma,
\end{aligned}
$$

a contradiction.

This proves the theorem by Lemma 3.1. □

3.2.3.2 DAST with respect to the $\|.\|_{L,\infty}$ norm

We next consider the DAST problem with respect to the $\|.\|_{L,\infty}$ norm. As we have already mentioned, this version of the DAST problem is also known as the k-Additive-Tree-Spanner problem and its complexity has been unknown till now to the best of our knowledge. Unfortunately, we have not been able to adapt the X3C gadget (which has been used in all other reductions in this chapter) in order to proof the NP-completeness of this problem and therefore we had to introduce the second gadget, i.e., the graph representation of 2-Hitting Set (2HS).

Before we actually prove the NP-completeness, we first need the following lemma, which states that forcing some of the edges into the spanning tree does not change the nature of the problem: assume that we are given a graph $G = (V, E)$ and a parameter $k \in \mathbb{N}$, $k \geq 3$. Let $\{v, w\}$ be an arbitrary non-bridge edge in G. Let G' be the graph that results from attaching a path $(v, u_1, \ldots, u_k, w)$ to G. Then it holds that G has a spanning tree T that contains the edge $\{v, w\}$ such that $\|D_T - D_G\|_{L,\infty} \leq k$ if and only if G' contains a spanning tree T' such that $\|D_{T'} - D_G\|_{L,\infty} \leq k$. Moreover, every such T' must contain the edge $\{v, w\}$. A similar technique with two cycles was used in [Cai94, Lemma 3] to guarantee that any minimum t-spanner (i.e., a spanning subgraph with smallest number of edges such that $d_G(u, v) \leq t \cdot d_T(u, v)$ for all $u, v \in V$) contains a certain edge. However, this construction does not work in the context of additive distance growth and trees.

Lemma 3.3 (Cycle assembly)**.** *Let $G = (V, E)$ be any graph and let $\{v, w\}$ be an arbitrary non-bridge edge in G. For $k > 3$, let G' be the graph resulting from adding a path $(v, u_1, \ldots, u_k, w)$ to G where $u_\mu \notin V$ for all $\mu \in \{1, \ldots, k\}$. There exists a spanning tree T of G which includes the edge $\{v, w\}$ and satisfies $\|D_T - D_G\|_{L,\infty} \leq k$ if and only if there exists a spanning tree T' of G' such that $\|D_{T'} - D_{G'}\|_{L,\infty} \leq k$.*

Proof. For any spanning tree T of a graph $G = (V, E)$, we define $\delta_T(v) =_{\text{def}} \max_{u \in V}(d_T(v, u) - d_G(v, u))$. Let P be the path $(v, u_1, \ldots, u_k, w)$ to be added to the graph $G = (V, E)$ with respect to the edge $\{v, w\}$. That is, $G' = G \cup P$. We prove the two directions separately.

($\Rightarrow$) Suppose there is a spanning tree T of G such that $\|D_T - D_G\|_{L,\infty} \leq k$ and edge $\{v, w\}$ belongs to T. Without loss of generality, we assume that $\delta_T(v) \leq \delta_T(w)$. Define T' to be the spanning tree in G' with edge set $E(T) \cup E(P)$ by removing the edge $\{u_{\lfloor \frac{k}{2} \rfloor}, u_{\lceil \frac{k+1}{2} \rceil}\}$ in the middle of P. We have two cases.

- Suppose $\delta_T(v) < k$. We have the following bounds on distance changes in T' with respect to G'.
  - For $x, y \in V(G)$ we have $d_{T'}(x, y) \leq d_{G'}(x, y) + k$.
  - For $x, y \in V(P)$ we have $d_{T'}(x, y) \leq d_{G'}(x, y) + k$.

– For $\mu \in \{1, \ldots, \lfloor \frac{k}{2} \rfloor\}$ and $y \in V(G)$ we find

$$\begin{aligned} d_{T'}(u_\mu, y) - d_{G'}(u_\mu, y) &= d_{T'}(u_\mu, v) + d_{T'}(v, y) - d_{G'}(u_\mu, v) - d_{G'}(v, y) \\ &= d_{T'}(v, y) - d_{G'}(v, y) \leq k. \end{aligned}$$

– For $\mu \in \{\lceil \frac{k+1}{2} \rceil, \ldots, k\}$ and $y \in V(G)$, we have a similar inequality, if the shortest path from u_μ to y in G' contains vertex w. Otherwise, we obtain

$$\begin{aligned} d_{T'}(u_\mu, y) - d_{G'}(u_\mu, y) &= d_{T'}(u_\mu, v) + d_{T'}(v, y) - d_{G'}(u_\mu, v) - d_{G'}(v, y) \\ &= 1 + d_{T'}(v, y) - d_{G'}(v, y) \leq 1 + (k-1) = k. \end{aligned}$$

This completes the first case.

- Suppose $\delta_T(v) = \delta_T(w) = k$. For $k \geq 0$ and for any vertex $z \in V$, define $B_{=k}(z) =_{\text{def}} \{x \in V \ : \ d_T(x, z) - d_G(x, z) = k\}$. First, we consider vertices $x, y \in B_{=k}(v) \cup B_{=k}(w)$ and claim that

  – either $d_T(v, x) = d_T(w, x) + 1$ and $d_T(v, y) = d_T(w, y) + 1$
  – or $d_T(w, x) = d_T(v, x) + 1$ and $d_T(w, y) = d_T(v, y) + 1$.

  Assume to the contrary that this is not true, i.e., we have $d_T(v, x) = d_T(w, x) + 1$ and $d_T(w, y) = d_T(v, y) + 1$ (which indeed constitutes the contrary to the assumption, because $|d_T(v, x) - d_T(w, x)| = 1$ for every vertex $x \in V$ and adjacent x, w. Also, by symmetry, it is enough consider this situation.) Now we may conclude that a path from x to y in T must pass v and w where x is nearer to w and y is nearer to v. Hence,

$$\begin{aligned} & d_T(x, y) - d_G(x, y) \\ &= d_T(x, w) + 1 + d_T(v, y) - d_G(x, y) \\ &\geq d_T(x, w) + 1 + d_T(v, y) - d_G(x, w) - d_G(v, y) - 1 && \text{(by triangle inequality)} \\ &\geq d_T(x, w) - d_G(x, w) + d_T(v, y) - d_G(v, y) - 2 && \text{(edge } \{v, w\} \text{ belongs to } T) \\ &\geq 2k - 4 && \text{(since } x, y \in B_{=k}(v) \cup B_{=k}(w)) \end{aligned}$$

  For $k > 4$ this leads to a contradiction and thus, our claim is true in this case. The case $k = 4$ will be treated separately below.

  So, without loss of generality, we suppose that $d_T(v, x) = d_T(w, x) + 1$ and $d_T(v, y) = d_T(w, y) + 1$. We obtain the following distance changes in T' with respect to G'

  – For $x, y \in V(G)$ we trivially have $d_{T'}(u_\mu, y) \leq d_{G'}(u_\mu, y) + k$.
  – For $\mu \in \{1, \ldots, \lfloor \frac{k}{2} \rfloor\}$ and $y \in V(G)$, the shortest path between u_μ and y visits v. Thus, $d_{T'}(u_\mu, y) \leq d_{G'}(u_\mu, y) + k$.
  – For $\mu \in \{\lceil \frac{k+1}{2} \rceil + 1, \ldots, k\}$ and $y \in V(G)$, the shortest path between u_μ and y visits w. Thus, $d_{T'}(u_\mu, y) \leq d_{G'}(u_\mu, y) + k$.
  – For $\mu = \lceil \frac{k+1}{2} \rceil$ and $y \in B_{=k}(v) \cup B_{=k}(w)$ we know from above that the shortest path between u_μ and y visits w and hence, $d_{T'}(u_\mu, y) \leq d_{G'}(u_\mu, y) + k$. For $y \notin B_{=k}(v) \cup B_{=k}(w)$ we obtain

$$d_{T'}(u_\mu, y) - dG'(u_\mu, y) = d_{T'}(u_\mu, v) + d_{T'}(v, y) - d_{G'}(u_\mu, v) - d_{G'}(v, y)$$

$$\leq \quad 1 + (k-1) \quad = \quad k$$

Finally, for $k = 4$, note that $d_T(u_\mu, v) = d_G(u_\mu, v)$ and $d_T(u_\mu, w) = d_G(u_\mu, w)$, if we remove the edge $\{u_2, u_3\}$. Hence,

$$d_{T'}(u_\mu, y) - d_{G'}(u_\mu, y) = \min\left(d_T(v, y) - d_G(v, y), d_T(w, y) - d_G(w, y)\right) \leq k.$$

This completes the second case.

($\Leftarrow$) Suppose there is a spanning tree T' for G' with $\|D_{T'} - D_{G'}\|_{L,\infty} \leq k$. We show that for any such tree, the edge $\{v, w\}$ must be in T'. Note that $\{v, w\}$ is in at least two cycles in G', where one is a cycle with P and another one is the cycle making $\{v, w\}$ a non-bridge-edge in G. These cycles must be broken in order for T' to be a tree. We show that there is only one possibility to break these cycles (see Figure 3.5 for an illustration):

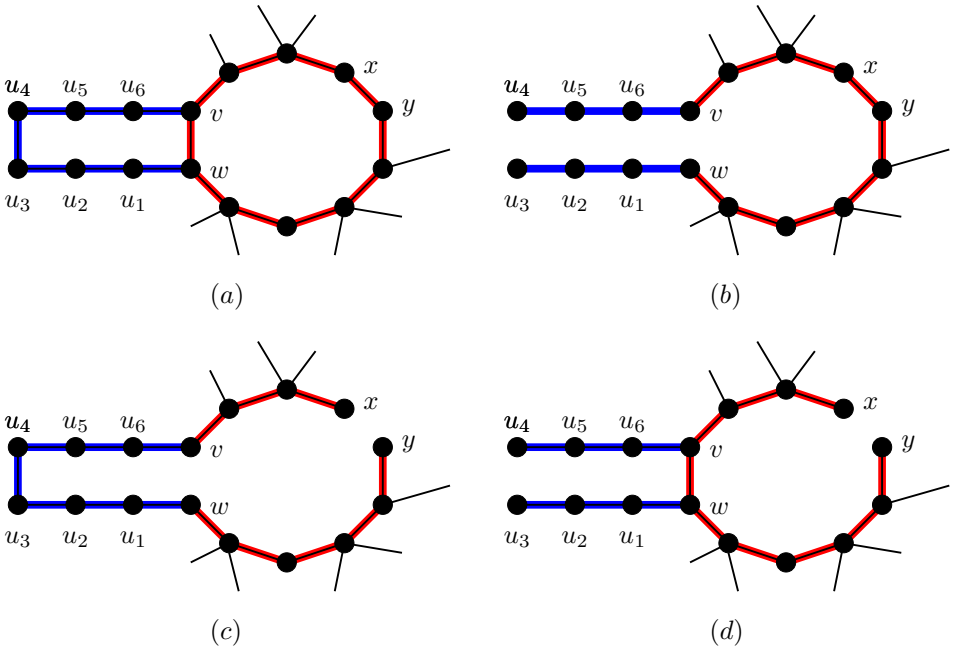


Figure 3.5: An illustration of the cycle-assembly operation as described in Lemma 3.3 for $k = 6$. Assume that removing the edge $\{x, y\}$ results in a tree containing the edge $\{v, w\}$ which satisfies the distance constraints. Part (a)T: the red lines indicate a cycle in the original graph, the blue lines indicate the newly attached cycle of length 6. Part (b): after deleting the edges $\{u_3, u_4\}$ and $\{v, w\}$ in some tree T, it clearly holds that $d_T(x, y) > d_{G'}(x, y) + k$. Part (c): after deleting the edges $\{v, w\}$ and $\{x, y\}$ in some tree T, it clearly holds that $d_T(u_3, u_4) > d_{G'}(u_3, u_4) + k$. Part (d): after deleting the edges $\{u_3, u_4\}$ and $\{x, y\}$ in some tree T, all distance constraints are again satisfied.

- Breaking the cycle in P at $\{u_\mu, u_{\mu+1}\}$ and the cycles in G at $\{v, w\}$ yields

$$d_{T'}(u_\mu, u_{\mu+1}) - d_{G'}(u_\mu, u_{\mu+1}) \quad = \quad d_{T'}(u_\mu, u_{\mu+1}) - 1$$

$$\begin{aligned} &= d_{T'}(u_\mu, v) + d_{T'}(v, w) + d_{T'}(w, u_{\mu+1}) - 1 \\ &\geq d_{T'}(u_\mu, v) + 2 + d_{T'}(w, u_{\mu+1}) - 1 \\ &= k + 1 \quad > \quad k, \end{aligned}$$

a contradiction.

- Breaking the cycles in P at $\{v, w\}$ and any of the other one at an arbitrary edge, say $\{x, y\}$ with $y \notin \{v, w\}$ yields

$$\begin{aligned} d_{T'}(x, y) - d_{G'}(x, y) &= d_{T'}(x, y) - 1 \\ &= d_{T'}(x, v) + d_{T'}(v, w) + d_{T'}(w, y) - 1 \\ &\geq d_{T'}(x, v) + k + 1 \quad > \quad k, \end{aligned}$$

again a contradiction.

It follows that when breaking the cycle in G at any edge $e \neq \{u, w\}$ and the cycle with P at the edge $\{u_{\lfloor \frac{k}{2} \rfloor}, u_{\lceil \frac{k+1}{2} \rceil}\}$, we can "reverse" the assembly - that is we can omit the part of T' that spans P, and thus obtain a tree T of G for which $\|D_T - D_G\|_{L,\infty} \leq k$ and $\{v, w\}$ is an edge in T. This proves the lemma. □

Theorem 3.5. DAST *with respect to* $\|\cdot\|_{L,\infty}$ *is* NP*-complete*

Proof. The containment in NP is obvious. We prove the hardness by reduction from 2HS using the graph representation $G(\mathcal{E}, \mathcal{V})$. Let $(\mathcal{E}, \mathcal{V}, k)$ be an instance of 2HS and define

$$\gamma =_{\text{def}} 2N + 3 + n(m + 2) + k,$$

where N is the number of edges of the graph $G(\mathcal{E}, \mathcal{V})$. Define the graph $G = (V, E)$ such that V consists of the vertices of the graph $G(\mathcal{E}, \mathcal{V})$ and additionally two vertices a and b and two vertex sets $\{a_1, \ldots, a_N\}$ and $\{b_1, \ldots, b_N\}$ and such that E consists of all edges in $G(\mathcal{E}, \mathcal{V})$ and additionally the edges $\{a, b\}$, $\{a, a'\}$ and $\{b, b'\}$ and the edge sets

$$\{\{a, a_1\}, \{a_N, a'\}\} \cup \{\{a_i, a_{i+1}\} \ : \ i \in \{1, \ldots, N - 1\}\}$$

and

$$\{\{b, b_1\}, \{b_N, b'\}\} \cup \{\{b_i, b_{i+1}\} \ : \ i \in \{1, \ldots, N - 1\}\}.$$

Let $A = (a, a_1, \ldots, a_N, a')$ and $A = (b, b_1, \ldots, b_N, b')$ be the two respective paths. The construction is illustrated in Figure 3.6. Let $\mathcal{F}$ be the set of all edges in all clause paths and in the paths A and B. We claim that$G$ has a spanning tree T containing all edges in $\mathcal{F}$ such that $\|D_T - D_G\|_{L,\infty} \leq \gamma$ if and only if $(\mathcal{E}, \mathcal{V}, k)$ has an admissible solution $\mathcal{V}'$ of size $\|\mathcal{V}'\| \leq k$. According to Lemma 3.5, we may force edges into spanning trees by a cycle assembly of length γ. Thus, the requirement that edges in $\mathcal{F}$ must be included in the solution spanning tree of G is not a restriction with respect to the requirement that $\|D_T - D_G\|_{L,\infty} \leq \gamma$.

Claim 3.1. *Let* $(\mathcal{E}, \mathcal{V}, k)$ *be an instance of* 2HS*. Then the graph* G *has a spanning tree* T *containing all edges in* $\mathcal{F}$ *such that* $\|D_T - D_G\|_{L,\infty} \leq \gamma$ *if* $(\mathcal{E}, \mathcal{V}, k)$ *has an admissible solution* $\mathcal{V}'$ *of size* $\|\mathcal{V}'\| \leq k$

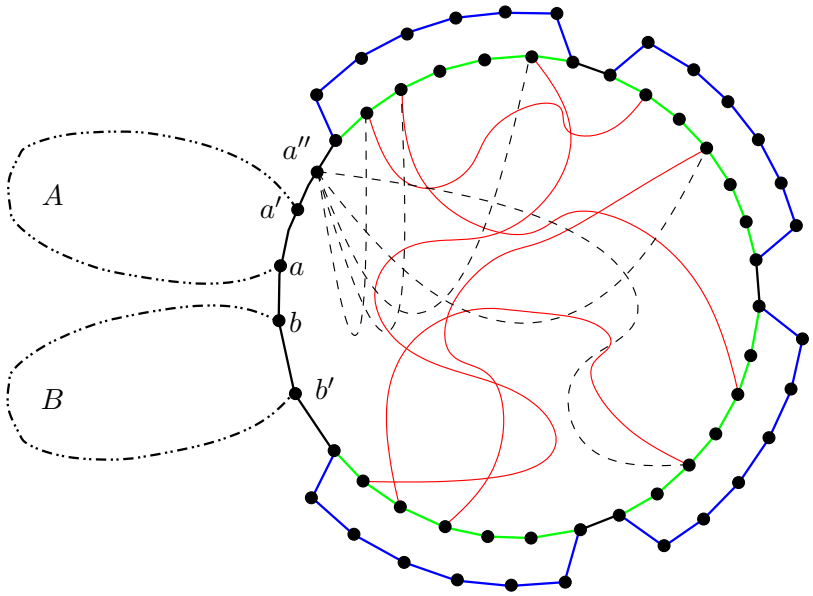


Figure 3.6: Twisted graph representation $G(\mathcal{E},\mathcal{V})$ of the 2HS instance $(\mathcal{E},\mathcal{V},2) = (\{\{v_1,v_2\},\{v_1,v_3\},\{v_2,v_4\},\{v_3,v_4\},\{v_1,v_4\}\},\{v_1,v_2,v_3,v_4\},2)$

Proof. Suppose $\mathcal{V}'$ is an admissible solution to $(\mathcal{E},\mathcal{V},k)$, i.e., $\|\mathcal{V}'\| \leq k$. Construct the spanning tree T of G as follows: for the part of G which corresponds to the graph $G(\mathcal{E},\mathcal{V})$ let $T_{\mathcal{V}'}$ be the solution tree corresponding to the admissible solution $\mathcal{V}'$ such that $T_{\mathcal{V}'}$ contains all edges in all clause paths and such that

$$d_{T_{\mathcal{V}'}}(a',b') \leq n\cdot(m+2)+2+k.$$

Such a $T_{\mathcal{V}'}$ exists according to Lemma 3.2. Let $E_{T_{\mathcal{V}'}}$ the edge set of $T_{\mathcal{V}'}$. Then the edge set E_T of the spanning tree T is composed of all edges in the paths A and B and the edges in $E_{T_{\mathcal{V}'}}$. By construction no edge in any clause path was removed and also no edge in either of the paths A and B and so this requirement is met. Second, all cycles in G are broken. Now, the *unique path* from a to b in T leads along the path A to the vertex a', along the shortest path from a' to b' in $G(\mathcal{E},\mathcal{V})$, along the path B to finally arrive at b. Together we have

$$\begin{aligned} d_T(a,b) &= 2(N+1)+d_{T_{\mathcal{V}'}}(a',b') \\ &\leq 1+2(N+1)+n(m+2)+2+k-1 \\ &= d_G(a,b)+\gamma. \end{aligned}$$

All that is left in order to prove the claim is to show that $d_T(a,b)-d_G(a,b)$ determines the value $\|D_T-D_G\|_{L,\infty}$: to this end, first note that for all pairs of vertices u and v from the set of vertices induced by the paths A and B not including a and b we have that $d_T(u,v)\leq d_T(a,b)$ and $d_G(u,v)\geq d_G(a,b)=1$. Therefore we have

$$d_T(u,v)-d_G(u,v)\leq d_T(u,v)-d_G(a,b)<d_T(a,b)-d_G(a,b).$$

For vertices u and v in G which are not contained in the set of vertices induced by the two paths A and B (including a and b), we have that $d_T(u,v)\leq N-1<2N\leq d_T(a,b)$ and

$d_G(u,v) \geq d_G(a,b) = 1$ and again

$$d_T(u,v) - d_G(u,v) \leq d_T(u,v) - d_G(a,b) < d_T(a,b) - d_G(a,b).$$

Therefore $d_T(a,b) - d_G(a,b)$ determines the value $\|D_T - D_G\|_{L,\infty}$ which proves the Claim. □

Claim 3.2. *Let $(\mathcal{E}, \mathcal{V}, k)$ be an instance of* 2HS. *Then $(\mathcal{E}, \mathcal{V}, k)$ has an admissible solution $\mathcal{V}'$ of size $\|\mathcal{V}'\| \leq k$ if the graph G has a spanning tree T containing all edges in $\mathcal{F}$ such that $\|D_T - D_G\|_{L,\infty} \leq \gamma$.*

Proof. Suppose T is some spanning tree of G containing all edges of all clause paths and all edges in the paths A and B satisfying $\|D_T - d_G\|_{L,\infty} \leq \gamma$.

First, assume that T contains all three edges $\{a,b\}$, $\{a',a''\}$ and $\{t_n,b'\}$. We claim that this implies that $\|D_T - D_G\|_{L,\infty} > \gamma$, a contradiction. To this end, let T be such a tree that contains all three edges $\{a,b\}$, $\{a',a''\}$ and $\{t_n,b'\}$ and consider the clause path corresponding to the clause $E_m = \{v_\mu, v_\nu\}$ from l_m^μ to l_m^ν. This path is not broken in T by assumption. Consider the shortest induced cycle in G that contains the paths A, B, the edges $\{a,b\}$, $\{a',a''\}$ and $\{t_n,b'\}$ and the respective clause path from l_m^μ to l_m^ν. It has length at least $2n(m+2)+2(N+1)+4$ and must be broken in T at some edge other that $\{a,b\}$, $\{a',a''\}$ and $\{t_n,b'\}$. Thus, for the two endpoints of this edge, say u and v, it holds that

$$d_T(u,v) - d_G(u,v) \geq 2n(m+2) + 2(N+1) + 3 > \gamma,$$

a contradiction. Thus, one of the three edges $\{a,b\}$, $\{a',a''\}$ and $\{t_n,b'\}$ is not contained in T, but the other two edges are (this is necessary for T to be a *spanning* tree). We consider the following cases.

(i) Assume that T does not contain the edge $\{a,b\}$. The value of $\|D_T - D_G\|_{L,\infty}$ is again determined by the distance between a and b in T by the same reasoning as in the proof of the previous claim. Now, suppose that $d_T(a,b) - d_G(a,b) \leq \gamma = 2N + n(m+2) + 3 + k$. Since

$$d_T(a,b) = 2(N+1) + d_T(a',b') \leq 2N + n(m+2) + 3 + k + 1,$$

it follows that

$$d_T(a',b') \leq 2 + n \cdot (m+2) + k.$$

Further, the part of T which is induced by vertices in $G(\mathcal{V},\mathcal{E})$ gives a spanning tree T' for $G(\mathcal{V},\mathcal{E})$ such that

$$d_{T'}(a',b') - d_{G(\mathcal{V},\mathcal{E})} \leq k.$$

From Lemma 3.2 we know that such if such a spanning tree T' exists then there is a solution $\mathcal{V}'$ for the instance $(\mathcal{E},\mathcal{V},k)$ such that $|\mathcal{V}'| \leq k$, which can easily be read off the broken literal paths.

(ii) Assume that T contains the edge $\{a,b\}$ and exactly one of the edges, either $\{a',a''\}$ or $\{t_n,b'\}$: then for the respective pair of vertices of the other edge the same considerations apply as in case (i) for the pair a and b: their increase in distance determines the value of $\|D_T - D_G\|_{L,\infty}$ and from the shortest path connecting them in T we can read off the corresponding hitting set $\mathcal{V}'$.

This proves the claim. □

This proves the theorem. □

The above gadget also gives at hand an alternative proof for the hardness of the following variant of the *maximum-stretch spanning tree problem*: decide on input an undirected graph $G = (V, E)$ and an algebraic number γ whether or not there exists a spanning tree $T = (V, E_T)$ of G such that

$$\max_{(u,v)\in E} d_T(u,v)/d_G(u,v) \leq \gamma.$$

The NP-completeness of the above decision problem follows readily from Claims 3.1 and 3.2 and the next observation.

Observation 3.1. *Let $(\mathcal{E}, \mathcal{V}, k)$ be an instance of* 2HS. *For every solution tree T of the graph G considered in the proof of Theorem 3.5 we had*

$$\|D_T - D_G\|_{L,\infty} = \max_{(u,v)\in E} d_T(u,v)/d_G(u,v) - 1.$$

Remark 3.2. *Note that the construction given in the original* NP*-completeness proof for the above problem [Cai94, CC95] is such that the other direction, i.e., the* NP*-completeness of* DAST *with respect to $\|.\|_{L,\infty}$, does not directly follow, nor is there an easy modification which yields the result.*

3.2.4 Centrality-approximating spanning trees

Finally, we consider the problem of finding a spanning tree of a graph that approximates the centralities vector c_G of the graph. Remember that for a graph G and a vertex $v \in V$, the closeness-centrality is defined as

$$c_G(v) =_{\text{def}} \Big(\sum_{u\in V} d_G(u,v)\Big)^{-1}.$$

(see (3.1)) and for any graph G and any subgraph $G' \subseteq G$ we have that $c_G(v) \geq c_{G'}(v)$ for all vertices. For the case of centrality approximating spanning trees, we particularly consider the decision versions with respect to the L_p *vector norm.*

Problem:	CAST (with respect to $\|.\|_r$).
Input:	A connected graph G and an algebraic number γ.
Question:	Is there a spanning tree T of G with $\|c_G - c_T\|_r \leq \gamma$?

Theorem 3.6. CAST *with respect to $\|\cdot\|_1$ is* NP*-complete.*

Proof. Containment in NP is obvious. We prove NP-hardness by reduction from X3C using a graph representation slightly different to the one we used so far. The difference lies in the following: the graph representation $G_{a,b}(\mathcal{C}, L)$ for an X3C instance $(\mathcal{C}, L)$ has edges $\{l_\mu, l_\nu\}$

for all pairs of literal vertices. In our new graph representation $G^*_{a,b}(\mathcal{C}, L) = (V^*, E^*)$ we omit these edges, i.e., we have

$$\begin{aligned} V^* &= V \\ E^* &= E \setminus \big\{\{l_\mu, l_\nu\} \ : \ \mu, \nu \in \{1, \ldots, 3m\} \text{ and } \mu \neq \nu\big\} \end{aligned}$$

where $G_{a,b}(\mathcal{C}, L) = (V, E)$. It is easy to see that Lemma 3.1 also holds for the new graph representation. Later we will set the parameters a, b and γ in a way that $(\mathcal{C}, L)$ has an admissible solution $\mathcal{S} \subseteq \mathcal{C}$ if and only if $G^*_{a,b}(\mathcal{C}, L)$ has a spanning tree T such that $\|c_{G^*_{a,b}(\mathcal{C},L)} - c_T\|_1 \leq \gamma$. In the following we may restrict ourselves to the cases where $m \geq 5$.

Suppose $\mathcal{S} \subseteq \mathcal{C}$ is an admissible solution to $(\mathcal{C}, L)$. Let $T_\mathcal{S}$ be the corresponding spanning tree in $G^*_{a,b}(\mathcal{C}, L)$. We obtain

$$c_{T_\mathcal{S}}(v)^{-1} = \begin{cases} 2s + 3(3 + 4b)m + 2a - 1 & \text{if } v \in R \\ s + 3(2 + 3b)m + a & \text{if } v \in X \\ 2s + 3(3 + 4b)m + 2a - 6(b + 1) - 1 & \text{if } v \in \mathcal{S} \\ 2s + 3(3 + 4b)m + 2a - 1 & \text{if } v \in \mathcal{C} \setminus \mathcal{S} \\ 3s + 3(4 + 5b)m + 3a - 8(b + 1) & \text{if } v \in L \\ 4s + 3(5 + 6b)m + 4a - 8(b + 1) - 1 & \text{if } v \in K \end{cases}$$

We set our parameters as follows:

$$\begin{aligned} a &=_{\text{def}} 3s(b + 1) + 3m(s - 1)(b + 1) + 3m(m - 1)(b + 1)^2 \\ b &=_{\text{def}} 9s + 1 \\ \gamma &=_{\text{def}} \|c_{G^*_{a,b}(\mathcal{C},L)} - c_{T_\mathcal{S}}\|_1 \end{aligned}$$

Again, i.e., as in the reduction used in Theorem 3.1, the value of γ can be calculated in time polynomial in the size of $(\mathcal{C}, L)$. Although at first sight it seems that there is some dependency on the solution $\mathcal{S}$, it (again) is easy to seen that all solutions (if they exists) have exactly the same value, which only depend on the size of $(\mathcal{C}, L)$.Thus, $T_\mathcal{S}$ is a spanning tree of $G^*_{a,b}(\mathcal{C}, L)$ having the desired centrality property.

Note that all parameters and the graph representation $G^*_{a,b}(\mathcal{C}, L)$ can be computed in polynomial time in the size of $(\mathcal{C}, L)$. In particular, it is not necessary to know exactly the vertices of $\mathcal{S}$.

Suppose that T is a spanning tree of $G^*_{a,b}(\mathcal{C}, L)$ satisfying $\|c_{G^*_{a,b}(\mathcal{C},L)} - c_T\|_1 \leq \gamma$. We compare the centrality of each vertex in the tree T with its centrality in a hypothetical solution tree for the X3C instance $(\mathcal{C}, L)$. For $v \in V$, define imitating centralities $\hat{c}(v)$ as follows: if $v \in V \setminus \mathcal{C}$, then $\hat{c}(v)$ is equal to the values $c_{T_\mathcal{S}}$ from above; for vertices $v \in \mathcal{C}$, we define

$$\hat{c}(v)^{-1} =_{\text{def}} \begin{cases} 2s + 3(3 + 4b)m + 2a - 6(b + 1) - 1 & \text{if } v \in \{C_1, \ldots, C_m\} \\ 2s + 3(3 + 4b)m + 2a - 1 & \text{if } v \in \{C_{m+1}, \ldots, C_s\}, \end{cases}$$

i.e., the clause vertices $C_1, \ldots, C_m$ simulate an admissible solution to $(\mathcal{C}, L)$. Note that $\|c_{G^*_{a,b}(\mathcal{C},L)} - \hat{c}\|_1 = \gamma$. We apply the characterization of a solution tree in Lemma 3.1 (in the version suitable for the graph representation $G^*_{a,b}(\mathcal{C}, L)$) and show that all conditions are satisfied.

- Assume to the contrary that the first condition in Lemma 3.1 does not hold, i.e., for some $\mu \in \{1, \ldots, s\}$, the edge $\{C_\mu, x\}$ does not belong to T. Simple calculations yield the following bounds on deviations from the imitating centralities.

- For $v \in R \cup X$ we obtain $c_T(v)^{-1} \geq \hat{c}(v)^{-1} + 2$. Note that this inequality is crucial in getting a contradiction as it holds for $a + 1$ vertices.
- For $v \in \mathcal{C}$ we obtain $c_T(v)^{-1} \geq \hat{c}(v)^{-1} - 6(b + 1)$.
- For $v \in L \cup K$, we have $c_T(v)^{-1} \geq \hat{c}(v)^{-1} - 2(s - 1) - 6(m - 1)(b + 1)$.

Thus, using the identity $\frac{1}{x+y} = \frac{1}{x} - \frac{y}{x(x+y)}$ which is at least true whenever $x > 0$ and $y \neq -x$, the total centrality of T can be estimated as

$$\sum_{v \in V} c_T(v) \quad \leq \quad \left(\sum_{v \in V} \hat{c}(v) \right) - \frac{2(a+1)}{(a+c_1)(a+c_2)} + \frac{A}{(2a+c_3)(2a+c_4)}$$

where c_1, c_2, c_3, c_4 and A are appropriate positive integers (that depend on s, m, and b). It is clear that the latter sum in the inequality is negative for a large enough. Inspecting the concrete values

$$\begin{aligned} c_1 &= s + 3(2 + 3b)m \\ c_2 &= s + 3(2 + 3b)m + 2 \\ c_3 &= 2s + 3(3 + 4b)m - 6(b + 1) - 1 \\ c_4 &= 2s + 3(3 + 4b)m - 12(b + 1) - 1 \\ A &= 6s(b + 1) + 6m(s - 1)(b + 1) + 18m(m - 1)(b + 1)^2, \end{aligned}$$

we see that $0 \leq c_1 \leq c_3$ and $0 \leq c_2 \leq c_4$ for $m \geq 5$. Thus, our choice of a from above is appropriate. Hence,

$$\| c_{G^*_{a,b}(\mathcal{C},L)} - c_T \|_1 > \| c_{G^*_{a,b}(\mathcal{C},L)} - \hat{c} \|_1 = \gamma,$$

a contradiction.

- The second condition of Lemma 3.1 holds because T is a spanning tree of $G^*_{a,b}(\mathcal{C}, L)$.
- Note that, if the first and second condition in Lemma 3.1 are both satisfied, then all edges but those between $\mathcal{C}$ and L are already fixed by now and the distances in T and $G_{a,b}(\mathcal{C}, L)$ are the same. Assume to the contrary that the third condition in Lemma 3.1 does not hold, i.e., there is a vertex C_μ having two or three neighbors in T. Let $\deg_T(v)$ denote the degree of vertex v in T. We consider several cases:
  - For $v \in R \cup X$ we clearly obtain $c_T(v)^{-1} = \hat{c}(v)^{-1}$.
  - For $v \in \mathcal{C}$ we have $c_T(v)^{-1} \geq \hat{c}(v)^{-1} - 6(b + 1)$.
  - For $v \in L$ it suffices to have $c_T(v)^{-1} \geq \hat{c}(v)^{-1}$.
  - For $v \in K$ we obtain $c_T(v)^{-1} \geq \hat{c}(v)^{-1} + 2(b + 1)(4 - \deg_T(u))$ where $u \in \mathcal{C}$ and T contains edges $\{v, w\}$ and $\{w, u\}$ for some $w \in L$. Note that since there is a vertex in $\mathcal{C}$ with at most three neighbors in T, there are at least b vertices in K such that $c_T(v)^{-1} \geq \hat{c}(v)^{-1} + 2(b + 1)$. This is the crucial inequality in getting a contradiction.

Using the identity $\frac{1}{x+y} = \frac{1}{x} - \frac{y}{x(x+y)}$ from above once more, we get the following estimation for the total centrality:

$$\sum_{v\in V} c_T(v) \leq \left(\sum_{v\in V} \hat{c}(v)\right) + \frac{6s(b+1)}{\hat{c}(v_0)^{-1}(\hat{c}(v_0)^{-1} - 6(b+1))} - \\ - \frac{2(b+1)b}{\hat{c}(u_0)^{-1}(\hat{c}(u_0)^{-1} + 4(b+1))},$$

where $v_0 \in \mathcal{C}$ and $u_0 \in K$. An easy estimation of the relation between $\hat{c}(v_0)^{-1}$ and $\hat{c}(u_0)^{-1}$ shows that, for $m \geq 5$, the latter difference is at most

$$\frac{18s(b+1) - 2(b+1)b}{\hat{c}(u_0)^{-1}(\hat{c}(u_0)^{-1} + 4(b+1))} < 0,$$

by our choice of b. Hence,

$$\|c_{G^*_{a,b}(\mathcal{C},L)} - c_T\|_1 > \|c_{G^*_{a,b}(\mathcal{C},L)} - \hat{c}\|_1 = \gamma,$$

a contradiction.

This proves the theorem by Lemma 3.1. □

Remark 3.3. *Observe that the graph representation used in the proof of Theorem 3.6 always produces planar graphs if* X3C *instance are assumed not to contain two or more identical clauses. That is,* CAST *with respect to* $\|\cdot\|_{L,1}$ *is* NP*-complete even when restricted to planar graphs.*

3.3 Approximating DMST

In this section, we present an efficient and simple polynomial-time 2-approximation algorithm for the optimization problem DMST with respect to the $\|.\|_{L,p}$ matrix norm for $1 \leq p < \infty$. From now on, we assume that graphs are simple and undirected but not necessarily unweighted.

Theorem 3.7. *Let* $p \in \mathbb{R}_+$ *satisfying* $1 \leq p < +\infty$*. Given an undirected non-negative weighted simple graph* $G = (V, E, w)$ *with* $\|V\| = n$ *and* $\|E\| = m$*, we can find in time* $O(n^2 \log n + nm)$ *a spanning tree* T *of* G *such that*

$$\|D_T\|_{L,p} \leq 2 \cdot \|D_G\|_{L,p} \leq 2 \cdot \|D_{\text{opt}}\|_{L,p},$$

where T_{opt} *is an optimal spanning for the problem* DMST *with respect to* $\|.\|_{L,p}$*.*

Proof. Let $G = (V, E)$ be undirected and simple. For every spanning trees T of G it clearly holds that

$$\sum_{u,v\in V} d_T(u,v)^p \geq \sum_{u,v\in V} d_G(u,v)^p. \tag{3.2}$$

Let $V = \{v_1, \ldots, v_n\}$ and let $X_1, \ldots, X_n$ be a collection of shortest-path trees rooted at $v_1, \ldots, v_n$. Let $\mu \in \{1, \ldots, n\}$ be such that

$$\sum_{u\in V} d_{X_\mu}(v_\mu, u)^p = \min_{1\leq i\leq n} \sum_{u\in V} d_{X_i}(v_i, u)^p.$$

Note that for every two spanning trees X and Y rooted at the same vertex r and all $u \in V$ it holds that $d_X(r,u) = d_Y(r,u) = d_G(r,u)$. We have that

$$n \cdot \sum_{u \in V} d_{X_\mu}(v_\mu, u)^p \leq \sum_{i=1}^{n} \sum_{u \in V} d_{X_i}(v_i, u)^p = \sum_{u,v \in V} d_G(u,v)^p. \tag{3.3}$$

Now,

$$\begin{aligned}
\|D_{X_\mu}\|_{L,p} &= \left(\sum_{u,v \in V} d_{X_\mu}(u,v)^p \right)^{1/p} \\
&\leq \left(\sum_{u,v \in V} (d_{X_\mu}(u,v_\mu) + d_{X_\mu}(v_\mu,v))^p \right)^{1/p} && (3.4) \\
&\leq \left(\sum_{u,v \in V} d_{X_\mu}(u,v_\mu)^p \right)^{1/p} + \left(\sum_{u,v \in V} d_{X_\mu}(v_\mu,v)^p \right)^{1/p} && (3.5) \\
&= 2 \cdot \left(n \cdot \sum_{u \in V} d_{X_\mu}(v_\mu,v)^p \right)^{1/p} \\
&\leq 2 \cdot \left(\sum_{u,v \in V} d_G(u,v)^p \right)^{1/p} && (3.6) \\
&= \|D_G\|_{L,p},
\end{aligned}$$

which proves the first inequality. Here, (3.4) follows from the triangle inequality, (3.5) from the Minkowski Inequality (Fact A.3) and (3.6) from Inequality (3.3). Now, let T_{opt} be a spanning tree such that $\|D_{T_{\text{opt}}}\|_{L,p} = \min_{T \in SP(G)} \|D_T\|_{L,p}$. Then (3.2) implies that $\|D_{T_{\text{opt}}}\| \geq \|D_G\|_{L,p}$ which gives the second inequality. The collection of shortest-path trees $X_1, \ldots, X_n$ can be computed in time $O(n^2 \log n + nm)$ using Dijkstra's Algorithm and the minimum tree X_μ can be found in time $O(n^2)$. The theorem follows. □

3.4 An application to bioinformatics

We close our investigations of the complexity of finding trees that approximate graph metrics under various similarity measures by giving an application to bioinformatics. Let $\mathcal{A}$ be a finite alphabet and let $S = (s_1, \ldots, s_k)$ be a set of finite strings over $\mathcal{A}$. A *multiple sequence alignment (MSA)* for S is given by a set $\bar{S} = (\bar{s}_1, \ldots, \bar{s}_k)$ over the alphabet $\mathcal{B} = \mathcal{A} \cup \{-\}$ such that all strings in $\bar{S}$ have the same length and such that for all $i \in \{1, \ldots, k\}$ the string which results from deleting all occurrences of the symbol '$-$' from $\bar{s}_i$ equals the string s_i. The *sum-of-pairs cost (SP)* of a multiple sequence alignment is defined as

$$C(\bar{S}) =_{\text{def}} \sum_{i=1}^{k} \sum_{j=i+1}^{k} \tilde{w}(\bar{s}_i, \bar{s}_j),$$

where $\tilde{w}(\bar{s}_i, \bar{s}_j)$ denotes the cost of the pair wise alignment of $(\bar{s}_i, \bar{s}_j)$ (where it is assumed that $\tilde{w}(-,-) = 0$). An *optimal MSA* for S under the SP cost function C is an MSA $\bar{S}^*$ such that

$$C(\bar{S}^*) = \min\big(C(\bar{S}) \ : \ \bar{S} \text{ is a MSA for } S\big)$$

A natural extension of the sum-of-pairs cost function is given by weighting specific pairs which may be of particular interest. Another possibility of weighting is given by the following *generalized sum-of-pairs (gSP) cost function*: let $p \in \mathbb{R}_+$ satisfying $1 \leq p < +\infty$. Then for an MSA $\bar{S}$,

$$C^{(p)}(\bar{S}) =_{\text{def}} \sum_{i=1}^{k} \sum_{j=i+1}^{k} \tilde{w}(\bar{s}_i, \bar{s}_j)^p.$$

The reasoning behind such a function is to enforce more balanced MSAs. We now describe how the 2-approximation algorithm from the last section can be used to find an 2-approximate solution to the MSA problem under the gSP objective for fixed p: for an input S let $G = (S, E, w)$ be the complete weighted graph on the input set S such that for all $s, t \in S$, $w(s,t)$ equals the cost of the optimal alignment for the pair (s,t). Let $T = (S, E_T)$ be some spanning tree of G. A *tree-driven alignment for* T is an alignment $\bar{S}$ such that for all $(s_i, s_j) \in E_T$ it holds that $(\bar{s}_i, \bar{s}_j)$ is an optimal alignment for the pair (s_i, s_j) with cost $\tilde{w}(\bar{s}_i, \bar{s}_j) = w(s_i, s_j)$. Note that for the cost of any tree driven alignment $\bar{S}$ and all $\bar{s}_i, \bar{s}_j \in \bar{S}$ it holds that $\tilde{w}(\bar{s}_i, \bar{s}_j) \leq d_T(s_i, s_j)$ for any reasonable choice of the pair wise alignments cost function (See, e.g., [Gus97]).

Let T be spanning tree of G such that $\|D_T\|_{L,p} \leq 2 \cdot \|D_G\|_{L,p}$. Such a tree can be found in time $O(\|S\|^3)$ by Theorem 3.7. Now, for the gSP cost of the alignment $\bar{S}_T$ which is driven by the tree T it holds that

$$C^{(p)}(\bar{S}_T) \leq \sum_{i=1}^{k} \sum_{j=i+1}^{k} \tilde{w}(\bar{s}_i, \bar{s}_j)^p \leq \frac{1}{2} \sum_{1 \leq i,j \leq k} d_T(\bar{s}_i, \bar{s}_j)^p = \frac{1}{2} \|D_T\|_{L,p}.$$

On the other hand, for the optimal alignment $\bar{S}^{*,p}$ under the gSP cost function we have that

$$C^{(p)}(\bar{S}^{*,p}) \geq \frac{1}{2} \|D_G\|_{L,p}.$$

Together, this gives

$$C^{(p)}(\bar{S}_T) \leq \frac{1}{2} \|D_T\|_{L,p} \leq 2 \cdot \frac{1}{2} \|D_G\|_{L,p} \leq 2 \cdot C^{(p)}(\bar{S}^{*,p}).$$

Therefore, the cost of the tree-driven MSA for the tree T is at most twice the cost of the optimal MSA. Note that the cost of computing an optimal alignment for a pairs of strings (s,t) of length l is $O(l^2)$. Thus, the graph G can be constructed efficiently.

3.5 Bibliographic notes

In this section, we review some related work that has not yet been mentioned in the course of our examinations. In addition to the already mentioned minimum diameter spanning trees [CGM80, HT95] and MAD-trees [JLRK78, DDGS03], there are several notions of distance-approximability by trees that have been considered in the literature. One variant is obtained

by considering the *stretch* $d_T(u,v)/d_G(u,v)$ over all distinct vertices $u, v \in V$. If the stretch is at most γ, then the tree is called γ-multiplicative tree spanner (see, e.g., [Pri97]). Finding a minimum maximum-stretch tree is NP-hard even for unweighted planar graphs [FK01b], and cannot be approximated by a factor better than $(1+\sqrt{5})/2$ unless P = NP [PR01]. An $O(\log n)$-approximation algorithm has been proposed by Emek and Peleg [EP04b]. The problem of finding the minimum *average*-stretch tree is also NP-hard [JLRK78]. The best know approximation algorithm for the OCT problem (optimum communication spanning tree) has an approximation ratio of $O(\log^2 n \log\log n)$ and is due to Emek, Spielman and Teng [EEST05].

Recently, combinations of additive and multiplicative tree-spanners have been proposed in [EP04a]. Another approach is based on (pseudo-)isometric trees [BCD99, KLM$^+$03], where the minimization is not over spanning trees but over all trees having the same number of vertices as the network in question. Since this loses a direct linkage between the tree and the network, we do not follow this vein.

Bartal [Bar96, Bar98] considered a variant, where a set of dominating tree metrics is used together with a probability distribution such that for every pair of vertices the expected stretch is at most a given constant α. Such a pair is called an α-probabilistic approximation of the original metric. He showed how to find an $O(\log n \log\log n)$-probabilistic approximation in polynomial time. This was improved later on to $O(\log n)$ by Fakcharoenphol, Rao an Talwar [FRT03]. Note that the trees of Bartal and Fakcharoenphol et al. are no spanning trees as they are *not necessarily subgraphs of* G. Alon, Karp, Peleg and West [AKPW95] gave a comparable result with the stronger requirement that trees are spanning trees, which was then significantly improved by the already mentioned approximation algorithm for the problem OCT problem.

Spanning *subgraphs* (not only trees) with certain bounds on distance increases have been intensively studied since the pioneering work of Awerbuch [Awe85], Peleg and Ullman [PU89] and Chew [Che89]. Most of these problems are typically motivated by problems in network design (see, e.g., [PS89, CC95, VRM$^+$97] and the surveys [Soa92, Epp00] for applications). The most general formulation of a spanner problem is due to Liestman and Shermer [LS93]: a spanning subgraph H of G is an $f(x)$-spanner for G if and only if $d_H(u,v) \leq f(d_G(u,v))$ for all $u, v \in V(G)$. As examples, for $f(x) = t + x$ we obtain additive t-spanners, and for $f(x) = t \cdot x$ we obtain multiplicative t-spanners. The computational problem then is to find an $f(x)$-spanner with the minimum number of edges, a problem somewhat dual to ours (as it fixes a bound on the distance increase and tries to minimize the size of the subgraphs, whereas we fix the size of the subgraph and try to minimize the bounds). Combinations of additive and multiplicative spanners – not necessarily trees – have also been considered, e.g., in [BKMP05].

In a series of papers, the hardness of the spanner problems has been exhibited [PS89, Cai94, BH98, Kor01]. The version of this problem that is probably the closest to the problems we consider is to ask for a given graph G and two given parameters m, t if there exists is an additive t-spanner for G with no more than m edges. This problem is NP-complete [LS93]. In the case that $m = n - 1$ is fixed, the problem considered in [LS93] becomes the problem of finding the best possible distance-approximating spanning tree with respect to $\|\cdot\|_{L,\infty}$. However, their NP-completeness proof relies heavily on the number of edges in the instance and hence a translation to an NP-completeness proof for the tree case is not obvious. We resolved this issue here.

The 2-approximation algorithm which we presented is an adaption of the 2-approximation algorithm for the minimum route cost spanning tree problem given by Wong [Won80].

Appendix 3.A Detailed proof of Lemma 3.2

Lemma *(Lemma 3.2)* Let $(\mathcal{E}, \mathcal{V}, k)$ be an instance of 2HS. Then

$$d_{G(\mathcal{E},\mathcal{V})}(a', b') = n \cdot (m + 2).$$

Further, there exists an admissible solution $\mathcal{V}' \subseteq \mathcal{V}$ to $(\mathcal{E}, \mathcal{V}, k)$ if and only if there exists a spanning tree T of $G(\mathcal{E}, \mathcal{V})$ containing all edges in the clause paths such that $d_T(a', b') - d_{G(\mathcal{E},\mathcal{V})}(a', b') \leq k$.

Proof. Let $(\mathcal{E}, \mathcal{V}, k)$ be an instance of 2HS. Clearly, we have that

$$d_{G(\mathcal{E},\mathcal{V})}(a', b') = 2 + n(m + 2).$$

We prove the two directions of the Lemma separately.

($\Rightarrow$) Suppose $\mathcal{V}'$ is an admissible solution to $(\mathcal{E}, \mathcal{V}, k)$, i.e., $\|\mathcal{V}'\| \leq k$. Construct a spanning tree $T_{\mathcal{V}'}$ from $G(\mathcal{E}, \mathcal{V})$ as follows:

- For each set $E_\mu = \{v_\nu, v_\kappa\} \in \mathcal{E}$ do the following: if $v_\nu \in \mathcal{V}'$, then *break the literal path at* l^ν_μ, i.e., remove the edge $\{l^\nu_{\mu-1}, l^\nu_\mu\}$. If $v_\kappa \in \mathcal{V}'$, then *break the literal path at* l^κ_μ, i.e., remove the edge $\{l^\kappa_{\mu-1}, l^\kappa_\mu\}$. Here, we identify $l^\nu_0 = s_\nu$ and $l^\kappa_0 = s_\kappa$. If both, v_ν and v_κ, are in $\mathcal{V}'$, then remove an arbitrary edge from the safety path connecting a'' and l^ν_μ.
- For each $v_\mu \in \mathcal{V}'$ remove the edge $\{l^\mu_m, t_\mu\}$.
- For each $v_\mu \notin \mathcal{V}'$ *break the elongation path*, i.e., remove the edge $\{s_\mu, e^\mu_1\}$.

First note that no edge in any clause path was removed. Second, it was assured that all cycles induced by literal paths and elongation paths are broken by the third and forth construction rules. Those cycles induced by literal paths and clause paths are also broken: by the second and third construction rules, for each set $E_\mu = \{v_\nu, v_\kappa\} \in \mathcal{E}$, at least one of the sets $\{\{l^\nu_{\mu-1}, l^\nu_\mu\}, \{l^\nu_m, t_\nu\}\}$ and $\{\{l^\kappa_{\mu-1}, l^\kappa_\mu\}, \{l^\kappa_m, t_\kappa\}\}$ is completely removed. Therefore either $l^\nu_m u$ or l^κ_μ is no longer reachable from s_ν or t_ν s_κ or t_κ, respectively, via a clause path. Also, an edge from the safety path connecting a'' to l^ν_μ is removed, except if both sets $\{\{l^\nu_{\mu-1}, l^\nu_\mu\}, \{l^\nu_m, t_\nu\}\}$ and $\{\{l^\kappa_{\mu-1}, l^\kappa_\mu\}, \{l^\kappa_m, t_\kappa\}\}$ are removed, an thus neither l^ν_μ nor l^κ_μ are reachable via clause paths from any of the vertices $s_\nu, t_\nu, s_\kappa, t_\kappa$. Now, all possible cycle have been broken and any path from a' to b' in $T_{\mathcal{V}'}$ leads along an interchanging sequence of elongation and literal paths. By means of construction, elongation paths are one edge shorter than literal paths and therefore

$$d_{T_{\mathcal{V}'}}(a', b') = 2 + (n - \|\mathcal{V}'\|)(m + 2) + \|\mathcal{V}'\|(m + 3) \leq d_{G(\mathcal{E},\mathcal{V})}(a', b') + k.$$

This proves the first direction.

($\Leftarrow$) Suppose T is some spanning tree of $G(\mathcal{E}, \mathcal{V})$ containing all edges of all clause paths satisfying $d_T(a', b') - d_{G(\mathcal{E},\mathcal{V})}(a', b') \leq k$. By means of construction, clause paths are such long, that the shortest path cannot lead via any clause path. Hence, the shortest path must lead along a sequence of elongation and literal paths. Now, since any non-broken elongation path has length $m + 2$ and the shortest path between a' and b' in

$G_{(\mathcal{E},\mathcal{V})}$ leads along literal paths, only, it is clear that the shortest path in T leads over *at most* k elongation paths. Let $\mathcal{V}'$ be those elements v_μ, for which the literal path $(s_\mu, l_1^\mu, \ldots, l_m^\mu, t_\mu)$ has been broken in T. This is exactly the set of elements for which the shortest path leads over an elongation path. For all other elements $v_\mu \notin \mathcal{V}'$, the literal path is still intact in T. Assume that for some set $E_\mu = \{v_\nu, v_\kappa\} \in \mathcal{E}$ it holds that $E_\mu \cap \mathcal{V}' = \emptyset$. The corresponding clause path connects l_μ^ν to l_μ^κ and from $\{v_\nu, v_k appa\} \cap \mathcal{V}' = \emptyset$ it follows that the vertex l_μ^ν is connected to the vertex t_ν is connected to the vertex s_κ is connected to the vertex l_μ^κ is connected to l_μ^ν which clearly gives a cycle. This is a contradiction to T being a tree. Thus $\mathcal{V}'$ is a set hitting all elements of $\mathcal{E}$ at least once.

□

Chapter 4

Smoothed analysis of trie height

> Tries are general purpose data structures for text processing and information retrieval. Also, the principle of recursive decomposition based upon successive bits of data items can be found in many other applications
>
> *Flajolet [Fla06]*

4.1 Introduction

4.1.1 Motivation

In this chapter, we consider a fundamental data structure for string processing, i.e., the trie. Experimental findings suggest that worst-case inputs, i.e., sets for which the height of the trie is unbounded, are isolated peaks in the input space and even small deviations from worst-case inputs yield logarithmic trie height. This holds particularly in the case of non-random data: for example, Nilsson and Tikkanen [NT02] have experimentally investigated the height of PATRICIA trees, or path-compressed tries, and other search structures. There, the height of a PATRICIA tree, built over a set of 50,000 unique random uniform strings was 16 on average and at most 20. For non-random data consisting of 19,461 strings from geometric data, of 16,542 ASCII character strings from a book, and of 38,367 strings from Internet routing tables, the height of a path-compressed trie, built over these data sets, was on average 21, 20 and 18, respectively, and at most 30, 41 and 24, respectively; another recent experimental study of an efficient implementation of b-tries is due to Heinz, William and Zobel [HWZ02], and Sinha and Zobel [SZ03]; again, there is a strong evidence that tries perform very well even on biological data like DNA sequences. In this chapter, we try to give an analytical explanation of these findings which requires fewer assumptions with respect to the random model than previous analyses. Clearly, the goal of such an approach is the support the practical findings, i.e., finding that logarithmic trie height holds even under weaker assumptions. To this end, we perform a smoothed analysis of the most important parameter of tries, i.e., its height.

In order to perform a meaningful smoothed analysis of trie height, we present a new *semi-random* model for strings: the set of input strings is chosen in advance by an *oblivious adversary* and then strings are randomly perturbed independently using the same perturbation functions. The oblivious adversary has full information on the parameters of the perturbation function but has no control over the random perturbations and the parameters once the input set is chosen. This model fits into the framework of smoothed analysis. A somewhat stronger model for semi-random sources was considered by Santha and Vazirani in [SV86], though

it was not in the context of tries but in the context of random and quasi-random number generators: there, an *adaptive adversary* had (limited) control over each of the biases in a sequence biased coin flips and full knowledge over the previous history. The class of string perturbation functions which we consider, can be represented by (Mealy-type) probabilistic finite automata. Probabilistic finite automata are a standard tool for modeling unreliable deterministic systems. Also, they provide a compact representation for a very natural class of string perturbation functions, namely, *random edit operations*, which occur in those settings and thus resemble some of the typical random influences that strings are subject to. To the best of our knowledge, we are the first to perform a smoothed analysis of trie parameters.

4.1.2 Our contribution

In summary, we contribute the following:

- We introduce a class of perturbation functions based on *(Mealy-type) probabilistic finite automata (PFAs)*. Our model provides a convenient mathematical framework to represent a class of very natural string perturbation functions, namely edit perturbations, and perturbation functions which are convex combinations of them. Edit perturbations and their convex combinations belong to the class of *star-like perturbation functions* because their representing PFAs are star-like graphs. We show that for a string which results from a star-like perturbation of an non-random input string, the *Rényi's Entropy of second order for sub-strings* does not exist in general. Thus, the semi random string model is not trivially included those previous models for which this limit exists.

- The main result of this chapter is a dichotomous-type of result: Theorem 4.1 gives a set of necessary and sufficient conditions for an *arbitrary* star-like perturbation function such that the smoothed trie height over an arbitrary input set is logarithmic in the number of strings if those conditions are satisfied and is unbounded, otherwise. Moreover, the conditions can easily be verified by looking at the transition probabilities of the representing PFA. A dichotomous-type of result on the height of random tries was also given in [Dev84]. It follows readily from Theorem 4.1 that the smoothed trie height under convex combinations of edit perturbations of arbitrary binary strings is logarithmic if and only if the convex combination does not collapse to deletions.

- We do not only derive the mentioned dichotomous-type of result, but we also investigate the quantitative influence of the "perturbation parameter" on the smoothed trie height (Theorem 4.2): in our model the parameter is characterized by the transition probabilities of the representing PFA.

- We study some extensions of our analysis: first, we derive quantitative lower and upper bounds for read-semi-deterministic perturbation functions. The second extension regards the perturbation model: before, we had assumed that the set of strings that were subject to perturbations was arbitrary. This accounted for the max-operator in the definition of smoothed complexity, or more intuitively speaking for the power of the adversary providing the input. Clearly, for many practical settings, at least some assumptions on the "shape" of the input space can be made, i.e., the power of the adversary can be restricted and some inputs can be excluded at the beginning as being irrelevant. We show that one can trade off these restrictions on the input set against the conditions on the transition probabilities of the perturbing PFA such that again

the smoothed trie height is logarithmic even if these PFA do not yield a logarithmic smoothed trie height over arbitrary input sets (Theorem 4.5). The last extension is an analysis of the smoothed height of b-tries (Theorem 4.6).

4.1.3 Chapter outline

This chapter is organized as follows: in Section 4.2 we first review some of the previous random string models, subject to which the analyses of random tries and related data-structures have been performed; second, we formally introduce smoothed trie height and state our perturbation model, which is based on (Mealy-type) probabilistic finite automata (PFAs). In Section 4.3, we compare the semi-random string model which results from star-like perturbations of non-random inputs to previous purely random string models and show how our model extends upon memory-less random sources. In Section 4.4 we state the main results of the chapter, i.e., Theorems 4.1 and 4.2, which we prove in Section 4.5. We derive the result by a detailed analysis of the computations of star-like PFAs using an approach that relies on rational generating functions and the *weighted words model*(cf. [FS07]). In Section 4.6, we drive upper and lower bounds for the trie height under read-semi-deterministic perturbation functions: the lower bounds are relatively easily derived, as those perturbation functions "simulate" memory-less random sources for specific inputs. Also, we extend our analysis to restricted input sets and b-tries.

4.2 Towards smoothed trie height

4.2.1 Previous studies: the height of random tries

The Model for the analysis of random tries Let $\mathcal{A} = \{\mathtt{a}_1, \ldots, \mathtt{a}_N\}$ be a finite alphabet of cardinality $N \geq 2$ and let $S \subseteq \mathcal{A}^\infty$ be a set of $\|S\| = n$ distinct strings. We assume without loss of generality that $\mathtt{a}_1 = \mathtt{A}$ and that $\mathtt{a}_N = \mathtt{Z}$. The parameters of interest for a trie are:

- the depth $D_S(m)$ of the m-th leaf in the trie,
- the average depth $D_S = \frac{1}{n}\sum_{i=1}^{n} D_S(m)$.
- the external path length $L_S = \sum_{i=1}^{n} D_S(m)$.
- the shortest path $h_S = \min_{1\leq i\leq n} D_S(m)$ and
- the height of the trie $H_S = \max_{1\leq i\leq n} D_S(m)$,

which is undoubtedly the most interesting parameter, because all other parameters can be upper bounded by H_S. For the average-case analysis, let Z be a random variable taking values from $\mathcal{A}$ and let $\{Z_i\}_{i=1}^{\infty}$ be a one-sided infinite sequence of suchlike valued random variables. The sequence $\{Z_i\}_{i=1}^{\infty}$ can be considered an infinite random string over $\mathcal{A}$ generated by a random mechanism, called the *random source*, where the random source determines the probabilities $\mathbf{P}\{Z_i = \mathtt{a}\}$ for all $i \in \mathbb{N}_+$ and $\mathtt{a} \in \mathcal{A}$. Now consider the probability space $(\Omega, \mathfrak{F}, \mu)$ generated by the random sequence $\{Z_i\}_{i=1}^{\infty}$, where $\Omega = \mathcal{A}$ and $\mathfrak{F}$ consists of all finite sets $\mathcal{A}^m$ for $m \in \mathbb{N}_+$ and μ is a probability measure defined on $\mathfrak{F}$ such that for $s \in \mathcal{A}^m$, $\mu(s) = \prod_{i=1}^{m} \mathbf{P}\{Z_i = s[i]\}$. A set of n infinite random strings is then given by n suchlike one-sided infinite sequences $\{Z_i^{(1)}\}_{i=1}^{\infty}, \ldots, \{Z_i^{(n)}\}_{i=1}^{\infty}$. In the analysis of random tries it is usually

assumed that the random strings are independent and identically distributed and thus one can easily define the probability space $(\Omega^{(n)}, \mathfrak{F}^{(n)}, \mu^{(n)})$ consisting of the n-dimensional product space

$$\Omega^{(n)} = \underbrace{\mathcal{A}^\infty \times \ldots \times \mathcal{A}^\infty}_{n \text{ times}},$$

and $\mathfrak{F}^{(n)}$ consisting of all subsets of $S \subseteq \mathcal{A}^m$ for $m \in \mathbb{N}_+$ satisfying $\|S\| = n$, and $\mu^{(n)}$ the probability measure given by the product of the individual probabilities of the n strings, i.e., $\mu^{(n)}(S) = \prod_{i=1}^n \mu(s_i)$ for $S = \{s_1, \ldots, s_n\}$. To analyze its behavior, H_S is viewed a random variable over the above sample space $\Omega^{(n)}$. It is well-known that in the worst-case H_S is unbounded for standard tries and $\Omega(n)$ for PATRICIA trees and other path-compressed tries. By fixing some probability measure μ, one can analyze the expected value of H_S and many more asymptotic properties, e.g., its asymptotic distribution. This has been done for various kinds of random sources: for independent identically distributed strings, the random source completely characterizes the probability space $(\Omega^{(n)}, \mathfrak{F}^{(n)}, \mu^{(n)})$.

The height of random tries The oldest model is the *memory-less random source*, were each symbol corresponds to a possible outcome of a Bernoulli trial. This means, we are given a parameter vector $p = (p_1, \ldots, p_N) \in (0,1)^N$ and for all $i \in \mathbb{N}_+$ and $j \in \{1, \ldots, N\}$ it holds that $\mathbf{P}\{Z_i = \mathtt{a}_j\} = p_j$. For the height $H_{\mathrm{B}}(n)$ of a random trie over a set of n independent strings produced by a memory-less random source with parameter vector $p \in (0,1)^N$ and $Q_2 = (p_1^2 + \ldots + p_N^2)^{-1}$ it holds that

$$H_{\mathrm{B}}(n) \overset{\text{w.h.p.}}{\to} 2 \log_{Q_2} n,$$

where the rate of convergence is $O(n^\varepsilon)$. The convergence of $H_{\mathrm{B}}(n)$ to $2 \log_{Q_2} n$ can also shown to be *asymptotically almost surely*, but we do not discuss this issue here.

Another model for random strings that is discussed intensively in the literature are *Markovian sources*: whereas the successive symbols in a string produced by a memory-less random source are independent, there is a Markovian dependency between those symbols in this model: a string can be considered the outcome of transitions of a finite and ergodic Markov chain with state space $\mathcal{A}$ which has reached its stationary distribution $\pi = (\pi_1, \ldots, \pi_N) \in (0,1)^N$ and has transition matrix $P = (p_{ij})_{1 \le i,j \le N}$: this is, for $i \ge 2$ it holds that $\mathbf{P}\{Z_{i+1} = \mathtt{a}_j | Z_i = \mathtt{a}_k\} = p_{kj}$, where it is implicitly assumed that $\mathbf{P}\{Z_1 = \mathtt{a}_j\} = \pi_j$. For the height $H_{\mathrm{M}}(n)$ of a random trie over a set of n independent strings produced by a stationary and ergodic Markovian source with transition matrix P and $\lambda_{[2]}$ the largest Eigenvalue of the Schur product $P \circ P = (p_{ij}^2)_{1 \le i,j \le r}$ and $Q_2 = (\lambda_{[2]})^{-1}$ it holds that

$$H_{\mathrm{M}}(n) \overset{\text{w.h.p.}}{\to} \log_{Q_2} n,$$

where the rate of convergence is $O(n^\varepsilon)$. Again, asymptotically almost sure convergence can be shown.

Pittel [Pit85, Pit86] considered the growth of different types of random trees under the assumption that the underlying random process $\{Z_i\}_{i=1}^\infty$ satisfies the *mixing property*: the sequence $\{Z_i\}_{i \ge 1}$ satisfies the mixing property, if there exists $n_0 \in \mathbb{N}$ and positive constants c_1, c_2 such that for *all* $1 \le m \le m + n_0 \le n$ and $A \in \mathfrak{F}_1^m$ and $B \in \mathfrak{F}_{m+n_0}^n$ it holds that

$$c_1 \cdot \mathbf{P}\{A\}\,\mathbf{P}\{B\} \le \mathbf{P}\{A \cap B\} \le c_2 \cdot \mathbf{P}\{A\}\,\mathbf{P}\{B\}\,, \tag{4.1}$$

where for $1 \leq k \leq l$, $\mathfrak{F}_k^l$ denotes the σ-field generated by the subsequence $\{Z_i\}_{i=k}^l$. Under this assumption, the following limit, called *Rényi's Entropy of second order* for sub-strings, exists[1]

$$h = \lim_{n\to\infty} \frac{-\ln \sum_{\alpha\in\mathcal{A}^n} \mathbf{P}\{Z_1^n = \alpha\}^2}{2n}, \tag{4.2}$$

where $Z_1^n = (Z_1, \ldots, Z_n)$, and the height $H_{\mathrm{MM}}(n)$ of a random trie built over a set of n independent strings produced by a mixing source satisfies

$$H_{\mathrm{MM}}(n) \overset{\text{w.h.p.}}{\to} \frac{\ln n}{h}.$$

Remark: it can be shown that both the memory-less random source and the Markovian source satisfy the mixing property and furthermore that the Rényi's Entropy of second order for sub-strings exists (cf. [Szp01]). This provides an alternative proof for the height of random tries under those models for random strings.

The following model for random strings is particularly interesting as it presents a dichotomous result and considers strings with unlimited dependency: Devroye [Dev82, Dev84, Dev92b] has introduced the *Density Model*, where each string can be considered the fractional binary expansion of a random variable from $[0, 1)$ and all n random variables are assumed to be independent having identical density. Let $f : [0, 1] \to [0, 1]$ be a density function. For $1 \leq i \leq 2^k$, $k \geq 0$, the dyadic intervals $\mathcal{I}_{k,i}$ of $[0, 1]$ are $[\frac{i-1}{2^k}, \frac{i}{2^k})$. For a random string $\{Z_i\}_{i=1}^\infty$ over the alphabet $\{0, 1\}$ and $i \in \mathbb{N}_+$, let $\tau(Z_1^i) = \sum_{j=1}^i Z_j \cdot 2^{i-j}$. Then for the random string $\{Z_i\}_{i=1}^\infty$ which equals the fractional binary expansion of the random variable x which is drawn according to f it holds that

$$\mathbf{P}\{Z_i = 0\} = \int_{\mathcal{I}_{i,\tau(Z_1^i)}} f(x)\, dx$$

and

$$\mathbf{P}\{Z_i = 1\} = \int_{\mathcal{I}_{i,\tau(Z_1^i)+1}} f(x)\, dx.$$

Particularly, it was shown that the height $H_{\mathrm{DM}}(n)$ of a random trie under the density model satisfies

$$-1 \leq \liminf_{n\to\infty} \mathbf{E}[H_{\mathrm{DM}}(n)] - \frac{\ln \alpha + e}{\ln 2} \leq \limsup_{n\to\infty} \mathbf{E}[H_{\mathrm{DM}}(n)] - \frac{\ln \alpha + e}{\ln 2} \leq 1, \tag{4.3}$$

if $\int f^2(x)dx < \infty$ and is unbounded, otherwise. Here, $\alpha = \frac{n^2 \int_0^1 f^2(x)\, dx}{2}$ and $e = 2.718\ldots$ is Euler's constant.

Another model, that allows for unlimited dependency is are *symbolic dynamical systems* which were introduced by Valleé [Val01] as a very general model for random strings.

[1] originally referred to as h_3 in [Pit85, Pit86] but we drop the subscript

4.2.2 From average-case to smoothed complexity

Now, there are many inputs for which some random mechanism is known that produces strings which approximate the statistical characteristics of the respective inputs. In those cases it seems appropriate for the analysis to assume that the input strings are prefixes of an infinite sequence of symbols generated by such a mechanism. But it is unclear if for every non-random real-world input there is such a mechanism. An analysis of trie parameters which yields essentially the same results without making any assumptions about the existence of a random source that approximates the inputs but which is performed with respect to some weaker assumption seems desirable. This weaker assumption is informally described as follows: we assume that an arbitrary input – worst-case or not, comprised of a set of strings such that some statistics are known or not – is constructed by an *oblivious adversary* but thereafter slightly perturbed randomly. The question which results from this setting is straightforward: how much must the input be perturbed such that the expected height of a trie which is built over the input is logarithmic, and how is the quantitative relation between the perturbation parameters and the expected trie height? To answer these questions, it seems appropriate to perform a smoothed analysis and to model a string by means of a *semi-random string model*, where non-random inputs are subject to slight random perturbations. We initiate this line of research by performing a smoothed analysis of trie height

4.2.3 Smoothed trie height

Having motivated the need of a smoothed analysis of trie parameters, we now turn to the formal definition of the smoothed trie height $H(S, n, X)$. Here, S and X denote the input set and the string perturbation function, respectively, and n is the number of strings that are stored in the trie. For our smoothed analysis, the input set S can either be arbitrary, i.e., the set of all infinite strings $\mathcal{A}^\infty$, or restricted. We consider both variants. In Section 4.4 we will assume that the inputs are unconstrained. This then restricts the perturbation functions that yield logarithmic trie height. In Section 4.6.2, we revise this setting and restrict the input set such that even perturbation functions that are less restricted yield logarithmic trie height. The smoothed trie height is formally defined as follows.

Definition 4.1. *Let $\mathcal{A}$ be a finite alphabet and let $S \subseteq \mathcal{A}^\infty$ be some non-empty set of infinite strings over $\mathcal{A}$. Given a perturbation function $X : \mathcal{A}^{\leq\infty} \to \mathcal{A}^{\leq\infty}$ the* smoothed trie height *for n strings over the set S under the perturbation function X, denoted by $H(S, n, X)$, is defined by*

$$H(S, n, X) =_{\text{def}} \max_{\substack{A \subseteq S \\ \|A\| = n \\ (\forall s,t \in A)\ s \neq t}} \mathbf{E}\left[\max_{s,t \in A} \operatorname{lcp}(X(s), X(t))\right].$$

Note that we assume that strings are perturbed independently. Now, we are ready to introduce our perturbation model.

4.2.4 Perturbations by probabilistic finite automata

In this section, we present our perturbation model which is based on probabilistic finite automata. In order to perform a meaningful smoothed analysis, we propose to consider perturbation functions that locally manipulate strings by random substitutions, insertions and

deletions and by a class of perturbation functions that generalize these operations. Those perturbation functions do resemble random influences which inputs undergo.

4.2.4.1 (Mealy-type) probabilistic finite automata

A probabilistic finite automaton [Paz71, Rab63] is a standard way to model an unreliable deterministic system or a communication channel. We suggest to consider random perturbation functions representable by probabilistic automata. It is not our aim to develop a general theory of automata-based perturbation functions. Instead, we use probabilistic finite automata as a compact, but nevertheless fairly general representation for string perturbation functions. We will define the probabilistic finite automata in a slightly non-standard way by separating input states from output states. This provides an easy way to describe automata computing non-length-respecting input-output relations.

A (Mealy-type) probabilistic finite automaton (PFA) over a finite alphabet $\mathcal{A}$ is a tuple $X = (R, W, \mu_R, \mu_W, \sigma)$ where:

- R is a non-empty, finite set of *input states*.
- W is a non-empty, finite set of *output states*.
- $\mu_R : R \times \mathcal{A} \times (R \cup W) \to [0,1]$ is the *transition probability function for input states* satisfying
$$(\forall q \in R)(\forall \mathtt{a} \in \mathcal{A}) \quad \sum_{p \in R \cup W} \mu_R(q, \mathtt{a}, p) = 1.$$
The semantics of the function μ_R is: if the PFA X is in input state q and the symbol $\mathtt{a}$ is read, move into state p with probability $\mu_R(q, \mathtt{a}, p)$. Note that possibly $\mu_R(q, \mathtt{a}, q) > 0$.
- $\mu_W : W \times \mathcal{A} \times (R \cup W) \to [0,1]$ is the *transition probability function for output states* satisfying
$$(\forall q \in W)(\forall \mathtt{a} \in \mathcal{A}) \quad \sum_{p \in R \cup W} \mu_W(q, \mathtt{a}, p) = 1.$$
The semantics of the function μ_W is: if the PFA X is in output state q, with probability $\mu_W(q, \mathtt{a}, p)$, write the symbol $\mathtt{a}$ and move into state p. Note that possibly $\mu_W(q, \mathtt{a}, q) > 0$.
- $\sigma : R \cup W \to [0,1]$ is the *initial probability distribution* satisfying $\sum_{q \in R \cup W} \sigma(q) = 1$.

String perturbations based on PFAs We will identify with a PFA X over the alphabet $\mathcal{A}$ a *random mapping* $X : \mathcal{A}^{\leq\infty} \to \mathcal{A}^{\leq\infty}$, mapping finite of infinite strings to finite or infinite strings. A *computation of a PFA X on an input symbol* $\mathtt{a} \in \mathcal{A}$ starts in some input state and stops when X moves into an input state, again. The (possibly empty) output of the computation is composed by concatenating all output symbols of transitions leaving output states along which X moved during the computation. A computation of X on an input string $t \in \mathcal{A}^{\leq\infty}$ is composed by the concatenation of the computations on the successive symbols of the string t, where the computation of X on the symbol $t[i+1]$ starts in that input state in which the computation of X on the symbol $t[i]$ stopped. The computation stops when X reaches an input state and there is no more input symbol left to read. If t is infinite the

computation never stops. The output of the computation is composed by concatenating all outputs of the computations on the individual symbols $t[1], t[2], \ldots$. A computation of X is said to have *output length* m if the output has length m and is said to have *input length* l if it has read l symbols of the input.

Drawing PFAs In all figures to follow, states in circles are input states and states in boxes are output states and as usual transitions are only drawn if their probability is strictly positive. Transitions are labeled by a tuples "$\mathtt{a}/x$" where $\mathtt{a} \in \mathcal{A}$ and $0 < x \leq 1$. The semantics is as follows: for a reading transition, $\mathtt{a}/x$ means "if we read symbol $\mathtt{a}$ then we move along the respective transition with probability x"; for a writing transition, "$\mathtt{a}/x$" means "with probability x we move along the respective transition and write $\mathtt{a}$". Rarely, transitions will also by labeled by "$\mathtt{a}^1|\ldots|\mathtt{a}^k/x$", where for $i \in \{1, \ldots, k\}$ it holds that $\mathtt{a}^i \in \mathcal{A}$ and again $0 < x \leq 1$. This kind of labeling is mainly introduced to improve the presentation of the figures. It is an abbreviation for k transitions that are labeled with the tuples $\mathtt{a}^i/x$ for $i \in \{1, \ldots, k\}$.

4.2.4.2 Edit perturbations of binary strings

Edit operations, i.e., *substituting*, *deleting* or *inserting* symbols, are among the most fundamental operations for locally manipulating strings. Therefore, a smoothed analysis with respect to perturbation functions that resemble these operations provides a better understanding of the practical performance of tries. We say that a perturbation function on strings is an *edit perturbation* if it perturbs the input by randomly substituting, inserting or deleting symbols. The PFAs which we introduced above are on the one hand a compact representation for edit perturbations, but on the other hand they also provide a convenient mathematical framework to model and analyze the smoothed trie height under a broad class of natural string perturbation functions. In order to ease the presentation, we assume for the moment that all strings are binary. We give PFAs for all edit perturbations. Finally, we give a PFA representations for the *convex combination* of the individual edit perturbations on binary strings.

Substitutions Let $p \in (0, 1)$ and let SUB_p be the PFA visualized in Figure 4.1. The semantics is as follows: if we are in input state s and read a symbol $a \in \{0, 1\}$ then we move into writing state q_a and move back to state s writing a with probability $1 - p$ and a symbol other than a with probability p. The automaton perturbs an input string by performing random substitutions on it.

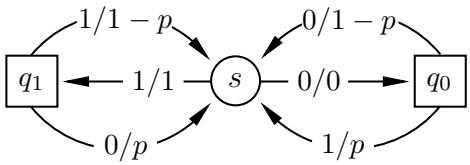


Figure 4.1: Substitution PFA with initial state distribution $\sigma(s) = 1$ and $\sigma(q_0) = \sigma(q_1) = 0$.

Insertions Let $p, q \in (0, 1)$ and let $\mathrm{INS}_{p,q}$ be the automaton depicted in Figure 4.2. It scans an input string $s \in \{0, 1\}^\omega$ and randomly inserts symbols with probability p. The inserted

symbol is 0 with probability q and 1 with probability $1 - q$.

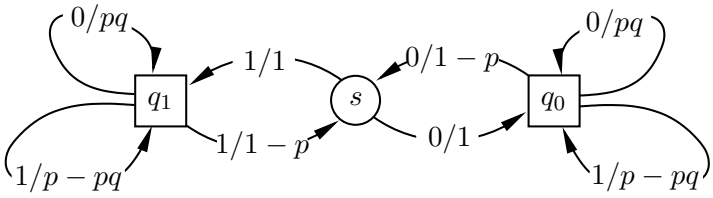


Figure 4.2: Insertion PFA with initial state distribution $\sigma(s) = 1$, $\sigma(q_0) = \sigma(q_1) = 0$.

Deletions. For $p \in (0, 1)$ let DEL_p be the automaton depicted in Figure 4.3 that while scanning an infinite binary string deletes the symbol at a certain position with probability p, independently.

Convex-combinations of edit perturbations Let $p_S, p_I, q_I, p_D \in (0, 1)$ be the respective parameters for the edit perturbations and let $v = (v_S, v_I, v_D) \in [0, 1]^3$ such that $v_S + v_I + v_D = 1$ be the parameter vector for the convex combination. We say that a perturbation function $Y : \{0, 1\}^{\leq\omega} \to \{0, 1\}^{\leq\omega}$ is the *convex combination of the binary edit perturbations*, if Y can be represented by the PFA depicted in Figure 4.4.

4.2.4.3 Star-like perturbation functions

All of the perturbations considered in the last section have in common that there is exactly one input state and that the computations on the individual symbols never move between distinct output states. We now formally define a class of perturbation functions which are characterized by exactly these properties. Since their representation is a directed star graph with multi-edges and loops, where the unique input state is the center vertex, the set of output states is the set of terminal vertices, and the transitions having strictly positive probability gives the set of edges, we call those PFAs and their respective perturbation functions *star-like*.

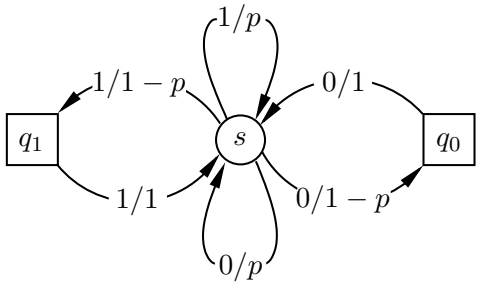


Figure 4.3: Deletion PFA with initial state distribution $\sigma(s) = 1$ and $\sigma(q_0) = \sigma(q_1) = 0$.

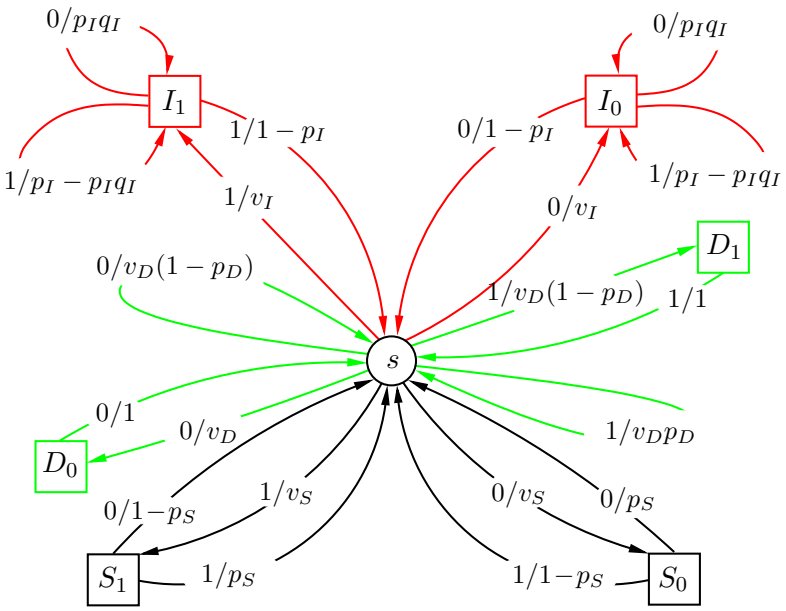


Figure 4.4: PFA Y representing the convex combination of the PFAs $\mathrm{INS}_{p_Iq_I}$, SUB_{p_S} and DEL_{p_D}: the part corresponding to $\mathrm{INS}_{p_Iq_I}$ is drawn red, the part corresponding to DEL_{p_D} is drawn green and the part corresponding to SUB_{p_S} is drawn black.

Definition 4.2 (Star-like automata)**.** *Let $\mathcal{A}$ be finite a alphabet and let $X = (R, W, \mu_R, \mu_W, \sigma)$ be a PFA over $\mathcal{A}$. X is said to be* star-like *if the following hold:*

(1) $\|R\| = 1$*, i.e.,* $R = \{s\}$*.*

(2) The function μ_W is such that

$$(\forall q, q' \in W, q \neq q')(\forall \mathtt{a} \in \mathcal{A}) \quad \mu_W(q, \mathtt{a}, q') = 0,$$

i.e., the graph induced by the vertex set W and the edge set $\{\{q, q'\} : (\exists \mathtt{a} \in \mathcal{A})\ \mu_W(q, \mathtt{a}, q') > 0\}$ consists of a number of connected components each of which is a single vertex.

(3) For all $q \in W$ it holds that $\sum_{\mathtt{a}\in\mathcal{A}} \mu_W(q, \mathtt{a}, q) < 1$, i.e., the probability that X loops at q is strictly less than one.

Further, we consider a strict subclass of the star-like perturbation functions, namely the class of those perturbation functions which are such that for each symbol $\mathtt{a} \in \mathcal{A}$ there is exactly one output state, say $q_{\mathtt{a}}$, that can be reached from s with positive probability when reading $\mathtt{a}$. If additionally to this the perturbation functions are *non-deleting*, i.e., there are no loops at s, then we say that they are *read-deterministic* perturbation functions. Otherwise, i.e., if there are symbols $\mathtt{a}$ which are deleted with positive probability, we say that the perturbation functions are *read-semi-deterministic*.

Definition 4.3 (Read-(semi-)deterministic automata)**.** *Let $\mathcal{A}$ be a finite alphabet and let $X = (\{s\}, W, \mu_R, \mu_W, \sigma)$ be a star-like PFA over $\mathcal{A}$. X is said to be* read-semi-deterministic *if for all $\mathtt{a} \in \mathcal{A}$ there exist a constant $p_{\mathtt{a}} \in [0,1]$ and* exactly one *output state $q_{\mathtt{a}}$ such that $\mu_R(s, \mathtt{a}, s) = p_{\mathtt{a}}$ and $\mu_R(s, \mathtt{a}, q_{\mathtt{a}}) = 1 - p_{\mathtt{a}}$. Further, X is said to be* read-deterministic*, if for all $\mathtt{a} \in \mathcal{A}$, $p_{\mathtt{a}} = 0$, i.e., μ_R has* no loops at s*.*

It is easy to verify that all edit perturbations are star-like perturbation functions and further that the functions INS_{pq} and SUB_p are read-deterministic and the function DEL_p is read-semi-deterministic.

4.3 Comparison to previous random string models

One property that the sequences from most random sources possess is the *mixing property* (see [Bra05] for a survey on the different definitions of mixing), which as we mentioned implies that the Rényi's Entropy of second order, i.e., the limit (4.2), exists. We show that this assumption does not hold in general for sequences which result from the perturbation of a non-random input sequence by means of a star-like perturbation function. Furthermore, we show that the limit (4.2) depends on the input string.

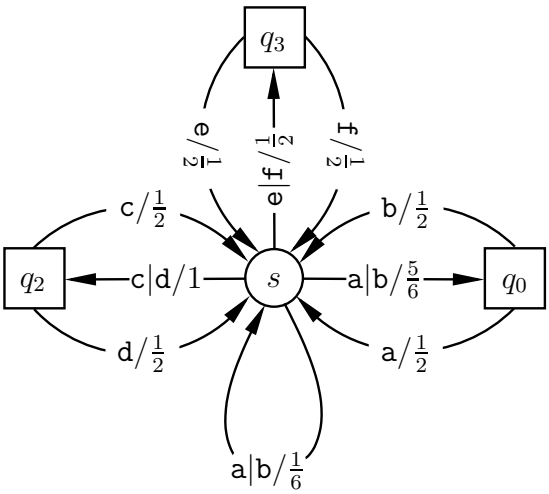


Figure 4.5: Example: a PFA over the alphabet $\mathcal{A} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}\}$ whose respective perturbation function is not mixing.

Proposition 4.1. *For the PFA X over the alphabet $\mathcal{A} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}\}$ depicted in Figure 4.5, there is no $n_0 \in \mathbb{N}_+$ such that there exist two constants $0 < c_1 \leq c_2$ such that for all possible input strings $t \in \mathcal{A}$, the sequence which results from the perturbation of t by means of X is mixing, i.e., satisfies* (4.1)*.*

Proof. The proof is by contradiction: assume that there are $n_0 \in \mathbb{N}_+$ and $0 < c_1 \leq c_2$ such that for all $t \in \mathcal{A}$ it holds that the sequence $\{Z_i\}_{i \geq 1}$ which results from the perturbation of a string $t \in \mathcal{A}^\infty$ by means of X is mixing, i.e., for all $1 \leq m \leq m + n_0 \leq n$ and all $A \in \mathfrak{F}_1^m$ and $B \in \mathfrak{F}_{m+n_0}^n$ it holds that

$$c_1 \cdot \mathbf{P}\{A\}\,\mathbf{P}\{B\} \leq \mathbf{P}\{A \cap B\} \leq c_2 \cdot \mathbf{P}\{A\}\,\mathbf{P}\{B\}\,, \tag{4.4}$$

where for $1 \leq k \leq l$, $\mathfrak{F}_k^l$ denotes the σ-field generated by the subsequence $\{Z_i\}_{i=k}^l$. Now, chose an arbitrary $n \geq n_0$ and define the events $A \in \mathfrak{F}_1^n$,$B \in \mathfrak{F}_{2n}^{3n}$ and $C \in \mathfrak{F}_{2n}^{3n}$ as follows:

- $A = \{s \in \mathcal{A}^\infty \ : \ \sum_{x \in \{\mathtt{c},\mathtt{d}\}} |s[1 \ldots n]|_x = n\}$,
- $B = \{s \in \mathcal{A}^\infty \ : \ \sum_{x \in \{\mathtt{a},\mathtt{b}\}} |s[2n+1 \ldots 3n]|_x = n\}$ and
- $C = \{s \in \mathcal{A}^\infty \ : \ \sum_{x \in \{\mathtt{e},\mathtt{f}\}} |s[2n+1 \ldots 3n]|_x = n\}$.

Let $t \in \mathcal{A}$ satisfying $t[1 \ldots 3n] \in \{\mathtt{a},\mathtt{b}\}^{3n}$ and $t[3n+1 \ldots 5n] \in \{\mathtt{c},\mathtt{d}\}^{2n}$ and $t[5n+1 \ldots] \in \{\mathtt{e},\mathtt{f}\}^\infty$. For A to hold, X must delete the first $3n$ symbols of t, for B to hold, X must not delete any of the first $3n$ symbols, and for C to hold, X must again delete the first $3n$ symbols of t. It can readily be verified that $\mathbf{P}\{A\} = \big(1/6\big)^{3n}$, $\mathbf{P}\{B\} = \big(5/6\big)^{3n}$, $\mathbf{P}\{A \cap B\} = 0$, $\mathbf{P}\{C\} = \big(1/6\big)^{3n}$, and $\mathbf{P}\{A \cap C\} = \mathbf{P}\{A\} = \big(1/6\big)^{3n}$. Therefore, we have

$$\frac{\mathbf{P}\{A \cap B\}}{\mathbf{P}\{A\}\,\mathbf{P}\{B\}} = 0 \quad \text{and} \quad \frac{\mathbf{P}\{A \cap C\}}{\mathbf{P}\{A\}\,\mathbf{P}\{C\}} = 216^n.$$

Thus, the sequence $\{Z_i\}_{i \geq 1}$ is not mixing, i.e., it does not satisfy (4.4) and the proposition follows. □

Thus, it is not possible to choose an n large enough in the analysis of $H(\mathcal{A}^\infty, n, X)$ such that one can consider sub-strings as though they were independent, which is a key ingredient in many average-case analyses. Nevertheless, Theorem 4.1 which we present in the next section implies that that $H(\mathcal{A}^\infty, n, X) \in O(\log n)$. For a semi-read-deterministic PFA $X = (\{s\}, W, \mu_R, \mu_W, \sigma)$ and $\mathtt{a} \in \mathcal{A}$, let

$$Q_\mathtt{a} = (\mu_W(q_\mathtt{a}, \mathtt{A}, q_\mathtt{a}) + \mu_W(q_\mathtt{a}, \mathtt{A}, s), \ldots, \mu_W(q_\mathtt{a}, \mathtt{Z}, q_\mathtt{a}) + \mu_W(q_\mathtt{a}, \mathtt{Z}, s)) \in (0,1)^N.$$

Proposition 4.2. *Let $l \in \mathbb{N}_+$ and let $X = (\{s\}, W, \mu_R, \mu_W, \sigma)$ be a read-semi-deterministic perturbation function over $\mathcal{A}$ in canonical form and let $t \in \mathcal{A}^\infty$ be such that $t[1 \ldots l] \in \{\mathtt{a}\}^l$. For $k \in \{1, \ldots, l\}$, let E_k be the event that $|X(t[1 \ldots l])| \geq k$. Then for all $i \in \{1, \ldots, k\}$ and all $j \in \{1, \ldots, N\}$ it holds that*

$$\mathbf{P}\{X(t)[i] = \mathtt{a}_j \mid E_k\} = Q_\mathtt{a}[j],$$

i.e., conditioned on the event E_k that the computations of input length l have output length at least k, the pair (t, X) induces the same probability measure on $\mathcal{A}^k$ as a memory-less random source with parameter vector $Q_\mathtt{a}$. Further, if $l = +\infty$ then for all $i \in \mathbb{N}_+$ and all $j \in \{1, \ldots, N\}$ it holds that $\mathbf{P}\{X(t)[i] = \mathtt{a}_j\} = Q_\mathtt{a}[j]$.

Proof. Note that t is an infinite string that starts with l repetitions of the symbol $\mathtt{a}$. To proof the proposition, assume that E_k holds and let $q_i \in \{s, q_\mathtt{a}\}$ be the state in which X is *after having written* $X(t)[i-1]$ and let $\mathtt{b} \in \mathcal{A}$. We must consider two cases. First assume that $i = 1$: since X is given in canonical form, we have that $q_1 = s$ and therefore

$$\mathbf{P}\{X(t)[i] = \mathtt{a}_j \mid E_k\} = \mu_W(q_\mathtt{a}, \mathtt{b}, q_\mathtt{a}) + \mu_W(q_\mathtt{a}, \mathtt{b}, s) = Q_\mathtt{a}[j].$$

Now, assume that $i > 1$: in this case we have to consider the two conditional probabilities $\mathbf{P}\{X(t)[i] = \mathtt{a}_j \mid E_k \wedge q_i = q_\mathtt{a}\}$ and $\mathbf{P}\{X(t)[i] = \mathtt{a}_j \mid E_k \wedge q_i = s\}$. The latter probability has already been considered in the case $i = 1$. For the former we have

$$\mathbf{P}\{X(t)[i] = \mathtt{a}_j \mid E_k \wedge q_i = q_\mathtt{a}\} = \mu_W(q_\mathtt{a}, \mathtt{b}, q_\mathtt{a}) + \mu_W(q_\mathtt{a}, \mathtt{b}, s) = Q_\mathtt{a}[j].$$

Together, this implies that for all $i \in \{1, \ldots, k\}$,

$$\mathbf{P}\{X(t)[i] = \mathtt{a}_j \mid E_k\} = Q_{\mathtt{a}}[j] \cdot \mathbf{P}\{q_i = q_{\mathtt{a}}\} + Q_{\mathtt{a}}[j] \cdot \mathbf{P}\{q_i = s\} = Q_{\mathtt{a}}[j].$$

The second claim of the proposition follows from the same reasoning whereat we can omit the conditioning on E_k. This proves the proposition. □

The last proposition has two consequences: it implies that for a sequence which is the output of a perturbation of an arbitrary string by a star-like PFA the limit (4.2) does not exist in general: let X be a semi-read-deterministic PFA such that for two distinct symbols $\mathtt{a}$ and $\mathtt{b}$ it holds that $Q_{\mathtt{a}} \neq Q_{\mathtt{b}}$. Then a standard calculation [Szp01] for the simulated memoryless random source gives that

$$\lim_{n\to\infty} \frac{-\ln(\sum_{\alpha\in\mathcal{A}^n} \mathbf{P}\{X(\mathtt{aa}\ldots)[1\ldots n] = \alpha\}^2)}{2n} = \ln Q_{\mathtt{a}}^{(2)}/2$$

and

$$\lim_{n\to\infty} \frac{-\ln(\sum_{\alpha\in\mathcal{A}^n} \mathbf{P}\{X(\mathtt{bb}\ldots)[1\ldots n] = \alpha\}^2)}{2n} = \ln Q_{\mathtt{b}}^{(2)}/2,$$

where for $i \in \{\mathtt{a}, \mathtt{b}\}$, $Q_i^{(2)} = ((Q_i[1])^2 + \ldots + (Q_i[N])^2)$. Besides this, the last proposition also enables us to give a lower bound on the smoothed trie height under a read-semi-deterministic perturbation function. The respective result is given in Section 4.6.

4.4 Main result: smoothed trie height under star-like perturbation functions

In this section we resent the main results of this chapter.

4.4.1 A dichotomous-type of result

Let X be a star-like perturbation function with representing PFA $X = (\{s\}, W, \mu_R, \mu_W, \sigma)$, where $W = \{q_1, \ldots, q_M\}$. Let $\mathcal{A} = \{\mathtt{A}, \ldots, \mathtt{Z}\}$. To ease the analysis we make the reasonable assumption that the perturbation starts in the input state s with probability one, i.e., that $\sigma(s) = 1$ and for all $i \in \{1, \ldots, M\}$, $\sigma(q_i) = 0$. We say that such a perturbation function is in *canonical form*. We prove the following dichotomous-type of result for star-like perturbation functions over arbitrary input sets in Section 4.5.

Theorem 4.1 (Smoothed trie height for arbitrary input sets)**.** *Let X be a star-like string perturbation function over the finite alphabet $\mathcal{A}$ in canonical form, represented by the PFA $X = (\{s\}, W, \mu_R, \mu_W, \sigma)$ such that for all $\mathtt{a} \in \mathcal{A}$ it holds that $\mu_R(s, \mathtt{a}, s) < 1$. Then the following statements are equivalent.*

(1) $(\forall \mathtt{a}, \mathtt{b} \in \mathcal{A})$ $\mu_R(s, \mathtt{a}, s) + \sum_{q\in W} \mu_R(s, \mathtt{a}, q) \cdot (\mu_W(q, \mathtt{b}, q) + \mu_W(q, \mathtt{b}, s)) < 1$

(2) $H(\mathcal{A}^\infty, n, X) \in O(\log n)$.

Before we discuss the meaning of the above theorem, we note that it directly yields the following corollary concerning the smoothed trie height under convex combinations of edit perturbations of arbitrary binary strings.

Corollary 4.1 (Smoothed trie height under convex combinations of edit perturbations)**.** *Let $p_S, p_I, q_I, p_D \in (0,1)$ and let $v = (v_S, v_I, v_D) \in [0,1]^3$ be such that $v_S + v_I + v_D = 1$ and let Y be the string perturbation function which is computed by the PFA depicted in Figure 4.4. Then $H(\{0,1\}^\infty, n, Y) \in O(\log n)$ if and only if $v_D < 1$. This is, the smoothed trie height under convex combinations of the edit perturbations is logarithmic if and only if the convex combination does not collapse to deletions.*

In general, statement (1) of the theorem gives a set of necessary and sufficient conditions such that the smoothed trie height $H(\mathcal{A}^\infty, n, X)$ is logarithmic in n if those conditions are satisfied and unbounded, otherwise. These conditions are especially appealing, because they can be verified easily and efficiently by looking at the transition probability function of the representing PFA. For general star-like perturbation functions, the verification can be done algorithmically in time $O(N^2 \cdot M)$. For read-semi-deterministic perturbation functions, the verification can be done even faster, i.e., in time $O(N \cdot M)$, which holds particularly because for such a perturbation function one can associate with each input symbol exactly one output state. Note, that the *additional constraint* regarding the deletion probabilities, i.e., that for all $\mathtt{a} \in \mathcal{A}$ it holds that $\mu_R(s, \mathtt{a}, s) < 1$, cannot be dropped: let $\mathtt{a} \in \mathcal{A}$ be such that $\mu_R(s, \mathtt{a}, s) = 1$ and let $t = \mathtt{aaa}\ldots$. Since $X(\mathtt{a}) = \epsilon$ with probability one, it becomes obsolete to speak of smoothed trie height in this particular case.

4.4.2 A quantitative analysis

When performing a smoothed analysis it is usual to quantify the influence of the parameters of the perturbation function on the smoothed complexity of a problem. Let $X(\{s\}, W, \mu_R, \mu_W, \sigma)$ be a star-like PFA over the finite alphabet $\mathcal{A}$ in canonical form. For the sake of exposition we define the following abbreviations: for $j \in \{1, \ldots, M\}$ the *return probability from state* q_j is defined as

$$\eta_j =_{\text{def}} \sum_{i=1}^{N} \mu_W(q_j, \mathtt{a}_i, s).$$

For $\mathtt{a} \in \mathcal{A}$ and $j \in \{1, \ldots, M\}$,

$$\rho_{\mathtt{a}j} =_{\text{def}} \mu_R(s, \mathtt{a}, q_j)$$

and

$$\rho_{\mathtt{a}} =_{\text{def}} \mu_R(s, \mathtt{a}, s).$$

We can give the following quantitative upper bound on the smoothed trie height under arbitrary star-like perturbation functions.

Theorem 4.2. *Let X be a star-like string perturbation function over a finite alphabet $\mathcal{A}$ in canonical form, represented by the PFA $X = (\{s\}, W, \mu_R, \mu_W, \sigma)$, where $W = \{q_1, \ldots, q_M\}$, such that for all $\mathtt{a} \in \mathcal{A}$ it holds that $p_{\mathtt{a}} < 1$ and such that Statement (1) of Theorem 4.1 holds with $\delta_{\mathtt{a}} < 1$ for $\mathtt{a} \in \mathcal{A}$, i.e.,*

$$\delta_{\mathtt{a}} = \max_{\mathtt{b} \in \mathcal{A}} \left(\rho_{\mathtt{a}} + \sum_{j=1}^{M} \rho_{\mathtt{a}j} \cdot (\mu_W(q_j, \mathtt{b}, q_j) + \mu_W(q_j, \mathtt{b}, s)) \right).$$

Let $\tilde{z}$ be the pole of minimum modulus of the function

$$\tilde{\mathcal{Z}}_X(z) = \prod_{\mathtt{a} \in \mathcal{A}} \left(1 - \delta_{\mathtt{a}} \cdot \rho_{\mathtt{a}} - \sum_{j=1}^{M} \frac{\delta_{\mathtt{a}} \cdot \rho_{\mathtt{a}j} \cdot \eta_j \cdot z}{1 - (1 - \eta_j) \cdot z} \right)^{-1}$$

Then for n sufficiently large and all $\varepsilon > 0$ it holds that

$$H(\mathcal{A}^\infty, n, X) \leq 2 \cdot \lceil (1 + \varepsilon) \log_{\tilde{z}} n \rceil + o(1).$$

4.5 The proof: smoothed trie height under star-like perturbation functions

In this section, we prove Theorem 4.1. We start with a result which holds for arbitrary string perturbation functions.

4.5.1 A tail bound for smoothed trie height

Using basic inequalities, we can show that $H(S, n, X)$ grows at most as $2 \log_{1/\gamma} n$, if the *coincidence probability of length m* of two independent perturbations of the same string $s \in S$, i.e., $\mathbf{P}\{\mathrm{lcp}(X(s), X(s)) \geq m\}$, can be bounded from above by γ^m for some $\gamma < 1$. Our approach to upper bound the smoothed trie height can be best compared to the approach taken by Szpankowski in [Szp91] (see Lemma 4) to upper bound the order statistics $\max_{1 \leq i < j \leq n} \mathrm{lcp}(s_i, s_j)$, where $s_1, \ldots, s_n$ are *not necessarily independent* random strings, using generalizations of the results on order statistics of suchlike distributed random variables given in [LR78]. The following lemma holds for arbitrary string perturbation functions.

Lemma 4.1 (Tail-bound for smoothed trie height)**.** *Let $\mathcal{A}$ be a finite alphabet and let $m_0 \in \mathbb{N}$ and $\gamma \in \mathbb{R}$ satisfying $0 < \gamma < 1$. Let $X : \mathcal{A}^{\leq \infty} \to \mathcal{A}^{\leq \infty}$ be a perturbation function and let $S \in \mathcal{A}^\infty$ be a non-empty set of infinite strings. Let $n > \gamma^{-m_0/2}$. If there is a polynomial Π of fixed degree $d \in \mathbb{N}$, such that for all $s \in S$ and all $m \geq m_0$ it holds that the coincidence probability of length m of two independent perturbations of s satisfies*

$$\mathbf{P}\{\mathrm{lcp}(X(s), X(s)) \geq m\} \leq \Pi(m) \cdot \gamma^m,$$

then for all $\varepsilon > 0$ it holds that

$$H(S, n, X) \leq 2 \cdot \lceil (1 + \varepsilon) \log_{1/\gamma} n \rceil + o(1).$$

Proof. Let S be a non-empty set of infinite strings over a finite alphabet $\mathcal{A}$. Let $\varepsilon > 0$ and let $k \in \mathbb{N}_+$ be arbitrary. Then

$$\begin{aligned}
H(S,n,X) &= \max_{\substack{A\subseteq S\\ \|A\|=n}} \mathbf{E}\left[\max_{s,t\in A} \operatorname{lcp}(X(s),X(t))\right] \\
&= \max_{\substack{A\subseteq S\\ \|A\|=n}} \sum_{i=1}^{\infty} \mathbf{P}\left\{\max_{s,t\in A} \operatorname{lcp}(X(s),X(t)) \geq i\right\} \\
&\leq \sum_{i=1}^{\infty} \max_{\substack{A\subseteq S\\ \|A\|=n}} \mathbf{P}\left\{\max_{s,t\in A} \operatorname{lcp}(X(s),X(t)) \geq i\right\} \\
&\leq k + \sum_{i=k+1}^{\infty} \max_{\substack{A\subseteq S\\ \|A\|=n}} \mathbf{P}\left\{\max_{s,t\in A} \operatorname{lcp}(X(s),X(t)) \geq i\right\} \qquad (4.5) \\
&\leq k + n^2 \cdot \sum_{i=k+1}^{\infty} \max_{s,t\in S} \mathbf{P}\{\operatorname{lcp}(X(s),X(t)) \geq i\}\,. \qquad (4.6)
\end{aligned}$$

Inequality (4.6) follows from Boole's Inequality and (4.5) holds, because the in sum of probabilities each addend of the first k addends can by bounded by one. Now we expand each addend of the right-hand side and apply Cauchy's Inequality in its standard from (Inequality (A.1)):

$$\begin{aligned}
\max_{s,t\in S} \mathbf{P}\{\operatorname{lcp}(X(s),X(t)) \geq i\} &= \max_{s,t\in S} \sum_{\alpha\in\mathcal{A}^i} \mathbf{P}\{\alpha \sqsubseteq X(s)\} \cdot \mathbf{P}\{\alpha \sqsubseteq X(t)\} \\
&\leq \max_{s,t\in S} \sqrt{\sum_{\alpha\in\mathcal{A}^i} \mathbf{P}\{\alpha \sqsubseteq X(s)\}^2} \cdot \sqrt{\sum_{\alpha\in\mathcal{A}^i} \mathbf{P}\{\alpha \sqsubseteq X(t)\}^2} \\
&\leq \max_{s\in S} \sum_{\alpha\in\mathcal{A}^i} \mathbf{P}\{\alpha \sqsubseteq X(s)\}^2 \\
&= \max_{s\in S} \mathbf{P}\{\operatorname{lcp}(X(s),X(s)) \geq i\}\,.
\end{aligned}$$

Now, we have that for all $k \in \mathbb{N}_+$ it holds that

$$H(S,n,X) \leq k + n^2 \cdot \sum_{i=k+1}^{\infty} \max_{s\in S} \mathbf{P}\{\operatorname{lcp}(X(s),X(s)) \geq i\}\,.$$

Let $d \in \mathbb{N}_+$ and let Π be a polynomial of degree d such that the assumption of the theorem holds. Set $k = 2 \cdot \lceil (1+\varepsilon)\log_{1/\gamma} n \rceil \geq m_0$. Then

$$\begin{aligned}
H(S,n,X) &\leq k + n^2 \cdot \sum_{i=k+1}^{\infty} \max_{s\in S} \mathbf{P}\{\operatorname{lcp}(X(s),X(s)) \geq i\} \\
&\leq 2 \cdot \lceil (1+\varepsilon)\log_{1/\gamma} n \rceil + \sum_{i=2\cdot\lceil (1+\varepsilon)\log_{1/\gamma} n\rceil+1}^{\infty} \Pi(i) \cdot n^2 \cdot \gamma^i
\end{aligned}$$

It is easy to see that the latter term is in $o(1)$:

$$\sum_{i=2\cdot\lceil (1+\varepsilon)\log_{1/\gamma} n\rceil+1}^{\infty} \Pi(i) \cdot n^2 \cdot \gamma^i = \sum_{i=1}^{\infty} \Pi(2 \cdot \lceil (1+\varepsilon)\log_{1/\gamma} n \rceil + i) \cdot n^2 \cdot \gamma^{2\cdot\lceil (1+\varepsilon)\log_{1/\gamma} n\rceil} \cdot \gamma^i$$

$$\leq \sum_{i=1}^{\infty} \Pi(2 \cdot \lceil (1+\varepsilon) \log_{1/\gamma} n \rceil + i) \cdot n^2 \cdot n^{-2-2\varepsilon} \cdot \gamma^i \in o(1).$$

This proves the lemma. □

4.5.2 Proof of Theorem 4.1

Proof. Let $X = (\{s\}, W, \mu_R, \mu_W, \sigma)$,where $W = \{q_1, \ldots, q_M\}$, be a star-like PFA over the alphabet $\mathcal{A} = \{\mathtt{A}, \ldots, \mathtt{Z}\}$ in canonical form. In order to prove the equivalence of the two statements, we claim that (2) ⇒ (1) and that (1) ⇒ (2). The theorem then follows.

Claim 4.1. *In the setting of Theorem 4.1, it holds that* (2) ⇒ (1).

Proof. We prove the claim by contraposition: to this end assume that (1) *does not hold*, i.e., there are symbols $\mathtt{a}, \mathtt{b} \in \mathcal{A}$ such that

$$\rho_{\mathtt{a}} + \sum_{i=1}^{M} \rho_{\mathtt{a}j} \cdot (\mu_W(q_j, \mathtt{b}, q_j) + \mu_W(q_j, \mathtt{b}, s) = 1.$$

Thus $\mathbf{P}\{\mathtt{b} \sqsubseteq X(\mathtt{a})\} = 1 - \mu_R(s, \mathtt{a}, s)$. Let $t = \mathtt{aa}\ldots$ and let $s = \mathtt{bb}\ldots$. Then X maps t to s with probability one. Therefore, $H(\mathcal{A}^\infty, n, X)$ is unbounded. The claim follows. □

The second claim is less easy to prove: in order to show that (1) ⇒ (2), we prove that (1) is a sufficient condition such that the tail-bound (Lemma 4.1) can be applied. Particularly, we show that under the assumption that (1), there are a polynomial Π of degree at most N, a constant γ satisfying $0 < \gamma < 1$, and a constant $m_0 \in \mathbb{N}_+$ such that for arbitrary $t \in \mathcal{A}^\infty$ and $m \geq m_0$,

$$\mathbf{P}\{\mathrm{lcp}(X(t), X(t)) \geq m\} = \sum_{\alpha \in \mathcal{A}^m} \mathbf{P}\{\alpha \sqsubseteq X(t)\}^2 \leq \Pi(m) \cdot \gamma^m.$$

Definition 4.4 ($\mu_X(\alpha, t)$)**.** *For $\alpha, t \in \mathcal{A}^{<\infty}$, let $\mu_X(\alpha, t)$ be the probability that a computation of X on t having input length $|t|$ has the sting α as prefix.*

Then, for $t \in \mathcal{A}^\infty$ and $\alpha \in \mathcal{A}^{<\infty}$,

$$\mathbf{P}\{\alpha \sqsubseteq X(t)\} = \sum_{l=1}^{\infty} \mu_X(\alpha, t[1\ldots l])$$

and thus for $m \in \mathbb{N}_+$,

$$\mathbf{P}\{\mathrm{lcp}(X(t), X(t)) \geq m\} = \sum_{\alpha \in \mathcal{A}^m} \Big(\sum_{l=1}^{\infty} \mu_X(\alpha, t[1\ldots l]) \Big)^2.$$

Next, we split the right-hand side of the above equation into two suitable parts by an application of Cauchy's Inequality in the form of Inequality (A.2): let $d \in \mathbb{R}_+$ be a constant to be defined in a moment. Then

$$\sum_{\alpha \in \mathcal{A}^m} \big(\sum_{l=1}^{\infty} \mu_X(\alpha, t[1\ldots l]) \big)^2$$

$$= \sum_{\alpha\in\mathcal{A}^m}\Big(\sum_{l=1}^{\lceil d\cdot m\rceil}\mu_X(\alpha,t[1\ldots l]) + \sum_{l=\lceil d\cdot m\rceil+1}^{\infty}\mu_X(\alpha,t[1\ldots l])\Big)^2$$

$$\leq 2\cdot\sum_{\alpha\in\mathcal{A}^m}\Big(\sum_{l=1}^{\lceil d\cdot m\rceil}\mu_X(\alpha,t[1\ldots l])\Big)^2 + 2\cdot\sum_{\alpha\in\mathcal{A}^m}\Big(\sum_{l=\lceil d\cdot m\rceil+1}^{\infty}\mu_X(\alpha,t[1\ldots l])\Big)^2. \quad (4.7)$$

Then, we prove an exponentially decreasing upper bound on each of the two addends in (4.7) under the assumption that (1). To this end, we define for $m\in\mathbb{N}_+$, $d\in\mathbb{R}_+$ and $t\in\mathcal{A}^\infty$

$$\Phi(t,m,d) =_{\text{def}} 2\cdot\sum_{\alpha\in\mathcal{A}^m}\Big(\sum_{l=1}^{\lceil d\cdot m\rceil}\mu_X(\alpha,t[1\ldots l])\Big)^2$$

and

$$\Psi(t,m,d) =_{\text{def}} 2\cdot\sum_{\alpha\in\mathcal{A}^m}\Big(\sum_{l=\lceil d\cdot m\rceil+1}^{\infty}\mu_X(\alpha,t[1\ldots l])\Big)^2.$$

Particularly, we claim.

Claim 4.2. *Assume that* (1) *holds with* $\delta_{\mathtt{a}}<1$ *for* $\mathtt{a}\in\mathcal{A}$*, i.e.,*

$$\delta_{\mathtt{a}} = \max_{\mathtt{b}\in\mathcal{A}}\left(\rho_{\mathtt{a}} + \sum_{j=1}^{M}\rho_{\mathtt{a}j}\cdot(\mu_W(q_j,\mathtt{b},q_j)+\mu_W(q_j,\mathtt{b},s))\right).$$

Let $d\in\mathbb{R}_+$ *be arbitrary but fixed and let* $\tilde{z}_1$ *be the pole of minimum modulus of the function*

$$\tilde{\mathcal{Z}}_X(z) = \prod_{\mathtt{a}\in\mathcal{A}}\left(1-\delta_{\mathtt{a}}\cdot\rho_{\mathtt{a}} - \sum_{j=1}^{M}\frac{\delta_{\mathtt{a}}\cdot\rho_{\mathtt{a}j}\cdot\eta_j\cdot z}{1-(1-\eta_j)\cdot z}\right)^{-1}.$$

Then

(1) $\tilde{z}_1>1$ *and*

(2) there exists a polynomial Π *of degree at most* N *and* $m_0\in\mathbb{N}_+$ *such that for all* $m\geq m_0$*,*

$$\Phi(t,m,d)\leq\Pi(m)\cdot\left(\frac{1}{\tilde{z}_1}\right)^m.$$

Claim 4.3. *For a star-like perturbation function* X *as in the setting of Theorem 4.1, there exist constants* $c,d\in\mathbb{R}_+$ *such that for* $m\in\mathbb{N}_+$*,*

$$\Psi(t,m,d)\leq c\cdot\left(\frac{1}{\tilde{z}_1}\right)^m,$$

where $\tilde{z}_1$ *is defined as in Claim 4.2.*

We postpone the proofs of the two claims to Sections 4.5.5 and 4.5.4. The following claim is a consequence of the the above claims.

Claim 4.4. *Assume that the conditions of Theorem 4.1 (Theorem 4.2) hold. There are constants $c, d \in \mathbb{R}_+$ and $m_0 \in \mathbb{N}_+$ and a polynomial Π of degree at most N such that for every string $t \in \mathcal{A}^\infty$ and all $m \geq m_0$,*

$$\mathbf{P}\{\mathrm{lcp}(X(t), X(t)) \geq m\} \leq \Psi(t, m, d) + \Phi(t, m, d) \leq (c + \Pi(m)) \cdot \left(\frac{1}{\tilde{z}_1}\right)^m .$$

Thus we may apply the tail-bound (Lemma 4.1). Together this shows the sought-after claim.

Claim 4.5. *In the setting of Theorem 4.1 it holds that* (1) $\Rightarrow$ (2).

This proves Theorem 4.1. □

Remark: Claim 4.4 together with Lemma 4.1 directly implies Theorem 4.2.

All that is left to prove the two Theorems, is to prove Claims 4.2 and 4.3.

4.5.3 Prerequisites: computations of star-like PFAs

Before we prove the two claims, we show how to express the term $\sum_{\alpha \in \mathcal{A}^m} \mu_X(\alpha, t[1 \ldots l])$, i.e., the probability that a computation of input length l on the prefix of t has output length at least m, subject to the transition probabilities of X.

Lemma 4.2. *Let $X = (\{s\}, W, \mu_R, \mu_W, \sigma)$,where $W = \{q_1, \ldots, q_M\}$, be a star-like PFA over the finite alphabet $\mathcal{A}$ in canonical form and let $t \in \mathcal{A}^\infty$ and $l \in \mathbb{N}_+$. Let $f : \mathbb{N} \to \{0, 1\}$ the following defined function: for $x \in \mathbb{N}$,*

$$f(x) =_{\mathrm{def}} \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{otherwise.} \end{cases}$$

The function f is used to indicate deleted symbols in the computations of X. Then

$$\sum_{\alpha \in \mathcal{A}^m} \mu_X(\alpha, t[1 \ldots l]) \leq 1/\tilde{\eta} \cdot \sum_{m_1 + \ldots + m_l = m} \prod_{i=1}^{l} \Big(f(m_i) \cdot \rho_{t[i]} + (1 - f(m_i)) \cdot \sum_{j=1}^{M} \rho_{t[i]j} \eta_j (1 - \eta_j)^{m_i - 1} \Big),$$

where $\tilde{\eta} = \min_{1 \leq j \leq M} \eta_j$ denotes the minimum return probability.

Proof. Remember, the term which we seek to bound is the probability that the computation of X on t of input length l has output length at least m. Since X is star-like and given in canonical form, each computation of X on t starts in the input state and then moves into some output state, from which it writes the output, before it moves into the input state again, where it reads the next symbol of the input and continues the computations as described above. Thus, each computation can be decomposed into the computations on the successive individual symbols of t. For a computation of input length l and output length m, there are $\binom{m+l-1}{l-1}$ possibilities to concatenate l such computations on the individual symbols such that they give a computation of output length m: this equals the number of decompositions of m into l non-negative addends, where addends might be equal to zero because computations might have output length equal to zero. The computations on the first $l - 1$ input symbols *must* return into the input state, whereas the computation on the l-th and last input symbol

may either loop at its output state or return back into the input state after having written the m-th and last symbol of the output.

Now, consider a fixed decomposition $m_1 + \ldots + m_l = m$ into possibly empty computations. For $i \in \{0, \ldots, l\}$, if $m_i = 0$ then the probability that the computation of X on the symbol $t[i]$ has output length zero is

$$\mathbf{P}\{X(t[i]) = \epsilon\} = \rho_{t[i]}. \tag{4.8}$$

For $i \in \{1, \ldots, l-1\}$, if $m_i > 0$ then the probability that the computation of X on the symbol $t[i]$ has output length *exactly* m_i is equal to

$$\sum_{\alpha \in \mathcal{A}^{m_i}} \mathbf{P}\{X(t[i]) = \alpha\} = \sum_{j=1}^{M} \rho_{t[i]j} \cdot \eta_j \cdot (1 - \eta_j)^{m_i - 1} \tag{4.9}$$

and the probability that the computation of X on the symbol $t[l]$ has output length *at least* $m_l > 0$ is equal to

$$\sum_{\alpha \in \mathcal{A}^{m_l}} \mathbf{P}\{\alpha \sqsubseteq X(t[l])\} = \sum_{j=1}^{M} \rho_{t[l]j} \cdot (1 - \eta_j)^{m_l - 1}. \tag{4.10}$$

Let $\tilde{\eta} = \min_{1 \le j \le M} \eta_j$ be the minimum return probability. The term (4.10) can be bounded as

$$\sum_{j=1}^{M} \rho_{t[i]j} \cdot (1 - \eta_j)^{m_l - 1} \le 1/\tilde{\eta} \cdot \sum_{j=1}^{M} \rho_{t[i]j} \cdot \eta_j \cdot (1 - \eta_j)^{m_l - 1}. \tag{4.11}$$

Now, we use the indicator function f to choose for $i \in \{1, \ldots, l\}$, the i-th (product-)addend in the product which bounds the probability that the computation of X on t of input length l that can be decomposed as $m_1 + \ldots + m_l = m$ has length at least m: if $m_i = 0$ then (4.8) is the appropriate addend; if $m_i > 0$ and $i \in \{1, \ldots, l-1\}$ then (4.9) is the appropriate addend; finally, if $i = l$ and $m_l > 0$ then (4.11) is the appropriate addend: thus,

$$1/\tilde{\eta} \cdot \prod_{i=1}^{l} \Big(f(m_i) \cdot \rho_{t[i]} + (1 - f(m_i)) \cdot \sum_{j=1}^{M} \rho_{t[i]j} \cdot \eta_j \cdot (1 - \eta_j)^{m_i - 1} \Big)$$

is the desired upper bound on the respective probability. Summing over all possible decompositions, we get

$$\sum_{\alpha \in \mathcal{A}^m} \mu_X(\alpha, t[1 \ldots l]) \le 1/\tilde{\eta} \cdot \sum_{m_1 + \ldots + m_l = m} \prod_{i=1}^{l} \Big(f(m_i) \cdot \rho_{t[i]} + (1 - f(m_i)) \cdot \sum_{j=1}^{M} \rho_{t[i]j} \eta_j (1 - \eta_j)^{m_i - 1} \Big),$$

which proves the lemma. □

4.5.4 Bounding $\Phi(t, m, d)$

In order to prove an exponentially decreasing upper bound on

$$\Phi(t, m, d) = 2 \cdot \sum_{\alpha \in \mathcal{A}^m} \Big(\sum_{l=1}^{\lceil dm \rceil} \mu_X(\alpha, t[1 \ldots l]) \Big)^2$$

for arbitrary but fixed $d \in \mathbb{R}_+$ an thereby prove Claim 4.2, we first apply Cauchy's inequality and then bound by counting over all possible $l \in \mathbb{N}_+$:

$$\begin{aligned} 2 \cdot \sum_{\alpha \in \mathcal{A}^m} \Big(\sum_{l=1}^{\lceil dm \rceil} \mu_X(\alpha, t[1 \ldots l]) \Big)^2 &\leq 2\lceil dm \rceil \cdot \sum_{\alpha \in \mathcal{A}^m} \sum_{l=1}^{\lceil d \cdot m \rceil} \mu_X(\alpha, t[1 \ldots l])^2 \\ &\leq 2\lceil dm \rceil \cdot \sum_{l=1}^{\infty} \sum_{\alpha \in \mathcal{A}^m} \mu_X(\alpha, t[1 \ldots l])^2 \end{aligned} \tag{4.12}$$

A crucial lemma The next Proposition follows from the definition of $\mu_X(\alpha, t[k \ldots l])$ and the fact that for a star-like PFA the computations can be decomposed into the computations on the individual input symbols

Proposition 4.3. *Let $j, k, l \in \mathbb{N}+$ satisfying $k \leq j < l$ and $\alpha \in \mathcal{A}^m$. Then*

$$\mu_X(\alpha, t[k \ldots l]) = \sum_{i=0}^{m} \mathbf{P}\{X(t[k \ldots j]) = \alpha[0 \ldots i]\} \cdot \mu_X(\alpha[i{+}1 \ldots m], t[j{+}1 \ldots l]) \,.$$

Lemma 4.3. *Let $X = (\{s\}, W, \mu_R, \mu_W, \sigma)$ be a star-like PFA in canonical form over the finite alphabet $\mathcal{A}$. For $\mathtt{a} \in \mathcal{A}$ let*

$$\delta_{\mathtt{a}} = \max_{\mathtt{b} \in \mathcal{A}} \left(\rho_{\mathtt{a}} + \sum_{j=1}^{M} \rho_{\mathtt{a}j} \cdot (\mu_W(q_j, \mathtt{b}, q_j) + \mu_W(q_j, \mathtt{b}, s)) \right) \tag{4.13}$$

and let $\tilde{\delta} = \min_{\mathtt{a} \in \mathcal{A}} \delta_{\mathtt{a}}$. Then for all infinite strings $t \in \mathcal{A}^\infty$ and all $k, l, m \in \mathbb{N}_+$ it holds that

$$\sum_{\alpha \in \mathcal{A}^m} \mu(\alpha, t[k \ldots l])^2 \leq (1/\tilde{\delta}) \cdot \prod_{j=k}^{l} \delta_{t[j]} \cdot \sum_{\alpha \in \mathcal{A}^m} \mu(\alpha, t[k \ldots l]) \,. \tag{4.14}$$

Proof of Lemma 4.3. Let X be a star-like PFA and let $t \in \mathcal{A}^\infty$ be an arbitrary input string and let $\tilde{\delta} = \min_{\mathtt{a} \in \mathcal{A}} \delta_{\mathtt{a}}$. For $\mathtt{a} \in \mathcal{A}$ and $\alpha \in \mathcal{A}^{\leq \infty}$ satisfying $|\alpha| \geq 1$ we have

$$\begin{aligned} \sum_{i=0}^{|\alpha|} \mathbf{P}\{X(\mathtt{a}) = \alpha[0 \ldots i]\} &= \mathbf{P}\{X(\mathtt{a}) = \varepsilon\} + \mathbf{P}\{X(\mathtt{a}) = \alpha[1]\} + \ldots \\ &\leq \mathbf{P}\{X(\mathtt{a}) = \varepsilon\} + \mathbf{P}\{\alpha[1] \sqsubseteq X(\mathtt{a})\} \\ &\leq \delta_{\mathtt{a}}. \end{aligned} \tag{4.15}$$

We prove the lemma by induction on the length $\ell = l - k + 1$ of the part of t which is read. First note that for $l < k$ the left and the right hand side of Inequality (4.14) are equal to zero. This holds particularly, because X is given in canonical form. Therefore we may without loss of generality assume that $l \geq k$ holds.

Induction basis: for $\ell = 1$ it holds that

$$\sum_{\alpha \in \mathcal{A}^m} \mu_X(\alpha, t[l])^2 \leq (\delta_{t[l]}/\tilde{\delta}) \cdot \sum_{\alpha \in \mathcal{A}^m} \mu_X(\alpha, t[l]) \,,$$

because probabilities are less than one.
Induction step: assume that (4.14) holds for $l-k \leq \ell - 2$. By Proposition 4.3 we get

$$\sum_{\alpha\in\mathcal{A}^m} \mu_X(\alpha, t[k\ldots l])^2 = \sum_{\alpha\in\mathcal{A}^m} \left(\sum_{i=0}^{m} \mathbf{P}\{X(t[k])=\alpha[0\ldots i]\} \cdot \mu_X(\alpha[i+1\ldots m], t[k+1\ldots l])\right)^2$$

We apply Jensen's Inequality: let $x_0,\ldots,x_m,y_0,\ldots,y_m \in \mathbb{R}^+$. Then

$$\Big(\sum_{i=1}^{m} x_i y_i\Big)^2 = \Big(\sum_{i=0}^{m} x_i\Big)^2 \cdot \Big(\frac{\sum_{i=0}^{m} x_i y_i}{\sum_{i=0}^{m} x_i}\Big)^2 \leq \Big(\sum_{i=0}^{m} x_i\Big) \cdot \Big(\sum_{i=0}^{m} x_i y_i^2\Big) \tag{4.16}$$

For $i \in \{0,\ldots,m\}$ we set

$$x_i = \mathbf{P}\{X(t[k]) = \alpha[0\ldots i]\}$$

and

$$y_i = \mu_X(\alpha[i+1\ldots m], t[k+1\ldots l])$$

in Inequality (4.16). Additionally we know from Inequality (4.15) that

$$\sum_{i=0}^{m} x_i = \sum_{i=0}^{m} \mathbf{P}\{X(t[k])=\alpha[0\ldots i]\} \leq \delta_{t[k]}.$$

Together, we get that

$$\Big(\sum_{i=0}^{m} x_i y_i\Big)^2 \leq \Big(\sum_{i=0}^{m} x_i\Big) \cdot \Big(\sum_{i=0}^{m} x_i y_i^2\Big) \leq \delta_{t[k]} \cdot \Big(\sum_{i=0}^{m} x_i y_i^2\Big)$$

which after re-translating gives

$$\sum_{\alpha\in\mathcal{A}^m} \left(\sum_{i=0}^{m} \mathbf{P}\{X(t[k])=\alpha[0\ldots i]\} \cdot \mu_X(\alpha[i+1\ldots m], t[k+1\ldots l])\right)^2$$
$$\leq \sum_{\alpha\in\mathcal{A}^m} \delta_{t[k]} \cdot \sum_{i=0}^{m} \mathbf{P}\{X(t[k])=\alpha[0\ldots i]\} \cdot \mu_X(\alpha[i+1\ldots m], t[k+1\ldots l])^2 .$$

Now we proceed as follows:

$$\begin{aligned}
&\sum_{\alpha\in\mathcal{A}^m} \delta_{t[k]} \cdot \sum_{i=0}^{m} \mathbf{P}\{X(t[k])=\alpha[0\ldots i]\} \cdot \mu_X(\alpha[i+1\ldots m], t[k+1\ldots l])^2 \\
&= \delta_{t[k]} \cdot \sum_{i=0}^{m} \left(\sum_{\alpha_1\in\mathcal{A}^i} \mathbf{P}\{X(t[k])=\alpha_1\}\right) \cdot \left(\sum_{\alpha_2\in\mathcal{A}^{m-i}} \mu_X(\alpha_2, t[i+1\ldots l])^2\right) \\
&\leq \delta_{t[k]} \cdot \sum_{i=0}^{m} \left(\sum_{\alpha_1\in\mathcal{A}^i} \mathbf{P}\{X(t[k])=\alpha_1\}\right) \cdot \left((1/\tilde{\delta}) \cdot \prod_{j=k+1}^{l} \delta_{t[j]} \cdot \sum_{\alpha_2\in\mathcal{A}^{m-i}} \mu_X(\alpha_2, t[k+1\ldots l])\right) \\
&= (1/\tilde{\delta}) \cdot \prod_{j=k}^{l} \delta_{t[j]} \cdot \sum_{i=0}^{m} \left(\sum_{\alpha_1\in\mathcal{A}^i} \mathbf{P}\{X(t[k])=\alpha_1\}\right) \cdot \left(\sum_{\alpha_2\in\mathcal{A}^{m-i}} \mu_X(\alpha_2, t[k+1\ldots l])\right)
\end{aligned}$$

$$= \quad (1/\tilde{\delta}) \cdot \prod_{j=k}^{l} \delta_{t[j]} \cdot \sum_{\alpha \in \mathcal{A}^m} \mu_X(\alpha, t[k \ldots l]) \,,$$

where the inequality follows from the induction hypothesis. Altogether, we have shown

$$\sum_{\alpha \in \mathcal{A}^m} \mu_X(\alpha, t[k \ldots l])^2 \leq (1/\tilde{\delta}) \cdot \prod_{j=k}^{l} \delta_{t[j]} \cdot \sum_{\alpha \in \mathcal{A}^m} \mu_X(\alpha, t[k \ldots l]) \,.$$

This proves the lemma. □

Remark 4.1. *Lemma 4.3 holds irrespectively of the conditions given by Statement* (1) *of Theorem 4.1.*

By the above lemma and Lemma 4.2, we can bound the inner sum of (4.12) as follows:

$$\begin{aligned}
&\sum_{\alpha \in \mathcal{A}^m} \mu_X(\alpha, t[1 \ldots l])^2 \\
&\leq \quad (1/\tilde{\delta}) \cdot \prod_{j=1}^{l} \delta_{t[j]} \cdot \sum_{\alpha \in \mathcal{A}^m} \mu_X(\alpha, t[1 \ldots l]) && (4.17) \\
&\leq \quad 1/\tilde{\eta} \cdot \sum_{m_1 + \ldots + m_l = m} \prod_{i=1}^{l} \Big(f(m_i) \cdot \delta_{t[i]} \rho_{t[i]} + (1 - f(m_i)) \cdot \sum_{j=1}^{M} \delta_{t[i]} \rho_{t[i]j} \eta_j (1 - \eta_j)^{m_i - 1} \Big), && (4.18)
\end{aligned}$$

where $\tilde{\eta} = \min_{1 \leq j \leq M} \eta_j$ and $f : \mathbb{N} \to \{0, 1\}$ is the indicator function for deleted letters in the computation of X.

Valid words and expressions Call each non-zero addend in the above sum (4.18) a *valid expression* of X on t. A valid expression is said to be of input length l if it corresponds to a set of computations of input length l and is said to be of output length m if its corresponding set of computations has output length m. Each valid expression is a product over the set of variables $\{\delta_{\mathtt{A}}, \ldots, \delta_{\mathtt{Z}}, \rho_{\mathtt{A}}, \ldots, \rho_{\mathtt{Z}}, \rho_{\mathtt{A}1}, \ldots, \rho_{\mathtt{Z}M}, \eta_1, \ldots, \eta_M\}$. The products have a regular structure which we exploit in order to bound the term (4.18): to this end, let $\mathcal{B}$ be the following alphabet, where we interpret variables as letters (we intentionally use the term 'letter' for an element of the alphabet and 'word' for a sequence of letters in order to avoid confusion)

$$\begin{aligned}
\mathcal{B} \quad =_{\text{def}} \quad & \{\delta_{\mathtt{A}}, \ldots, \delta_{\mathtt{Z}}\} \\
\cup \quad & \{\eta_j \ : \ j \in \{1, \ldots, M\}\} \cup \{(1 - \eta_j) \ : \ j \in \{1, \ldots, M\}\} \\
\cup \quad & \{\rho_{\mathtt{a}j} \ : \ j \in \{1, \ldots, M\} \text{ and } \mathtt{a} \in \mathcal{A}\} \cup \{\rho_{\mathtt{a}} \ : \ \mathtt{a} \in \mathcal{A}\}.
\end{aligned}$$

Each valid expression in (4.18) can readily be identified with a word w over the alphabet $\mathcal{B}$: e.g., the word

$$\delta_{\mathtt{a}} \ \rho_{\mathtt{a}} \ \delta_{\mathtt{b}} \ \rho_{\mathtt{b}} \ \delta_{\mathtt{a}} \ \rho_{\mathtt{a}2} \ (1 - \eta_2) \ (1 - \eta_2) \ (1 - \eta_2) \ \eta_2$$

corresponds to a valid expression of X on $t = \mathtt{aba}\ldots$ of input length 3 and output length 4 and thus to a set of computations of X on t, where each computations deletes the first two letters and then moves into state q_2 after having read the letter $\mathtt{a}$, whereupon it loops three times at q and then moves back to the input state again. Clearly, *not all words* over $\mathcal{B}$ are

valid expression of X on t. Call a word a *valid word* if it is. Let $\mathcal{W}_X^{(t,l,m)}$ be the set of all valid words over $\mathcal{B}$ that correspond to a valid expression of X on t that has the respective input and output lengths.

$$\mathcal{W}_X^{(t,l,m)} =_{\text{def}} \{w \in \mathcal{B}^{<\infty} \ : \ w \text{ is a valid word having input length } l \text{ and output length } m\},$$

To see that the words in the above language have indeed a regular structure, we first consider the following *regular language*

$$\begin{aligned}
\mathcal{W}_X^{(t,l)} = \{ \quad w \quad & \in \mathcal{B}^{\leq\infty} \\
w \quad & := W_1 \, W_2 \, \ldots \, W_l \\
W_1 \quad & := \delta_{t[1]} \, \rho_{t[1]} \ \mid \delta_{t[1]} \, \rho_{t[1]1} \, \eta_1 \, (1-\eta_1)^* \mid \ \ldots \ \mid \delta_{t[1]} \, \rho_{t[1]M} \, \eta_M \, (1-\eta_M)^* \\
& \vdots \\
W_l \quad & := \delta_{t[l]} \, \rho_{t[l]} \ \mid \delta_{t[l]} \, \rho_{t[l]1} \, \eta_1 \, (1-\eta_1)^* \mid \ \ldots \ \mid \delta_{t[l]} \, \rho_{t[l]M} \, \eta_M \, (1-\eta_M)^* \}
\end{aligned}$$

Here, we use $W_1, \ldots, W_l$ as *placeholders* for the above defined regular expressions. Now, it holds that

$$\mathcal{W}_X^{(t,l,m)} = \{w \in \mathcal{W}_X^{(t,l)} \ : \ \sum_{j=1}^{M} (|w|_{\eta_j} + |w|_{(1-\eta_j)}) = m\}$$

and further that

$$(\forall w \in \mathcal{W}_X^{(t,l)}) \ \sum_{\mathtt{a} \in \mathcal{A}} |w|_{\delta_\mathtt{a}} = l.$$

In order to evaluate the term (4.18) using the framework of valid words, we follow the *weighted words model*: we define the weight $\pi(w)$ of a word $w \in \mathcal{B}^{<\infty}$ as the product of all letters which constitute w, where the multiplicity of a letter in the product equals the number of times it occurs in the word w:

$$\pi(w) =_{\text{def}} \sum_{x \in \mathcal{B}} x^{|w|_x}.$$

E.g., for

$$w' = \delta_\mathtt{a} \, \rho_\mathtt{a} \, \delta_\mathtt{b} \, \rho_\mathtt{b} \, \delta_\mathtt{a} \, \rho_{\mathtt{a}2} \, (1-\eta_2) \, (1-\eta_2) \, (1-\eta_2) \, \eta_2$$

the example word from above we have that

$$\pi(w') = \delta_\mathtt{a}^2 \cdot \delta_\mathtt{b} \cdot \rho_\mathtt{a} \cdot \rho_\mathtt{b} \cdot \rho_{\mathtt{a}2} \cdot (1-\eta_2)^3 \cdot \eta_2,$$

as $|w'|_{(1-\eta_2)} = 3$ and $|w'|_\mathtt{a} = |w'|_{\rho_\mathtt{a}} = |w'|_\mathtt{b} = |w'|_{\rho_\mathtt{b}} = |w'|_{\rho_{\mathtt{a}2}} = |w'|_{\eta_2} = 1$. Also, the weight of a set is defined as the weight of all elements in the set. The weight of a valid word equals the value of its corresponding valid expression. Thus

$$\pi\left(\mathcal{W}_X^{(t,l,m)}\right) = \sum_{m_1+\ldots+m_l=m} \prod_{i=1}^{l} \left(f(m_i) \cdot \delta_{t[i]} \rho_{t[i]} + (1-f(m_i)) \cdot \sum_{j=1}^{M} \delta_{t[i]} \rho_{t[i]j} \eta_j (1-\eta_j)^{m_i-1} \right).$$

Note that for $l, l' \in \mathbb{N}_+$ satisfying $l \neq l'$ it holds that $\mathcal{W}_X^{(t,l,m)} \cap \mathcal{W}_X^{(t,l',m)} = \emptyset$. The consequence of the preceding calculations is captured in the next proposition.

Proposition 4.4.

$$\sum_{l=1}^{\infty}\prod_{i=1}^{l}\delta_{t[i]}\cdot\sum_{\alpha\in\mathcal{A}^m}\mu_X(\alpha,t[1\ldots l])\leq(1/\tilde{\eta})\cdot\sum_{l=1}^{\infty}\pi\left(\mathcal{W}_X^{(t,l,m)}\right).$$

Proof.

$$\begin{aligned}
&\sum_{l=1}^{\infty}\prod_{i=1}^{l}\delta_{t[i]}\cdot\sum_{\alpha\in\mathcal{A}^m}\mu_X(\alpha,t[1\ldots l])\\
&\leq\ (1/\tilde{\eta})\cdot\sum_{l=1}^{\infty}\left(\sum_{m_1+\ldots+m_l=m}\prod_{i=1}^{l}\left(f(m_i)\cdot\delta_{t[i]}\rho_{t[i]}+(1-f(m_i))\cdot\sum_{j=1}^{M}\delta_{t[i]}\rho_{t[i]j}\eta_j(1-\eta_j)^{m_i-1}\right)\right)\\
&=\ (1/\tilde{\eta})\cdot\sum_{l=1}^{\infty}\pi\left(\mathcal{W}_X^{(t,l,m)}\right).
\end{aligned}$$

This proves the proposition. □

Reordering valid words In order to evaluate the sum over the weights of all valid words of output length m over all input lengths, i.e., $\sum_{l=1}^{\infty}\pi\left(\mathcal{W}_X^{(t,l,m)}\right)$, we first construct a family of structurally simpler sets $\mathcal{Y}_X^{(t,1,m)},\mathcal{Y}_X^{(t,2,m)},\ldots$ such that

- for $l,l'\in\mathbb{N}_+$ satisfying $l\neq l'$ it holds that $\mathcal{Y}_X^{(t,l,m)}\cap\mathcal{Y}_X^{(t,l',m)}=\emptyset$ and
- for each $l\in\mathbb{N}_+$ there exists a *weight preserving bijection* g between $\mathcal{W}_X^{(t,l,m)}$ and $\mathcal{Y}_X^{(t,l,m)}$.

Then the sum over all weights of all sets $\mathcal{Y}_X^{(t,l,m)}$ over all l is equal to the sum over all weights of all sets $\mathcal{W}_X^{(t,l,m)}$ over all l. To this end, let for $l\in\mathbb{N}_+$ and $i\in\{1,\ldots,N\}$,

$$l_i=|t[1\ldots l]|_{\mathtt{a}_i}$$

be the number of occurrences of the symbol $\mathtt{a}_i$ in the prefix $t[1\ldots l]$. The following regular language is derived from $\mathcal{W}_X^{(t,l)}$ by reordering the constituent sub-words.

$$\begin{aligned}
\mathcal{Y}_X^{(t,l)}=\{\quad w\quad &\in\mathcal{B}^{\leq\infty}\\
w\quad &:=Y_1\ Y_2\ \ldots\ Y_l\\
Y_1\quad &:=\delta_{\mathtt{a}_1}\ \rho_{\mathtt{a}_1}\ \mid\delta_{\mathtt{a}_1}\ \rho_{\mathtt{a}_1 1}\ \eta_1\ (1-\eta_1)^*\mid\ \ldots\ \mid\delta_{\mathtt{a}_1}\ \rho_{\mathtt{a}_1 M}\ \eta_M\ (1-\eta_M)^*\\
&\vdots\\
Y_{l_1}\quad &:=\delta_{\mathtt{a}_1}\ \rho_{\mathtt{a}_1}\ \mid\delta_{\mathtt{a}_1}\ \rho_{\mathtt{a}_1 1}\ \eta_1\ (1-\eta_1)^*\mid\ \ldots\ \mid\delta_{\mathtt{a}_1}\ \rho_{\mathtt{a}_1 M}\ \eta_M\ (1-\eta_M)^*\\
Y_{l_1+1}\quad &:=\delta_{\mathtt{a}_2}\ \rho_{\mathtt{a}_2}\ \mid\delta_{\mathtt{a}_2}\ \rho_{\mathtt{a}_2 1}\ \eta_1\ (1-\eta_1)^*\mid\ \ldots\ \mid\delta_{\mathtt{a}_2}\ \rho_{\mathtt{a}_2 M}\ \eta_M\ (1-\eta_M)^*\\
&\vdots\\
Y_{l_1+l_2}\quad &:=\delta_{\mathtt{a}_2}\ \rho_{\mathtt{a}_2}\ \mid\delta_{\mathtt{a}_2}\ \rho_{\mathtt{a}_2 1}\ \eta_1\ (1-\eta_1)^*\mid\ \ldots\ \mid\delta_{\mathtt{a}_2}\ \rho_{\mathtt{a}_2 M}\ \eta_M\ (1-\eta_M)^*\\
&\vdots\\
Y_{l-l_N+1}\quad &:=\delta_{\mathtt{a}_N}\ \rho_{\mathtt{a}_N}\ \mid\delta_{\mathtt{a}_N}\ \rho_{\mathtt{a}_N 1}\ \eta_1\ (1-\eta_1)^*\mid\ \ldots\ \mid\delta_{\mathtt{a}_N}\ \rho_{\mathtt{a}_N M}\ \eta_M\ (1-\eta_M)^*\}
\end{aligned}$$

$$\begin{aligned}
&\vdots \\
Y_l \quad &:= \delta_{\mathtt{a}_N}\ \rho_{\mathtt{a}_N}\ \mid \delta_{\mathtt{a}_N}\ \rho_{\mathtt{a}_N 1}\ \eta_1\ (1-\eta_1)^* \mid \ \ldots\ \mid \delta_{\mathtt{a}_N}\ \rho_{\mathtt{a}_N M}\ \eta_M\ (1-\eta_M)^*\}
\end{aligned}$$

Now, define

$$\mathcal{Y}_X^{(t,l,m)} =_{\text{def}} \{w \in \mathcal{Y}_X^{(t,l)}\ :\ \sum_{j=1}^{M}(|w|_{\eta_j} + |w|_{(1-\eta_j)}) = m\}.$$

It follows readily that there exists a bijection g between $\mathcal{Y}_X^{(t,l)}$ and $\mathcal{W}_X^{(t,l)}$ such that for every word $w \in \mathcal{Y}_X^{(t,l)}$ there exists a word $w' \in \mathcal{W}_X^{(t,l)}$ which is composed of the same set of letters and thus $\pi(w) = \pi(w')$. Further, this bijection is also a bijection between $\mathcal{Y}_X^{(t,l,m)}$ and $\mathcal{W}_X^{(t,l,m)}$. This yields the next proposition.

Proposition 4.5.

$$\sum_{l=1}^{\infty} \pi\left(\mathcal{W}_X^{(t,l,m)}\right) = \sum_{l=1}^{\infty} \pi\left(\mathcal{Y}_X^{(t,l,m)}\right)$$

Embedding Still, the sum over the weights of these structurally simpler sets depends on the structure of the input string t, which is unknown. Thus, in order to get rid of this dependency on t, we define another regular language $\mathcal{Y}_X$:

$$\begin{aligned}
\mathcal{Y}_X = \{ \quad w \quad &\in \mathcal{B}^{\leq\infty} \\
w \quad &:= Y_1\ Y_2\ \ldots\ Y_N \\
Y_1 \quad &:= (\delta_{\mathtt{A}}\ \rho_{\mathtt{A}} \mid \delta_{\mathtt{A}}\ \rho_{\mathtt{A}1}\ \eta_1\ (1-\eta_1)^* \mid \ \ldots\ |\delta_{\mathtt{A}}\ \rho_{\mathtt{A}M}\ \eta_M\ (1-\eta_M)^*)^* \\
&\vdots \\
Y_N \quad &:= (\delta_{\mathtt{Z}}\ \rho_{\mathtt{Z}} \mid \delta_{\mathtt{Z}}\ \rho_{\mathtt{Z}1}\ \eta_1\ (1-\eta_1)^* \mid \ \ldots\ |\delta_{\mathtt{Z}}\ \rho_{\mathtt{Z}M}\ \eta_M\ (1-\eta_M)^*)^*\}
\end{aligned}$$

Again, $Y_1, \ldots, Y_l$ are placeholders for regular expressions. Define

$$\mathcal{Y}_X^{(m)} =_{\text{def}} \{w \in \mathcal{Y}_X\ :\ \sum_{i=1}^{M}(|w|_{\eta_i} + |w|_{(1-\eta_i)}) = m\}.$$

We have that

$$\mathcal{Y}_X^{(t,l,m)} = \{w \in \mathcal{Y}_X^{(m)}\ :\ \text{for all } \mathtt{a} \in \mathcal{A} \text{ it holds that } |w|_{\delta_{\mathtt{a}}} = |t[1\ldots l]|_{\mathtt{a}}\}$$

and for every $t \in \mathcal{A}^{\infty}$ and all $m \in \mathbb{N}_+$ it holds that

$$\bigcup_{l=1}^{\infty} \mathcal{Y}_X^{(t,l,m)} \subset \mathcal{Y}_X^{(m)}.$$

Thus we get the following proposition.

Proposition 4.6. *For every $t \in \mathcal{A}^{\infty}$ and all $m \in \mathbb{N}_+$ it holds that*

$$\sum_{l=1}^{\infty} \pi\left(\mathcal{Y}_X^{(t,l,m)}\right) = \pi\left(\bigcup_{l=1}^{\infty} \mathcal{Y}_X^{(t,l,m)}\right) \leq \pi\left(\mathcal{Y}_X^{(m)}\right).$$

Altogether, we have established the following relation between the sum over all terms (4.17) over all $l \in \mathbb{N}_+$ and the weight of the set $\mathcal{Y}_X^{(m)}$:

Lemma 4.4. *For $m \in \mathbb{N}_+$ and $t \in \mathcal{A}^\infty$,*

$$\sum_{l=1}^{\infty} \sum_{\alpha \in \mathcal{A}^m} \prod_{i=k}^{l} \delta_{t[i]} \cdot \mu_X(\alpha, t[1 \ldots l]) \leq (1/\tilde{\eta}) \cdot \pi\left(\mathcal{Y}_X^{(m)}\right).$$

Proof.

$$\begin{aligned}
\sum_{l=1}^{\infty} \prod_{i=1}^{l} \delta_{t[i]} \sum_{\alpha \in \mathcal{A}^m} \mu_X(\alpha, t[1 \ldots l]) &= (1/\tilde{\eta}) \cdot \sum_{l=1}^{\infty} \pi\left(\mathcal{W}_X^{(t,l,m)}\right) && \text{by Proposition 4.4} \\
&= (1/\tilde{\eta}) \cdot \sum_{l=1}^{\infty} \pi\left(\mathcal{Y}_X^{(t,l,m)}\right) && \text{by Proposition 4.5} \\
&\leq (1/\tilde{\eta}) \cdot \pi\left(\mathcal{Y}_X^{(m)}\right) && \text{by Proposition 4.6}
\end{aligned}$$

This proves the lemma. □

Bounding $\pi\left(\mathcal{Y}_X^{(m)}\right)$ using the expansion of rational generating functions In order to get a bound on the term $\pi\left(\mathcal{Y}_X^{(m)}\right)$, we proceed as follows: after having given the regular specification of the set $\mathcal{Y}_X$, we translate this specification into the language of generating functions, where we use the variable z to mark the length, i.e., the number $\sum_{i=1}^{M}(|w|_{\eta_i} + |w|_{(1-\eta_i)})$ for a word $w \in \mathcal{Y}_X$. Also, we symbolically use the letters as variables. A regular specification for a set of combinatorial objects translates into a *invariably positive rational* generating function, where we have the following relationship between the operators of the regular description and the algebraic operators: let $a, b \in \mathcal{B}$. Then *union*, i.e., '$a|b$', corresponds to '$a + b$', *Cartesian product* , i.e., '$a\ b$', corresponds to '$a \cdot b$' and *sequence building*, i.e., 'a^*', corresponds to $1/(1-a)$ (where $a \neq \epsilon$). Thus, the regular specification of the language $\mathcal{Y}_X$ readily lends itself to the following ordinary multivariate generating function, where the atomic elements of the regular specification are treated as variables. A formal proof is given in the Appendix 4.A.

Lemma 4.5 (MGF of valid words)**.** *The ordinary multivariate generating function corresponding to the language $\mathcal{Y}_X$ is*

$$\mathcal{Z}_X(\mathbf{z}, \vec{\delta}, \vec{\eta}, \vec{\rho}) = \prod_{\mathtt{a} \in \mathcal{A}} \left(1 - \delta_{\mathtt{a}} \cdot \rho_{\mathtt{a}} - \sum_{j=1}^{M} \frac{\delta_{\mathtt{a}} \cdot \rho_{\mathtt{a}j} \cdot \eta_j \cdot \mathbf{z}}{1 - (1 - \eta_j) \cdot \mathbf{z}}\right)^{-1}$$

where

- $\vec{\eta} = (\eta_1, \ldots, \eta_M, (1 - \eta_1), \ldots, (1 - \eta_M))$,
- $\vec{\delta} = (\delta_{\mathtt{A}}, \ldots, \delta_{\mathtt{Z}})$ *and*
- $\vec{\rho} = (\rho_{\mathtt{A}}, \ldots, \rho_{\mathtt{Z}}, \rho_{\mathtt{A}1}, \ldots \rho_{\mathtt{Z}M})$.

Here the variable z marks the number $\sum_{i=1}^{M}(|w|_{\eta_i} + |w|_{(1-\eta_i)})$ and the other variables mark the number of occurrences of the respective letters.

In order to evaluate $\pi\left(\mathcal{Y}_X^{(m)}\right)$, we follow the *weighted words model*: this is, the former variables are treated as parameters in the new generating function. The respective function is then

$$\tilde{\mathcal{Z}}_X(z) = \prod_{\mathtt{a}\in\mathcal{A}} \left(1 - \delta_{\mathtt{a}}\cdot\rho_{\mathtt{a}} - \sum_{j=1}^{M} \frac{\delta_{\mathtt{a}}\cdot\rho_{\mathtt{a}j}\cdot\eta_j\cdot z}{1-(1-\eta_j)\cdot z}\right)^{-1}.$$

We use the following result on the expansion of rational functions (Theorem 2.1) given in the preliminary Section 2.3.

Theorem *(Expansion of rational functions)* If $f(z)$ is a rational function that is analytic at zero and has poles at $z_1 \le z_2 \le \ldots \le z_k$ then its coefficients are a sum of *exponential polynomials*: there exist k polynomials $\Pi_1(z), \ldots, \Pi_k(z)$ such that for m larger than some fixed m_0,

$$[z^m]\, f(z) = \sum_{j=1}^{k} \Pi_j(m)\cdot\left(\frac{1}{z_j}\right)^m.$$

Furthermore, the polynomial Π_j has degree equal to the order of the pole at z_j minus one.

Clearly, $\tilde{\mathcal{Z}}_X(z)$ is analytic at zero. By construction of the regular language $\mathcal{Y}_X$, all poles of $\tilde{\mathcal{Z}}$ are of order at most N (where N is the cardinality of $\mathcal{A}$). Let $\tilde{z}_1 \le \ldots \le \tilde{z}_{r'}$, where $r' \in \{1,\ldots,N\}$, be these poles (which have not yet been specified) and let $\tilde{z}_1$ the pole of smallest modulus. Then according to the above theorem we have that

$$\pi\left(\mathcal{Y}_X^{(m)}\right) = [z^m]\,\tilde{\mathcal{Z}}_X = \sum_{i=1}^{r'} \Pi_i(m)\cdot\left(\frac{1}{\tilde{z}_i}\right)^m \le \left(\frac{1}{\tilde{z}_1}\right)^m \cdot \sum_{i=1}^{r'} \Pi_i(m), \tag{4.19}$$

where for $i \in \{1,\ldots,r'\}$ Π_i is a polynomial of degree at most equal to the order of the pole at z_i minus one. We claim that the pole of minimum modulus of $\tilde{\mathcal{Z}}_X(z)$ is strictly larger than one. This proves (1) of Claim 4.2.

Claim 4.6. *Assume that for all $i \in \{1,\ldots,N\}$ it holds that $\rho_{\mathtt{a}_i} < 1$ and $\delta_{\mathtt{a}_i} < 1$. Then the pole of minimum modulus $\tilde{z}_1$ of the function $\tilde{\mathcal{Z}}_X(z)$ satisfies $\tilde{z}_1 > 1$.*

Proof. To see that $\tilde{z}_1 > 1$, consider the i-th addend of the product sum for $i \in \{1,\ldots,N\}$. Its denominator is a function f_i of the form

$$\begin{aligned} f_i(z) &= 1 - \delta_{\mathtt{a}_i}\cdot\rho_{\mathtt{a}_i} - \sum_{j=1}^{M} \frac{\delta_{\mathtt{a}_i}\cdot\rho_{\mathtt{a}_i j}\cdot\eta_j\cdot z}{1-(1-\eta_j)\cdot z} \\ &= \sum_{j=1}^{M} \frac{\rho_{\mathtt{a}_i j}}{(1-\rho_{\mathtt{a}_i})}\cdot\left(1 - \delta_{\mathtt{a}_i}\cdot\rho_{\mathtt{a}_i} - \frac{\delta_{\mathtt{a}_i}\cdot(1-\rho_{\mathtt{a}_i})\cdot\eta_j\cdot z}{1-(1-\eta_j)\cdot z}\right), \end{aligned}$$

i.e., it is a convex combination of a family of functions $f_{i,1},\ldots,f_{i,M}$, where for $i \in \{1,\ldots,M\}$,

$$f_{i,j}(z) = 1 - \delta_{\mathtt{a}_i}\cdot\rho_{\mathtt{a}_i} - \frac{\delta_{\mathtt{a}_i}\cdot(1-\rho_{\mathtt{a}_i})\cdot\eta_j\cdot z}{1-(1-\eta_j)\cdot z}$$

has its (unique) root at

$$z_{i,j} = \frac{1 - \delta_{\mathtt{a}_i}\rho_{\mathtt{a}_i}}{1 - \delta_{\mathtt{a}_i}\rho_{\mathtt{a}_i} - (1-\delta_{\mathtt{a}_i})\eta_j} > 1$$

and is positive in $[0, z_{i,j})$. Note that for all $j \in \{1, \ldots, M\}$, $z_{i,j} > 1$ holds particularly because both $\delta_{\mathtt{a}_i} < 1$ and $\rho_{\mathtt{a}_i} < 1$. For *any* convex combination $\sum_{j=1}^{M} \lambda_j \cdot f_{i,j}(z)$ with $\sum_{j=1}^{M} \lambda_j = 1$ it holds that the root z_i of the convex combination satisfies

$$z_i \geq \min_{1\leq j\leq M}\{z_{i,j}\} = \left(\max_{1\leq j\leq M} \frac{(1-\rho_{\mathtt{a}_i}\delta_{\mathtt{a}_i})}{(1-\rho_{\mathtt{a}_i}\delta_{\mathtt{a}_i}) - (1-\delta_{\mathtt{a}_i})\eta_j}\right)^{-1} > 1.$$

Now, the pole $\tilde{z}_1$ of smallest modulus of $\tilde{\mathcal{Z}}_X(z) = \prod_{i=1}^{N} \frac{1}{f_i(z)}$ satisfies

$$\tilde{z}_1 \geq \left(\max_{\substack{1\leq i\leq N\\ 1\leq j\leq M}} \frac{(1-\rho_{\mathtt{a}_i}\delta_{\mathtt{a}_i}) - (1-\delta_{\mathtt{a}_i})\eta_j}{(1-\rho_{\mathtt{a}_i}\delta_{\mathtt{a}_i})}\right)^{-1} > 1.$$

This proves the claim. □

Now, we are in a position to prove the exponentially decreasing upper bound on $\Phi(t, m, d)$.

Proof of Claim 4.2. First note that (1) follows from Claim 4.6. To see that (2) holds:

$$\begin{aligned}
\Phi(t,m,d) &= 2 \cdot \sum_{\alpha\in\mathcal{A}^m} \Big(\sum_{l=1}^{\lceil dm\rceil} \mu_X(\alpha, t[1\ldots l])\Big)^2 && \text{(Def.)}\\
&\leq 2\lceil dm\rceil \cdot \sum_{l=1}^{\infty} \sum_{\alpha\in\mathcal{A}^m} \mu_X(\alpha, t[1\ldots l])^2 && \text{(Ineq. (4.12))}\\
&\leq (2\lceil dm\rceil)/(\tilde{\delta}) \cdot \sum_{l=1}^{\infty} \prod_{i=1}^{l} \delta_{t[i]} \cdot \sum_{\alpha\in\mathcal{A}^m} \mu_X(\alpha, t[1\ldots l]) && \text{(Lem. 4.3)}\\
&\leq (2\lceil dm\rceil)/(\tilde{\eta}\tilde{\delta}) \cdot \pi\left(\mathcal{Y}_X^{(m)}\right) && \text{(Lem. 4.4)}\\
&= (2\lceil dm\rceil)/(\tilde{\eta}\tilde{\delta}) \cdot [z^m]\, \tilde{\mathcal{Z}}_X(z) && \text{(Lem. 4.5)}\\
&\leq (2\lceil dm\rceil)/(\tilde{\eta}\tilde{\delta}) \cdot \sum_{i=1}^{r'} \Pi_i(z) \cdot \left(\frac{1}{\tilde{z}_1}\right)^m && \text{(Thm. 2.1 and Ineq. (4.19))}
\end{aligned}$$

for all $m \geq m_0$, where m_0 is the constant given by Theorem 2.1. Since $\lceil dm\rceil < (d+1)\cdot m$ the claim follows with $\Pi(z) = (2(d+1)m)/(\tilde{\eta}\tilde{\delta}) \cdot \sum_{i=1}^{r'} \Pi_i(z)$. □

4.5.5 Bounding $\Psi(t, m, d)$

In this section, we show that the for $d \in \mathbb{R}_+$ sufficiently large and all $m \in \mathbb{N}_+$,

$$\Psi(t,m,d) \leq c \cdot \left(\frac{1}{\tilde{z}_1}\right)^m,$$

where $\tilde{z}_1$ is defined as in Claim 4.2. To this end, fix

$$d =_{\text{def}} \min\{d' \in \mathbb{R} \;:\; (\sqrt[d']{e} \cdot p_{\max} < 1) \;\wedge\; (e\cdot(d'+1)\cdot(p_{\max})^{d'-1} < 1/\tilde{z}_1)\},$$

where $p_{\max} = \max_{a \in \mathcal{A}} \rho_a$ was the maximum deletion probability. Set

$$c_{4.3} =_{\text{def}} \frac{\sqrt[d+1]{e}}{\tilde{\eta} \cdot (1 - \sqrt[d+1]{e} \cdot p_{\max})}.$$

Here, $e \approx 2.718\ldots$ is the base of the natural logarithm and $\tilde{\eta} = \min_{q \in W} \eta_q$. We prove Claim 4.3 by showing that for the above choice of constants it holds for $m \in \mathbb{N}_+$ that

$$\Psi(t, m, d) \leq c_{4.3} \cdot \left(\frac{1}{\tilde{z}_1}\right)^m.$$

Proof of Claim 4.3. Let d and $p_{\max}$ be defined as above. We start as follows:

$$\begin{aligned}
\Psi(t, m, d) &= 2 \cdot \sum_{\alpha \in \mathcal{A}^m} \Big(\sum_{l=\lceil d \cdot m \rceil + 1}^{\infty} \mu(\alpha, t[1\ldots l]) \Big)^2 \\
&\leq \sum_{\alpha \in \mathcal{A}^m} \sum_{l=\lceil d \cdot m \rceil + 1}^{\infty} \mu(\alpha, t[1\ldots l]) \\
&= \sum_{l=\lceil d \cdot m \rceil + 1}^{\infty} \sum_{\alpha \in \mathcal{A}^m} \mu(\alpha, t[1\ldots l]) \,.
\end{aligned}$$

This holds particularly, because we deal with probabilities, i.e., quantities less than one. Thus, we have bounded $\Psi(t, m, d)$ by that part of the probability mass induced by X on input t which corresponds to the cases in which X has read a relatively long prefix of t. Next, we consider the expansion of $\sum_{\alpha \in \mathcal{A}^m} \mu(\alpha, t[1\ldots l])$ for a fixed $l \geq \lceil d \cdot m \rceil + 1$ due to Lemma 4.2:

$$\begin{aligned}
&\sum_{\alpha \in \mathcal{A}^m} \mu(\alpha, t[1\ldots l]) \\
&\leq 1/\tilde{\eta} \cdot \sum_{m_1 + \ldots + m_l = m} \prod_{i=1}^{l} \Big(f(m_i) \cdot \rho_{t[i]} + (1 - f(m_i)) \cdot \sum_{i=1}^{M} \rho_{t[i]j} \eta_j (1 - \eta_j)^{m_i - 1} \Big) \\
&\leq 1/\tilde{\eta} \cdot \binom{m + l - 1}{l - 1} \cdot (p_{\max})^{l-m}. \qquad (4.20)
\end{aligned}$$

Inequality (4.20) follows from the fact that for $l \geq \lceil d \cdot m \rceil + 1$ in every decomposition $m = m_1 + \ldots + m_l$ of m into l non-negative addends, there are at least $l - m$ indices i, where $1 \leq i \leq l$ such that $m_i = 0$. For each such m_i, it holds that $f(m_i) = 1$ and thus a factor of $\rho_{t[i]} \leq p_{\max}$ is "added" in the product. Also, there are at most $\binom{m+l-1}{l-1}$ such decompositions. Using Stirling's Approximation for the binomial coefficient and Fact A.6 we may further bound as follows:

$$\begin{aligned}
\sum_{l=\lceil d \cdot m \rceil + 1}^{\infty} \sum_{\alpha \in \mathcal{A}^m} \mu(\alpha, t[1\ldots l]) &\leq 1/\tilde{\eta} \cdot \sum_{l=\lceil d \cdot m \rceil + 1}^{\infty} \binom{m + l - 1}{l - 1} \cdot (p_{\max})^{l-m} \\
&\leq 1/\tilde{\eta} \cdot (p_{\max})^{(d-1)m} \cdot \sum_{l=0}^{\infty} \binom{\lceil d \cdot m \rceil + m + l}{m} \cdot (p_{\max})^l \\
&\leq \frac{\sqrt[d+1]{e}}{\tilde{\eta}} \cdot \big(e(d+1)(p_{\max})^{(d-1)} \big)^m \cdot \sum_{l=0}^{\infty} (\sqrt[d+1]{e} \cdot p_{\max})^l
\end{aligned}$$

$$= \frac{\sqrt[d+1]{e}}{\tilde{\eta} \cdot (1 - \sqrt[d+1]{e} \cdot p_{\max})} \cdot \left(e(d+1)(p_{\max})^{(d-1)}\right)^m \qquad (4.21)$$
$$= c_{4.3} \cdot \left(\frac{1}{\tilde{z}_1}\right)^m.$$

Here, (4.21) holds, because $e(d+1)(p_{\max})^{(d-1)} < \frac{1}{\tilde{z}_1}$ by our choice of d. Hence, Claim 4.3 follows. □

4.6 Extensions

4.6.1 Upper and lower bounds for semi-read deterministic perturbation functions

4.6.1.1 Arbitrary read-semi-deterministic perturbation functions

For semi-read-deterministic perturbation functions, we can give non-matching lower and upper bounds. Non-matching upper and lower bounds can also be found in the dichotomous-type of result of Devroye [Dev84]. The upper bound follows from the last theorem. For the lower bound, we need the following proposition

Proposition 4.7. *Let $X = (\{s\}, W, \mu_R, \mu_W, \sigma)$ be a read-semi-deterministic PFA over a finite alphabet $\mathcal{A}$ in canonical form and let*

$$P_2 = \max_{\mathtt{a} \in \mathcal{A}} \left(\sum_{i=1}^{N} Q_{\mathtt{a}}[i]^2 \right).$$

Then for all $\varepsilon > 0$,

$$H(\mathcal{A}^\infty, n, X) \geq 2(1-\varepsilon) \log_{1/P_2} n - o(1).$$

In principle, the proposition follows directly from Proposition 4.2 and the well-known results on the height of random tries over memory-less sources (cf. Section 4.2.1), if would allow as input a set of n equal strings each of which is composed of exactly one input symbol. But, since a trie is built over a set of *distinct* strings, we have to be a little more carefully: to overcome this obstacle, we consider an input set composed of n distinct strings each of which starts with a prefix $t \in \{\mathtt{a}\}^m$ for m sufficiently large such that the influence of the suffixes of the strings can be neglected and the analysis reduces to the classical analysis, i.e., the case of analyzing tries over memory-less random sources.

Proof. Let $X = (\{s\}, W, \mu_R, \mu_W, \sigma)$ be a read-semi-deterministic PFA over the finite alphabet $\mathcal{A}$ in canonical form. Let P_2 be defined as above and let A be a set of n infinite strings each of which starts with $2n$ repetitions of a symbol $\mathtt{a} \in \mathcal{A}$ such that

$$P_2 = \sum_{i=1}^{N} Q_{\mathtt{a}}[i]^2.$$

Clearly, it holds that

$$H(\mathcal{A}^\infty, n, X) = \max_{\substack{S \subseteq \mathcal{A}^\infty \\ \|S\| = n \\ (\forall s,t \in S)\ s \neq t}} \mathbf{E}\left[\max_{s,t \in S} \operatorname{lcp}(X(s), X(t))\right] \geq \mathbf{E}\left[\max_{s,t \in A} \operatorname{lcp}(X(s), X(t))\right].$$

For $0 < \varepsilon < 1$, let $k = 2(1-\varepsilon)\log_{1/P_2} n$. For every string $s \in A$ the probability that $|X(s[1 \ldots 2n])| \geq k$, i.e., that the computation of X on the prefix of s of input length $2n$ has length at least k satisfies

$$\begin{aligned}
\mathbf{P}\{|X(s[1\ldots 2n])| \geq k\} &= 1 - \mathbf{P}\{|X(s[1\ldots 2n])| < k\} \\
&\geq 1 - \mathbf{P}\{\text{at least } 2n-k \text{ symbols are deleted}\} \\
&= 1 - \sum_{i=0}^{k-1} \binom{2n}{i} \cdot (p_{\mathtt{a}})^{2n-i} \\
&\geq 1 - (p_{\mathtt{a}})^n \cdot \left(k \cdot \left(\frac{6n}{k} \right)^k \cdot (p_{\mathtt{a}})^{n-k} \right) \\
&\geq 1 - (p_{\mathtt{a}})^n \cdot \left(k \cdot e^{\ln(6n)/(p_{\mathtt{a}}k) - \ln 1/p_{\mathtt{a}} \cdot n} \right) \\
&= 1 - o((p_{\mathtt{a}})^n).
\end{aligned}$$

Now, with probability $1-o((p_{\mathtt{a}})^n)$ the prefix of length k of the output of the computation of X on s has the same distribution on $\mathcal{A}^k$ as the prefix of a string that is written by a memory-less random source with parameter vector $P \in (0,1)^N$. For such a source and two random strings s, t it holds for every $m \in \mathbb{N}_+$ that

$$\mathbf{P}\{\mathrm{lcp}(s,t) \geq m\} = P_2^m.$$

Let $A = \{s_1, \ldots, s_n\}$ and for $\mu, \nu \in \{1, \ldots, n\}$ let $C_{\mu\nu} = \mathrm{lcp}(s_\mu, s_\nu)$. Now, we can follow essentially the same vein as in the computations for the memory-less random source (see Section 4.2.3 in [Szp01]): since k is exponentially smaller than n, we have for n sufficiently large that

$$S_1 = \sum_{1 \leq i < j \leq n} \mathbf{P}\{C_{ij} \geq k\} \geq \frac{n^2}{4} \cdot P_2^k \cdot (1 - o((p_{\mathtt{a}})^n))^2$$

and

$$S_2' = \sum_{\substack{1 \leq i < j \leq n \\ 1 \leq e < f \leq n \\ \{i,j\} \cap \{e,f\} = \emptyset}} \mathbf{P}\{C_{ij} \geq k \wedge C_{ef} \geq k\} \leq \frac{n^4}{16} \cdot P_2^{2k} \cdot (1 - o((p_{\mathtt{a}})^n))^4 \leq \frac{n^4}{16} \cdot P_2^{2k}$$

and

$$S_2'' = \sum_{\substack{1 \leq i < j \leq n \\ 1 \leq e \leq n}} \mathbf{P}\{C_{ie} \geq k \wedge C_{ej} \geq k\} \leq n^3 \cdot P_3^k \cdot (1 - o((p_{\mathtt{a}}))^n)^3 \leq n^3 \cdot P_3^k,$$

where $P_3 = \sum_{i=1}^N Q_{\mathtt{a}}[i]^3$.

Claim 4.7 (Lemma 4.4 in [Szp01])**.**

$$P_3^{1/3} = \left(\sum_{i=1}^N Q_{\mathtt{a}}[i]^3 \right)^{1/3} \leq \left(\sum_{i=1}^N Q_{\mathtt{a}}[i]^2 \right)^{1/2} = P_2^{1/2}.$$

Let H be the height of a trie which is build over the set A. Using the second moment method, we arrive at

$$\mathbf{P}\{H \geq k\} \geq \frac{S_1^2}{S_1 + S_2' + S_2''}$$

$$\begin{aligned} &\geq \frac{(1-o((p_{\mathtt{a}})^n))^2}{4n^{-2}\cdot P_2^{-k}+1+16\cdot P_3^k/(n\cdot P_2^{2k})} \\ &\geq \frac{(1-o((p_{\mathtt{a}})^n))^2}{1+4n^{-2\varepsilon}+16n^{-\varepsilon}} \\ &= 1-O(1/n^{\varepsilon}), \end{aligned}$$

where the last inequality follows from $k = 2(1-\varepsilon)\log_{1/P_2} n$. The above implies that for every $\varepsilon > 0$,

$$H(\mathcal{A}^\infty, n, X) \geq \mathbf{E}\left[\max_{1\leq i\leq n} \mathrm{lcp}(X(s_i), X(s_j))\right] = \mathbf{E}[H] \geq 2(1-\varepsilon)\log_{1/P_2} n - o(1).$$

This proves the proposition □

Remember that for a semi-read-deterministic perturbation function we can identify with each input symbol $\mathtt{a} \in \mathcal{A}$ a writing state $q_{\mathtt{a}}$. Let $\eta_{\mathtt{a}}$ be the corresponding return probability. The function $\tilde{\mathcal{Z}}$ simplifies as follows

$$\tilde{\mathcal{Z}}_X(z) = \prod_{i=1}^{N}\left(1-\delta_{\mathtt{a}_i}\cdot\rho_{\mathtt{a}_i} - \frac{\delta_{\mathtt{a}_i}\cdot\eta_{\mathtt{a}_i}\cdot(1-\rho_{\mathtt{a}_i})\cdot z}{1-(1-\eta_{\mathtt{a}_i})\cdot z}\right)^{-1}.$$

Here, the pole of the i-th component for $i \in \{1,\ldots,N\}$ is

$$\tilde{z}_i = \left(1-\frac{1-\delta_{\mathtt{a}_i}}{1-\delta_{\mathtt{a}_i}\cdot\rho_{\mathtt{a}_i}}\cdot\eta_{\mathtt{a}_i}\right)^{-1}.$$

The pole of minimum modulus of the function $\tilde{\mathcal{Z}}$ is $\tilde{z} = \min_{1\leq i\leq N} \tilde{z}_i$. This implies the next corollary.

Corollary 4.2. *Let $X = (\{s\}, W, \mu_R, \mu_W, \sigma)$ be a read-semi-deterministic PFA over a finite alphabet $\mathcal{A}$ in canonical form. For $\mathtt{a} \in \mathcal{A}$ let*

$$\delta_{\mathtt{a}} = \rho_{\mathtt{a}} + (1-\rho_{\mathtt{a}})\cdot\max_{\mathtt{b}\in\mathcal{A}}(\mu_W(q_{\mathtt{a}},\mathtt{b},q_{\mathtt{a}}) + \mu_W(q_{\mathtt{a}},\mathtt{b},s))$$

and let

- $\gamma = \max_{\mathtt{a}\in\mathcal{A}}\left(\sum_{\mathtt{b}\in\mathcal{A}}(\mu_W(q_{\mathtt{a}},\mathtt{b},q_{\mathtt{a}}) + \mu_W(q_{\mathtt{a}},\mathtt{b},s))^2\right)$ *and*
- $\tilde{z} = \min_{\mathtt{a}\in\mathcal{A}}\left(1-\frac{1-\delta_{\mathtt{a}}}{1-\delta_{\mathtt{a}}\cdot\rho_{\mathtt{a}}}\cdot\eta_{\mathtt{a}}\right)^{-1}$.

Then for all $\varepsilon > 0$ it holds that

$$2(1-\varepsilon)\log_{1/\gamma} n - o(1) \leq H(\mathcal{A}^\infty, n, X) \leq 2\lceil(1+\varepsilon)\log_{\tilde{z}} n\rceil + o(1).$$

4.6.1.2 Random substitutions

For the perturbation function SUB_p on binary input strings, computed by the PFA depicted in Figure 4.1, we can even show *matching upper and lower bounds* on $H(\{0,1\}^\infty, n, \mathrm{SUB}_p)$.

Theorem 4.3. *Let $p \in (0,1)$ and let SUB_p be the perturbation function corresponding to the PFA SUB_p (depicted in Figure 4.1). Let $P_2 = (1-p)^2 + p^2$. Then for n sufficiently large and all $\varepsilon > 0$,*

$$2(1-\varepsilon)\log_{1/P_2} n - o(1) \leq H(\{0,1\}^\infty, n, \mathrm{SUB}_p) \leq 2\lceil(1+\varepsilon)\log_{1/P_2} n\rceil + o(1),$$

Proof of Theorem 4.3. Assume that $p \in (0,1)$.
For the upper bound note, that SUB_p is a *length-preserving* perturbation function, i.e., for all $\alpha, t \in \{0,1\}^{<\omega}$ we have that $|\mathrm{SUB}_p(t)| = |t|$ and

$$\mathbf{P}\{\mathrm{SUB}_p(t) = \alpha\} = (1-p)^{|t|-d(\alpha,t)} p^{d(\alpha,t)},$$

where $d(\alpha,t)$ denotes the Hamming distance between α and t. Thus for $m \geq 1$ we have

$$\begin{aligned}
\mathbf{P}\{\mathrm{lcp}(\mathrm{SUB}_p(t), \mathrm{SUB}_p(t)) \geq m\} &= \sum_{\alpha\in\{0,1\}^m} \mathbf{P}\{\alpha \sqsubseteq \mathrm{SUB}_p(t)\}^2 \\
&= \sum_{i=0}^{m} \sum_{\substack{\alpha\in\{0,1\}^m \\ d(\alpha,t)=i}} \mathbf{P}\{\alpha \sqsubseteq \mathrm{SUB}_p(t)\}^2 \\
&= \sum_{i=0}^{m} \binom{m}{i} (1-p)^{2(m-i)} p^{2i} \\
&= \left((1-p)^2 + p^2\right)^m
\end{aligned}$$

The upper bound follows from the tail-bound, i.e., Lemma 4.1, where $m_0 = 1$, $\gamma = P_2$ and $\Pi(z) = 1$. The lower bound follows from Proposition 4.7. This proves the Theorem. □

The analysis of the function SUB_p is also interesting from a second perspective: for $p \in (0,1)$ and an arbitrary input string $t \in \mathcal{A}^\infty$, one has

$$\lim_{m\to\infty} \frac{-\ln\left(\sum_{\alpha\in\mathcal{A}^n} \mathbf{P}\{\mathrm{SUB}_p(t)[1\ldots m] = \alpha\}^2\right)}{2m} = \ln\left(p^2 + (1-p)^2\right)/2,$$

which equals the Rényi's entropy of second order for sub-words produced by a biased memoryless random source with parameter vector $P = (p, 1-p) \in (0,1)^2$. Thus, for this particular case out analysis nicely coincides with the known results. For the function INS_{pq} an analogous result seems already out of reach. Yet, there is also a direct analysis for the smoothed trie height $H(\{0,1\}^\infty, n, \mathrm{INS}_{pq})$ which does not require the machinery of star-like automata.

4.6.1.3 A direct analysis of random insertions

For the perturbation function INS_{pq}, computed by the PFA depicted in Figure 4.2, the following result can be shown without using the machinery of star-like automata.

Theorem 4.4. *Let $p, q \in (0,1)$ and let INS_{pq} be the perturbation function corresponding to the PFA INS_{pq} (depicted in Figure 4.2). Let $Q = 1/\left(p\sqrt{(1-q)^2+q^2} + (1-p)\right)^2$ and $P_I = (pq)^2 + (1-pq)^2$ Then for n sufficiently large and all $\varepsilon > 0$,*

$$2(1-\varepsilon)\log_{1/P_I} n - o(1) \leq H(\{0,1\}^\infty, n, X) \leq 2\lceil(1+\varepsilon)\log_Q n\rceil + o(1).$$

In order to prove the theorem, we define a binomial coefficient on strings. Let α and β be two strings over the same alphabet such that their lengths satisfy $|\alpha| = m \leq |\beta|$. We define the binomial coefficient, which counts the number of different occurrences of α as a substring of β:

$$\binom{\beta}{\alpha} =_{\text{def}} \|\{(i_1, \ldots, i_m) : 1 \leq i_1 < \cdots < i_m \leq |\beta| \text{ and } \beta[i_1] \ldots b[i_m] = \alpha\}\|.$$

The next lemma will be very useful.

Lemma 4.6. *For all $\alpha \in \{0,1\}^{<\omega}$, $m \in \mathbb{N}$, and $x, y \in \mathbb{R}$*

$$\sum_{|\beta|=m} \binom{\beta}{\alpha} x^{|\beta|_1 - |\alpha|_1} y^{|\beta|_0 - |\alpha|_0} = \binom{m}{|\alpha|} (x+y)^{m-|\alpha|}.$$

Proof. First, we give a recursive definition of the generalized binomial sum.

$$T_{m,\alpha}(x,y) =_{\text{def}} \sum_{|\beta|=m} \binom{\beta}{\alpha} x^{|\beta|_1 - |\alpha|_1} y^{|\beta|_0 - |\alpha|_0}$$

For arbitrary $b \in \{0,1\}$, we can decompose $T_{m,\alpha b}(x,y)$ as

$$\begin{aligned}
T_{m,\alpha b}(x,y) &= \sum_{|\beta|=m} \binom{\beta}{\alpha b} x^{|\beta|_1 - |\alpha b|_1} y^{|\beta|_0 - |\alpha b|_0} \\
&= \sum_{|\beta|=m-1} \left(\binom{\beta 0}{\alpha b} x^{|\beta 0|_1 - |\alpha b|_1} y^{|\beta 0|_0 - |\alpha b|_0} + \binom{\beta 1}{\alpha b} x^{|\beta 1|_1 - |\alpha b|_1} y^{|\beta 1|_0 - |\alpha b|_0} \right) \\
&= \sum_{|\beta|=m-1} \left(\binom{\beta}{\alpha b} + (1-b) \binom{\beta}{\alpha} \right) x^{|\beta 0|_1 - |\alpha b|_1} y^{|\beta 0|_0 - |\alpha b|_0} + \\
&\quad + \sum_{|\beta|=m-1} \left(\binom{\beta}{\alpha b} + b \binom{\beta}{\alpha} \right) x^{|\beta 1|_1 - |\alpha b|_1} y^{|\beta 1|_0 - |\alpha b|_0} \\
&= \sum_{|\beta|=m-1} \binom{\beta}{\alpha b} (x^{|\beta|_1 - |\alpha b|_1} y^{1+|\beta|_0 - |\alpha b|_0} + x^{1+|\beta|_1 - |\alpha b|_1} y^{|\beta|_0 - |\alpha b|_0}) + \\
&\quad + \sum_{|\beta|=m-1} \binom{\beta}{\alpha} (x^{|\beta|_1 - |\alpha|_1} y^{|\beta|_0 - |\alpha|_0}) ((1-b) x^{-b} y^{b} + b x^{1-b} y^{-(1-b)}) \\
&= (x+y) T_{m-1,\alpha b}(x,y) + T_{m-1,\alpha}(x,y) \qquad (4.22)
\end{aligned}$$

Now we can proof the lemma by induction on m: for $m = |\alpha| \geq 0$, we obviously have $T_{m,\alpha}(x,y) = 1$. Now suppose that the claim is true for $m-1$ and all α' with $|\alpha'| \leq m-1$. Let $\alpha = \alpha' b$. We get

$$\begin{aligned}
T_{m,\alpha}(x,y) &= T_{m,\alpha' b}(x,y) \\
&= (x+y) T_{m-1,\alpha' b}(x,y) + T_{m-1,\alpha'}(x,y) \qquad (4.23) \\
&= (x+y) \binom{m-1}{|\alpha|} (x+y)^{m-1-|\alpha|} + \binom{m-1}{|\alpha|-1} (x+y)^{m-|\alpha|} \qquad (4.24) \\
&= \binom{m}{|\alpha|} (x+y)^{m-|\alpha|}
\end{aligned}$$

Here, Equality (4.23) follows from the the recursion, i.e., Equality (4.22), and Equality (4.24) from the induction hypothesis. The last line follows from the identity $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$ for binomial coefficients. This proves the lemma. □

Having proved the lemma, we are now able to prove the Theorem concerning the smoothed trie height under the perturbation function INS_{pq}.

Proof of Theorem 4.4. For an input string $t \in \{0,1\}^\omega$ and a fixed output string $\alpha \in \{0,1\}^m$ we get[2]

$$\mathbf{P}\{\alpha \sqsubseteq \mathrm{INS}_{pq}(t)\} = \sum_{i=0}^{m} \binom{\alpha}{t[1\ldots i]} (1-p)^i p^{m-i} \cdot \frac{(1-q)^{|\alpha|_1 - |t[1\ldots i]|_1}}{\left(q^{|\alpha|_0 - |t[1\ldots i]|_0}\right)^{-1}} \tag{4.25}$$

Here, Equality (4.25) can be seen as follows: since INS_{pq} is a *elongating* mapping, a fixed prefix α is the result of a computation of INS_{pq} on input $t[1\ldots i]$, where $1 \le i \le m$. For a prefix of fixed length i, there are exactly $\binom{\alpha}{t[1\ldots i]}$ ways to choose the i positions at which the original symbols $t[1], \ldots, t[i]$ appear in α. Then, the symbol 1 must be inserted $|\alpha|_1 - |t[1\ldots i]|_1$ times and the symbol 0 must be inserted $|\alpha|_0 - |t[1\ldots i]|_0$ times, where for a fixed choice the order of insertions is fixed and thus the probability is $(1-q)^{|\alpha|_1 - |t[1\ldots i]|_1} q^{|\alpha|_0 - |t[1\ldots i]|_0}$. Using this we get that

$$\begin{aligned}
&\mathbf{P}\{\mathrm{lcp}(\mathrm{INS}_{pq}(t), \mathrm{INS}_{pq}(t)) \ge m\} \\
&= \sum_{\alpha\in\{0,1\}^m} \mathbf{P}\{\alpha \sqsubseteq \mathrm{INS}_{pq}\}^2 \\
&= \sum_{\alpha\in\{0,1\}^m} \left(\sum_{i=0}^{m} \binom{\alpha}{t[1\ldots i]} (1-p)^i p^{m-i} \cdot \frac{(1-q)^{|\alpha|_1 - |t[1\ldots i]|_1}}{\left(q^{|\alpha|_0 - |t[1\ldots i]|_0}\right)^{-1}}\right)^2 \\
&\le \sum_{\alpha\in\{0,1\}^m} (m+1) \cdot \sum_{i=0}^{m} (1-p)^{2i} p^{2(m-i)} \binom{\alpha}{t[1\ldots i]}^2 \cdot \frac{(1-q)^{2|\alpha|_1 - 2|t[1\ldots i]|_1}}{\left(q^{2|\alpha|_0 - 2|t[1\ldots i]|_0}\right)^{-1}} && (4.26) \\
&\le \sum_{\alpha\in\{0,1\}^m} (m+1) \cdot \sum_{i=0}^{m} \binom{m}{i} (1-p)^{2i} p^{2(m-i)} \binom{\alpha}{t[1\ldots i]} \cdot \frac{(1-q)^{2|\alpha|_1 - 2|t[1\ldots i]|_1}}{\left(q^{2|\alpha|_0 - 2|t[1\ldots i]|_0}\right)^{-1}} && (4.27) \\
&= (m+1) \cdot \sum_{i=0}^{m} \binom{m}{i} (1-p)^{2i} p^{2(m-i)} \sum_{\alpha\in\{0,1\}^m} \binom{\alpha}{t[1\ldots i]} \cdot \frac{(1-q)^{2|\alpha|_1 - 2|t[1\ldots i]|_1}}{\left(q^{2|\alpha|_0 - 2|t[1\ldots i]|_0}\right)^{-1}} \\
&= (m+1) \cdot \sum_{i=0}^{m} \binom{m}{i} (1-p)^{2i} p^{2(m-i)} \binom{m}{i} \left((1-q)^2 + q^2\right)^{m-i} && (4.28) \\
&\le (m+1) \cdot \left(\left(p\sqrt{(1-q)^2 + q^2} + (1-p)\right)^2\right)^m.
\end{aligned}$$

Here, Inequality (4.26) follows from Cauchy's Inequality. Inequality (4.27) holds because $\binom{\alpha}{t[1\ldots i]} \le \binom{m}{i}$ and Equality (4.28) is an application of Lemma 4.6. The upper bound then follows from the Tail bound, where $m_0 = 1$, $\gamma = 1/Q$ and $\Pi(z) = 1 \cdot z + 1$. The lower bound follows from Proposition 4.7. This proves the theorem. □

[2] In order to make the formulas shorter, we write $\frac{a}{b^{-1}}$ in stead of $a \cdot b$.

4.6.2 Smoothed trie height for restricted input sets

Theorem 4.1 gives necessary and sufficient conditions for a perturbation function X over an alphabet $\mathcal{A}$ such that $H(\mathcal{A}^\infty, n, X) \in O(\log n)$ if and only if the conditions are satisfied. Certainly, these conditions are quite restrictive in the sense that they require X to be absolutely perturbing: for every symbol $b \in \mathcal{A}$ it is required that there are at least two distinct symbols that occur with positive probability as prefix of the non-empty computation $X(b)$. In order to account for a more realistic situation, we now show that the requirement of being absolutely perturbing can be weakened. We assume that the alphabet $\mathcal{A}$ is composed of two distinct alphabets $\mathcal{A}_{\bar{P}}$ and $\mathcal{A}_P$, where the subscripts stand for *non-perturbing* and *perturbing*, respectively. The semantics is the following: let $\mathcal{A} = \mathcal{A}_{\bar{P}} \cup \mathcal{A}_P$. Assume that the perturbation function $X : \mathcal{A} \to \mathcal{A}$ is such that the restriction on $\mathcal{A}_{\bar{P}}$ is the identity, i.e., for all $a \in \mathcal{A}$ it holds that $\mathbf{P}\{X(\mathtt{a}) = \mathtt{a}\} = 1$, and the restriction on $\mathcal{A}_P$ is absolutely perturbing. Clearly, in this setting $H(\mathcal{A}^\infty, n, X)$ is unbounded, because for every string $t \in \mathcal{A}_{\bar{P}}^\omega$ it holds that X maps t onto t, deterministically. We therefore consider only inputs from a subset $\mathcal{L} \subset \mathcal{A}^\infty$ which is such that for all $t \in \mathcal{L}$ and every sufficiently long substring of t there is at least one symbol from the set $\mathcal{A}_P$ in the substring. In this revised setting, we can prove that $H(\mathcal{L}, n, X) \in O(\log n)$ if and only if the restriction of X on $\mathcal{A}_P$ satisfies the conditions of Theorem 4.1. Note that in the proof of Lemma 4.3, it is not claimed that the δ's are strictly less than one. It is easy to see that if for all sufficiently long sub-strings of the input a constant fraction of the δ's is strictly less than one then we can amortize those δ's over all other δ's which are equal to one.

Theorem 4.5. *Let $\mathcal{A}_{\bar{P}}$ and $\mathcal{A}_P$ be distinct finite alphabets and let $\mathcal{A} = \mathcal{A}_{\bar{P}} \cup \mathcal{A}_P$. Let $\mathcal{L} \subset \mathcal{A}^\infty$ be a set of infinite strings such that there exists a constant $\tau = \tau(\mathcal{L})$ such that for all $t \in \mathcal{L}$ and $i, j \in \mathbb{N}$ such that $j \geq \tau$ it holds that*

$$\sum_{\mathtt{a} \in \mathcal{A}_P} |t[i \ldots i+j]|_{\mathtt{a}} \geq 1.$$

Let $X = (\{s\}, W, \mu_R, \mu_W, \sigma)$ be a star-like perturbation function over $\mathcal{A}$ in canonical form, which is equal to the identity when restricted on $\mathcal{A}_{\bar{P}}$ and assume that for all $\mathtt{a} \in \mathcal{A}_P$ it holds that $\rho_{\mathtt{a}} < 1$. Then the following statements are equivalent.

(1) $(\forall \mathtt{a} \in \mathcal{A}_P)(\forall \mathtt{a} \in \mathcal{A})$ $\rho_{\mathtt{a}} + \sum_{q \in W} \rho_{\mathtt{a}q} \cdot (\mu_W(q, \mathtt{b}, q) + \mu_W(q, \mathtt{b}, s)) < 1.$

(2) $H(\mathcal{L}, n, X) \in O(\log n)$.

Proof. Let $\mathcal{A}_{\bar{P}} = \{\mathtt{A}', \ldots, \mathtt{Z}'\}$ and $\mathcal{A}_P = \{\mathtt{A}, \ldots, \mathtt{Z}\}$ be distinct finite alphabets and let $\mathcal{A} = \mathcal{A}_{\bar{P}} \cup \mathcal{A}_P$ and let $\mathcal{L}$ and τ be the set of strings with its corresponding constant.

In order to prove the equivalence of the two statements, we claim that (2) $\Rightarrow$ (1) and that (1) $\Rightarrow$ (2). Then, the theorem follows. The first claim follows from the same reasoning as the respective claim in the proof of Theorem 4.1.

Claim 4.8. *In the setting of Theorem 4.5, it holds that* (2) $\Rightarrow$ (1).

Proof. We prove the claim by contraposition: to this end assume that (1) does not hold, i.e., there are symbols $\mathtt{a} \in \mathcal{A}_P$ and $\mathtt{b} \in \mathcal{A}$ such that

$$\mu_R(s, \mathtt{a}, s) + \sum_{q \in W} \mu_R(s, \mathtt{a}, q) \cdot (\mu_W(q, \mathtt{b}, q) + \mu_W(q, \mathtt{b}, s)) = 1.$$

Thus $\mathbf{P}\{\mathtt{b} \sqsubseteq X(\mathtt{a})\} = 1 - \mu_R(s, \mathtt{a}, s)$. Let $t = \mathtt{aa}\ldots$ and let $s = \mathtt{bb}\ldots$. Then X maps t to s with probability one. Therefore, $H(\mathcal{A}^\infty, n, X)$ is unbounded. The claim follows. □

The proof of the second claim is along the lines with the proof of the respective claim in the original theorem: we have

$$\mathbf{P}\{\mathrm{lcp}(X(t), X(t)) \geq m\} \leq \Phi(t, m, d) + \Psi(t, m, d),$$

where for $m \in \mathbb{N}_+$, $d \in \mathbb{R}_+$ and $t \in \mathcal{L}$,

$$\Phi(t, m, d) = \quad 2 \cdot \sum_{\alpha \in \mathcal{A}^m} \Big(\sum_{l=1}^{\lceil d \cdot m \rceil} \mu_X(\alpha, t[1 \ldots l]) \Big)^2$$

and

$$\Psi(t, m, d) = \quad 2 \cdot \sum_{\alpha \in \mathcal{A}^m} \Big(\sum_{l=\lceil d \cdot m \rceil + 1}^{\infty} \mu_X(\alpha, t[1 \ldots l]) \Big)^2.$$

Claim 4.9. *Assume that* (1) *holds with* $\delta < 1$, *i.e.,*

$$\delta = \max_{\substack{\mathtt{a} \in \mathcal{A}_P \\ \mathtt{b} \in \mathcal{A}}} \left(\rho_{\mathtt{a}} + \sum_{j=1}^{M} \rho_{\mathtt{a}j} \cdot (\mu_W(q_j, \mathtt{b}, q_j) + \mu_W(q_j, \mathtt{b}, s)) \right).$$

Let $d \in \mathbb{R}_+$ *be arbitrary but fixed and let* $\gamma = (\delta)^{1/\tau}$. *Then there exists a polynomial* $\Pi(z)$ *of degree at most* $\|\mathcal{A}\|$ *and a constant* $m_0 \in \mathbb{N}_+$ *such that for all* $m \geq m_0$,

$$\Phi(t, m, d) \leq \Pi(m) \cdot \gamma^m.$$

Proof. Let $d \in \mathbb{R}_+$ be arbitrary, but fixed. From Lemma 4.3 and Cauchy's Inequality, we get

$$\Phi(t, m, d) \leq (2\lceil dm \rceil)/\delta \cdot \sum_{l=1}^{\infty} \prod_{i=1}^{l} \delta_{t[i]} \sum_{\alpha \in \mathcal{A}^m} \mu_X(\alpha, t[1 \ldots l]),$$

where $\delta_{\mathtt{A}'} = \ldots = \delta_{\mathtt{Z}'} = 1$ and for $\mathtt{a} \in \mathcal{A}_P$, $\delta_{\mathtt{a}} \leq \delta < 1$. By averaging we find that for $l \in \mathbb{N}_+$,

$$\prod_{i=1}^{l} \delta_{t[i]} \cdot \sum_{\alpha \in \mathcal{A}^m} \mu_X(\alpha, t[1 \ldots l]) \leq \gamma^l/\delta \cdot \sum_{\alpha \in \mathcal{A}^m} \mu_X(\alpha, t[1 \ldots l]). \tag{4.29}$$

Let $\tilde{z}_1$ be the pole minimum modulus of

$$\tilde{\mathcal{Z}}_X(z) = \prod_{\mathtt{a} \in \mathcal{A}} \left(1 - \gamma \rho_{\mathtt{a}} - \sum_{j=1}^{M} \frac{\gamma \cdot \rho_{\mathtt{a}j} \cdot \eta_j \cdot z}{1 - (1 - \eta_j) \cdot z} \right)^{-1}.$$

Now, for $m \geq m_0$, where m_0 is the constant from Theorem 2.1, and $\Pi(z)$ the polynomial of degree at most $\|\mathcal{A}\|$ corresponding to the expansion of the function the sum of the right-hand side of (4.29) can be bounded by

$$\gamma^l \cdot \sum_{\alpha \in \mathcal{A}^m} \mu_X(\alpha, t[1 \ldots l]) \leq \Pi(z) \cdot \left(\frac{1}{\tilde{z}_1} \right)^m.$$

By Claim 4.6 (i.e., that $\gamma < 1$ and for all $\mathtt{a} \in \mathcal{A}$, $\rho_{\mathtt{a}} < 1$) it follows that $\tilde{z}_1 > 1$. Therefore

$$\Phi(t, m, d) \leq 2(d+1)m/\delta^2 \cdot \Pi(m) \cdot \left(\frac{1}{\tilde{z}_1}\right)^m$$

which proves the claim. □

Now, by Claim 4.3 we have that for suitable constants $c, d \in \mathbb{R}_+$ it holds that for all $m \in \mathbb{N}_+$,

$$\Psi(t, m, d) \leq c \cdot \left(\frac{1}{\tilde{z}_1}\right)^m.$$

Together this gives that for $m \geq m_0$,

$$\mathbf{P}\{\mathrm{lcp}(X(t), X(t)) \geq m\} \leq \Psi(t, m, d) + \Phi(t, m, d) \leq (c + 2(d+1)m/\delta^2 \cdot \Pi(m)) \cdot \left(\frac{1}{\tilde{z}_1}\right)^m.$$

Thus we may apply the tail-bound.

Claim 4.10. *In the setting of Theorem 4.5, it holds that* (1) ⇒ (2).

This proves Theorem 4.5 □

4.6.3 Smoothed height of b-tries

In this section, we study the smoothed height of b-tries (see the preliminary Section 2.2.4 for a formal definition of b-tries). For $s_1, \ldots, s_{b+1} \in \mathcal{A}^{\leq\infty}$ let

$$\mathrm{lcp}(s_1, \ldots, s_{b+1}) =_{\mathrm{def}} \max\left(j \in \mathbb{N} \;:\; (\forall i \in \{2, \ldots, b+1\})\; s_1[0 \ldots j] \sqsubseteq s_i\right)$$

be the function that measures the length of the longest common prefix of $s_1, \ldots, s_{b+1}$.

Definition 4.5. *Let $b \in \mathbb{N}_+$ and let $\mathcal{A}$ be a finite alphabet and let $S \subseteq \mathcal{A}^\infty$ be some non-empty set of infinite strings over $\mathcal{A}$. Given a perturbation function $X : \mathcal{A}^{\leq\infty} \to \mathcal{A}^{\leq\infty}$ the* smoothed b-trie height *for n strings over the set S under the perturbation function X, denoted by $H(S, n, X)$, is defined by*

$$H^{(b)}(S, n, x) =_{\mathrm{def}} \max_{\substack{A \subseteq S \\ \|A\| = n}} \mathbf{E}\left[\max_{t_1, \ldots, t_{b+1} \in A} \mathrm{lcp}(X(t_1), \ldots, X(t_{b+1}))\right].$$

We can bound the height of a b-trie using a generalized version of Lemma 4.1. Before, we need the following lemma.

Lemma 4.7. *Let $k \in \mathbb{N}_+$ and let $b \in \mathbb{R}_+$ satisfying $b \geq 1$ and let $b_1 + \ldots + b_k = b$ be a decomposition of b into strictly positive real addends. Let $a_{1,1}, \ldots, a_{1,n}, \ldots, a_{k,1}, \ldots, a_{k,n} \in \mathbb{R}_+$ such that for all $j \in \{2, \ldots, k\}$,*

$$\sum_{i=1}^{n} a_{1,i}^b \geq \sum_{i=1}^{n} a_{j,i}^b.$$

Then

$$\sum_{i=1}^{n} a_{1,i}^{b_1} \cdot \ldots \cdot a_{k,i}^{b_k} \leq \sum_{i=1}^{n} a_{1,i}^b \tag{4.30}$$

Proof. The proof is by induction over k:
Induction basis: for $k = 1$, (4.30) holds with equality.
Induction Step: let $k > 1$ and assume that (4.30) holds for all decompositions of b into up to $k-1$ addends. Then

$$\begin{aligned}
\sum_{i=1}^{n} a_{1,i}^{b_1} \cdot \ldots \cdot a_{k,i}^{b_k} &\leq \left(\sum_{i=1}^{n} (a_{k,i}^{b_k})^{\frac{b}{b_k}}\right)^{\frac{b_k}{b}} \cdot \left(\sum_{i=1}^{n} (a_{1,i}^{b_1} \cdot \ldots \cdot a_{k-1,i}^{b_{k-1}})^{\frac{b}{b-b_k}}\right)^{\frac{b-b_k}{b}} && (4.31)\\
&\leq \left(\sum_{i=1}^{n} a_{1,i}^{b}\right)^{\frac{b_k}{b}} \cdot \left(\sum_{i=1}^{n} a_{1,i}^{b}\right)^{\frac{b-b_k}{b}} && (4.32)\\
&= \sum_{i=1}^{n} a_{1,i}^{b}.
\end{aligned}$$

Here, Inequality (4.31) is an application of Hölder's Inequality (Fact A.4) and (4.32) follows from the induction hypothesis and the assumption, i.e., that $\sum_{i=1}^{n} a_{1,i}^{b} \geq \sum_{i=1}^{n} a_{k,i}^{b}$. This proves the lemma. □

Lemma 4.8 (Tail-bound for smoothed b-trie height)**.** *Let $b \in \mathbb{N}_+$ and let $\mathcal{A}$ be a finite alphabet and let $m_0 \in \mathbb{N}$ and $\gamma \in \mathbb{R}$ satisfying $0 < \gamma < 1$. Let $X : \mathcal{A}^{\leq\infty} \to \mathcal{A}^{\leq\infty}$ be a perturbation function and let $S \in \mathcal{A}^{\infty}$ be a non-empty set of infinite strings. Let $n > \gamma^{-m_0/(b+1)}$. If there is a polynomial $\Pi(z)$ of fixed degree $d \in \mathbb{N}$, such that for all $s \in S$ and all $m \geq m_0$ it holds that the coincidence probability of two independent perturbations of s satisfies*

$$\sum_{\alpha \in \mathcal{A}^m} \mathbf{P}\{\alpha \sqsubseteq X(s)\}^{b+1} \leq \Pi(m) \cdot \gamma^m,$$

then for all $\varepsilon > 0$ it holds that

$$H^{(b)}(S, n, X) \leq (b+1) \cdot \lceil (1+\varepsilon) \log_{1/\gamma} n \rceil + o(1).$$

Proof. The proof follows the same vein as the proof of Lemma 4.1. Let S be a non-empty set of infinite strings over a finite alphabet $\mathcal{A}$. Let $\varepsilon > 0$ and let $k \in \mathbb{N}_+$ be arbitrary. Then from Boole's Inequality, we arrive are

$$H^{(b)}(S, n, X) \leq k + n^{b+1} \cdot \sum_{i=k+1}^{\infty} \max_{t_1, \ldots, t_{b+1} \in S} \mathbf{P}\{\mathrm{lcp}(X(t_1), \ldots, X(t_{b+1})) \geq i\}. \tag{4.33}$$

Now, we can expand each addend of (4.33) as

$$\begin{aligned}
&\max_{t_1, \ldots, t_{b+1} \in S} \mathbf{P}\{\mathrm{lcp}(X(t_1), \ldots, X(t_{b+1})) \geq i\}\\
&= \max_{t_1, \ldots, t_{b+1} \in S} \left(\sum_{\alpha \in \mathcal{A}^i} \mathbf{P}\{\alpha \sqsubseteq X(t_1)\} \cdot \ldots \cdot \mathbf{P}\{\alpha \sqsubseteq X(t_1)\}\right)\\
&\leq \max_{t \in S} \sum_{\alpha \in \mathcal{A}^i} \mathbf{P}\{\alpha \sqsubseteq X(t)\}^{b+1},
\end{aligned}$$

where the last inequality follows from Lemma 4.7. Let $d \in \mathbb{N}_+$ and let $\Pi(z)$ be a polynomial of degree d such that the assumption of the theorem holds. Set $k = (b+1)\cdot\lceil(1+\varepsilon)\log_{1/\gamma} n\rceil \geq m_0$. Then

$$\begin{aligned} H^{(b)}(S,n,X) &\leq k + n^{b+1}\cdot\sum_{i=k+1}^{\infty}\max_{t\in S}\sum_{\alpha\in\mathcal{A}^i}\mathbf{P}\{\alpha\sqsubseteq X(t)\}^{b+1} \\ &\leq (b+1)\cdot\lceil(1+\varepsilon)\log_{1/\gamma} n\rceil + \sum_{i=(b+1)\cdot\lceil(1+\varepsilon)\log_{1/\gamma} n\rceil+1}^{\infty}\Pi(i)\cdot n^{b+1}\cdot\gamma^i \end{aligned}$$

Again, it is easy to see that the latter term is in $o(1)$:

$$\begin{aligned} &\sum_{i=(b+1)\cdot\lceil(1+\varepsilon)\log_{1/\gamma} n\rceil+1}^{\infty}\Pi(i)\cdot n^{b+1}\cdot\gamma^i \\ &= \sum_{i=1}^{\infty}\Pi((b+1)\cdot\lceil(1+\varepsilon)\log_{1/\gamma} n\rceil + i)\cdot n^{b+1}\cdot\gamma^{(b+1)\cdot\lceil(1+\varepsilon)\log_{1/\gamma} n\rceil}\cdot\gamma^i \in o(1). \end{aligned}$$

This proves the lemma. □

Theorem 4.6. *Let $b \in \mathbb{N}_+$ and let X be a star-like string perturbation function over a finite alphabet $\mathcal{A}$ in canonical form, represented by the PFA $X = (\{s\}, W, \mu_R, \mu_W, \sigma)$, where $W = \{q_1,\ldots,q_M\}$, such that for all $\mathtt{a}\in\mathcal{A}$ it holds that $p_{\mathtt{a}} < 1$ and such that Statement (1) of Theorem 4.1 holds with $\delta_{\mathtt{a}} < 1$ for $\mathtt{a}\in\mathcal{A}$, i.e.,*

$$\delta_{\mathtt{a}} = \max_{\mathtt{b}\in\mathcal{A}}\left(\rho_{\mathtt{a}} + \sum_{j=1}^{M}\rho_{\mathtt{a}j}\cdot(\mu_W(q_j,\mathtt{b},q_j) + \mu_W(q_j,\mathtt{b},s))\right).$$

Let $\tilde{z}$ be the pole of minimum modulus of the function

$$\tilde{\mathcal{Z}}_X(z) = \prod_{\mathtt{a}\in\mathcal{A}}\left(1 - \delta_{\mathtt{a}}\cdot\rho_{\mathtt{a}} - \sum_{j=1}^{M}\frac{\delta_{\mathtt{a}}\cdot\rho_{\mathtt{a}j}\cdot\eta_j\cdot z}{1-(1-\eta_j)\cdot z}\right)^{-1}$$

Then for $P = (\tilde{z})^{\lfloor(b+1)/2\rfloor}$ and n sufficiently large and all $\varepsilon > 0$ it holds that

$$H^{(b)}(\mathcal{A}^\infty, n, X) \leq (b+1)\cdot\lceil(1+\varepsilon)\log_P n\rceil + o(1).$$

Proof. Let $b \in \mathbb{N}_+$. If the conditions of the Theorem hold, then there according to Claim 4.4 there are constants $c, d \in \mathbb{R}_+$ and $m_0 \in \mathbb{N}_+$ and a polynomial Π of degree at most N such that for all $m \geq m_0$ and all $s \in \mathcal{A}^\infty$ it holds that

$$\begin{aligned} \sum_{\alpha\in\mathcal{A}^m}\mathbf{P}\{\alpha\sqsubseteq X(s)\}^{b+1} &\leq \sum_{\alpha\in\mathcal{A}^m}\mathbf{P}\{\alpha\sqsubseteq X(s)\}^{2\cdot\lfloor(b+1)/2\rfloor} \\ &\leq \left(\sum_{\alpha\in\mathcal{A}^m}\mathbf{P}\{\alpha\sqsubseteq X(s)\}^2\right)^{\lfloor(b+1)/2\rfloor} \\ &\leq \left((c+\Pi(m))\cdot\left(\frac{1}{\tilde{z}}\right)^m\right)^{\lfloor(b+1)/2\rfloor}. \end{aligned}$$

The theorem then follows readily from Lemma 4.8. □

4.7 Bibliographic notes

A profound introduction into the average-case analysis of algorithms on strings can be found in the book of the same title by Szpankowski [Szp01]. Tries were first introduced and analysed by Fredkin [Fre60] and Knuth (the actual version of the book is [Knu97]).

The height of random tries built over n independent strings produced by an unbiased memory-less source has been analysed, e.g., in [Knu97, Fla83], where the authors of [FO82, Fla83, FS86] used methods from complex analysis to derive the asymptotic distribution of the height of random tries. The techniques therein were then generalized to binary strings produced by a biased memory-less source by the authors of [JR86]. Szpankowski [Szp91], Szpankowski and Jacquet [JS91a] and Szpankowski and Apostolico [AS92] have examined the height of random tries under different premises using a completely different methodology than the previous authors: instead of using methods which are nowadays subsumed under the notion of combinatorial analysis, i.e., generating functions, integral and Mellin transforms, saddle point methods and suchlike, they used probabilistic tools, such as bounds on order statistics of (nearly arbitrarily distributed) random variables and the second-moment-method [AS00] to derive sharp estimates for the expected trie height. Szpankowski [Szp91], Szpankowski and Jacquet [JS91a] and Szpankowski and Apostolico [AS92] have examined the height of random tries under strings produced by a Markovian source. Clément, Valleé and Flajolet [CFV01] have analysed the height and other parameters of random tries under the symbolic dynamical systems. Additionally studies of the height of random tries can be found, e.g., in [Dev02, Dev05].

The height and other parameters of random PATRICIA trees and other trie-like data structures has also been analyzed: clearly, the height of a PATRICA tree over a set of strings is bounded from above by the height of a trie over the same set of strings. Sharper results on the asymptotic distribution function of the various parameters of PATRICIA trees have been derived. Particularly, the height of a PATRICA tree is on average 50 percent smaller (see, e.g., [Pit85, FS86, Szp88, Szp91, Dev92a, Dev02, Dev05] and references therein). In contrast to PATRICIA tries and ordinary tries, where the average height is in $\Theta(\log n)$ for many string density functions, the average height of a LCP trie over a set of n independent binary strings produced by a memory-less source can be shown to be in $O(\log\log n)$ in the biased case and $O(\log^* n)$ in the unbiased (or symmetric) case. The theoretical enhancement which is achieved by the level- and path-compression technique over ordinary tries and has been analysed in [AN93, AN94, Dev01, DS05]. Suffix trees have been analysed under memory-less sources in [JS91b, Szp91, AS92, DSR92, JS94] and under the assumption that the random source satisfies the mixing condition in [Szp93b, Szp93a].

Appendix 4.A A detailed proof of Lemma 4.5

Lemma *(Lemma 4.5)* The ordinary multivariate generating function corresponding to the language $\mathcal{Y}_X$ is

$$\mathcal{Z}_X(\mathbf{z}, \vec{\delta}, \vec{\eta}, \vec{\rho}) = \prod_{\mathtt{a}\in\mathcal{A}} \left(1 - \delta_{\mathtt{a}}\cdot\rho_{\mathtt{a}} - \sum_{j=1}^{M} \frac{\delta_{\mathtt{a}}\cdot\rho_{\mathtt{a}j}\cdot\eta_j\cdot\mathbf{z}}{1-(1-\eta_j)\cdot\mathbf{z}}\right)^{-1}$$

where $\vec{\eta} = (\eta_1, \ldots, \eta_M, (1-\eta_1), \ldots, (1-\eta_M))$, $\vec{\delta} = (\delta_{\mathtt{A}}, \ldots, \delta_{\mathtt{Z}})$ and $\vec{\rho} = (\rho_{\mathtt{A}}, \ldots, \rho_{\mathtt{Z}}, \rho_{\mathtt{A}1}, \ldots \rho_{\mathtt{Z}M})$. Here the variable z marks the number $\sum_{i=1}^{M}(|w|_{\eta_i} + |w|_{(1-\eta_i)})$ and the other variables mark the number of occurrences of the respective letters.

Proof. In order to make the proof more readable, we mark the number of occurrences of a letter by a variable with the corresponding Latin symbol.

- For $i \in \{1, \ldots, N\}$, $\delta_{\mathtt{a}_i}$ is marked by d_i.
- For $i \in \{1, \ldots, N\}$, $\rho_{\mathtt{a}_i}$ is marked by r_i.
- For $i \in \{1, \ldots, N\}$ and $j \in \{1, \ldots, M\}$, $\rho_{\mathtt{a}_i j}$ is marked by $r_{i,j}$.
- For $j \in \{1, \ldots, M\}$, η_j is marked by e_j and
- For $j \in \{1, \ldots, M\}$, $(1-\eta_j)$ is marked by $(1-e_j)$.

Also, z marks the number $\sum_{i=1}^{M}(|w|_{\eta_i} + |w|_{(1-\eta_i)})$. Consider the i-th addend in the product for $i \in \{1, \ldots N\}$. The $\{\delta_{\mathtt{a}_i}\ \rho_{\mathtt{a}_i}\}$ is generated by the multivariate generating function (MGF)

$$g_i(d_i, r_i) = d_i \cdot r_i.$$

For $j \in \{1, \ldots, M\}$, the set of words

$$\{\delta_{\mathtt{a}_i}\ \rho_{\mathtt{a}_i j}\ \eta_j, \delta_{\mathtt{a}_i}\ \rho_{\mathtt{a}_i j}\ \eta_j(1-\eta_j), \delta_{\mathtt{a}_i}\ \rho_{\mathtt{a}_i j}\ \eta_j(1-\eta_j)(1-\eta_j), \ldots\}$$

is generated by the MGF

$$f_{ij}(z, d_i, r_{i,j}, e_j, (1-e_j)) = \frac{d_i \cdot r_{i,j} \cdot e_j \cdot z}{1-(1-e_j)\cdot z}$$

Now $i \in \{1, \ldots, N\}$ the words corresponding to the regular expression

$$(\delta_{\mathtt{a}_i}\ \rho_{\mathtt{a}_i} \mid \delta_{\mathtt{a}_i}\ \rho_{\mathtt{a}_i 1}\ \eta_1\ (1-\eta_1)^* \mid \ \ldots\ \mid \delta_{\mathtt{a}_i}\ \rho_{\mathtt{a}_i M}\ \eta_M\ (1-\eta_M)^*)^* \tag{4.34}$$

are generated by the function

$$f_i(z, d_i, r_i, r_{i,1}, \ldots, r_{i,M}, e_1, \ldots, e_M, (1-e_1), \ldots, (1-e_M)) = \left(1 - g_i(d_i, r_i) - \sum_{j=1}^{M} f_{ij}(z, d_i, r_{i,j}, e_j, (1-e_j))\right)^{-1}$$

$$= \left(1 - d_i \cdot r_i - \sum_{j=1}^{M} \frac{d_i \cdot r_{i,j} \cdot e_j \cdot z}{1-(1-e_j) \cdot z}\right)^{-1} .$$

Now, the set $\mathcal{Y}_X$, which is defined by a regular expression that is the concatenation of the regular expression (4.34) for each input symbol $\mathtt{a}_i$ for $i \in \{1, \ldots, N\}$ is generated the the product over the corresponding MGF's. Re-substituting the respective variables proves the Lemma. □

Chapter 5

Conclusions

per aspera ad astra

Seneca

In Chapter 3, we have studied the complexity of finding spanning trees whose metrics approximate graph metrics under various similarity measures, computed by applying different matrix norms to the distance matrix of the tree and the graph. We could show that all problems are NP-complete independent of the norms used, except for the case of minimizing distances with respect to the L_∞ matrix-norm which is the polynomial-time solvable minimum diameter spanning tree problem [CGM80, HT95]. For the DMST problem with respect to the $\|.\|_{L,p}$ matrix norm, we could give an efficient polynomial time 2-approximation algorithm for all $1 \leq p < \infty$. This version of the problem is particularly interesting for network design and besides this has potential applications in combinatorial optimization. We have given an example from the area of bioinformatics and have shown how the 2-approximation algorithm can be used to find a 2-approximate solution for the multiple sequence alignment problem under the generalized sum-of-pairs objective. For $p > 1$ those alignments tend to be more balanced because large distances contribute much more to the objective than small ones do, due to the convexity of the polynomial functions. For the DAST problems the version with respect to the $\|.\|_{L,\infty}$ matrix norm was particularly interesting because it equals the problem of finding an additive tree spanner. To the best of our knowledge we are the first to prove the hardness of this problem. In order to be able to use such trees in the context of network analysis, i.e., as combinatorial network abstractions, it is inevitable to look for approximation algorithms, exponential algorithms with small bases, or fixed-parameter algorithms. Technically, an interesting problem is left open: no results are known with respect to the spectral norm $\|\cdot\|_2$, i.e., $\|A\|_2 = \lambda_{\max}(A)$ where $\lambda_{\max}(A)$ is the largest eigenvalue of a symmetric matrix $A \in \mathbb{R}^{n \times n}$. Notice that for a symmetric matrix A, we have $\|A\|_2 \leq \|A\|_{L,\infty} \leq \|A\|_{L,p}$ for all $p \in \mathbb{N}_+$ and $\|A\|_2 \leq \|A\|_{L,\infty} \leq \|A\|_1 = \|A\|_\infty$. Can we expect that computing distance-minimizing spanning trees with respect to $\|\cdot\|_2$ is polynomial-time solvable (in the light that NP-completeness appears with coarser norms)?

In Chapter 4, we performed a smoothed analysis of trie height under *star-like* string perturbation functions. Our analysis gives a better analytical explanation to the practical performance of tries and suchlike data structures on non-random inputs. For previous analyses it was necessary to assume that the input strings are generated by some common random mechanism. Our smoothed analysis of trie height shows that this assumption is though sufficient not necessary. Moreover, a much weaker assumption turned out to be sufficient: even

if an input is a worst case input then slight perturbations of that input suffice to turn it into a "good" input. This is particularly interesting for biological sequences because such data are frequently perturbed by mutations. More technically, it has turned out that worst-case inputs are not only very rare in the input space – which already was known from the previous average-case analyzes – but moreover that they constitute isolated peaks in the input space. The extensions of our analysis towards trading off restrictions of the input against conditions for the perturbation functions has shown that it suffices to have very few random perturbations in order to find a logarithmic trie height. Our analysis of the upper bound is asymptotically relatively tight except for one step: the application of Jensen's Inequality in Lemma 4.3. For the other steps, particularly for the over-counting according to Lemma 4.4 and the resulting evaluation of the rational generating function, it does not seem that there is an asymptotically tighter way. Besides this, the extension towards the analysis of b-trie height shows that by increasing the number of strings that can be stored in each leaf of a trie, one can easily diminish the expected height of the data structure and therefore from a practical point of view showing a sightly smaller base at the cost of an even more involved analysis does not seem worthwhile.

The analyses which we performed are only a first step, as they leave many new and challenging research problems, notably, the smoothed analysis of other trie parameters, and the smoothed analysis of parameters of related data structures for information retrieval, e.g., LCP-tries and suffix trees. Besides this, a further line of research is clearly directed towards more general perturbation functions: star-like string perturbation functions only constitute a small sub-class of all possible PFA-based string perturbation functions: particularly, in the model which we have considered, "failures" are independent of earlier failures. For some situations this assumption is unrealistic. Thus it is desirable to consider more general string perturbation functions represented by arbitrary PFAs. Besides this, it is clear that not all possible string perturbation functions can be modeled by PFAs: e.g., *transpositions* of unlimited length require a stronger model of automata. From this point of view, *probabilistic push-down automata* are interesting, because they provide a way to model such random transpositions, which occur frequently in non-random data such as DNA sequences.

Appendix A

Mathematical facts

In this section, we give a number of well-known facts which were used in our calculations.

Probability and combinatorics

Fact A.1 (Multinomial Expansion). *Let $a_1, \ldots, a_m \in \mathbb{R}_+$. Then*

$$\left(a_1 + \ldots + a_m\right)^n = \sum_{k_1+\ldots+k_m=n} \frac{n!}{k_1! \ldots k_m!} a_1^{k_1} \ldots a_m^{k_m}.$$

General inequalities

Fact A.2 (Cauchy's Inequality). *Let $a_1, \ldots, a_n, b_1, \ldots, b_n$ be real numbers. Then*

$$\left(\sum_{i=1}^n a_i b_i\right)^2 \leq \left(\sum_{i=1}^n a_i^2\right) \cdot \left(\sum_{i=1}^n b_i^2\right). \tag{A.1}$$

We also make use of the following form:

$$\left(\sum_{i=1}^n a_i\right)^2 \leq n \cdot \sum_{i=1}^n a_i^2. \tag{A.2}$$

Fact A.3 (Minkowski Inequality). *Let $a_1, \ldots, a_n, b_1, \ldots, b_n$ be real numbers and let $p > 1$. Then*

$$\left(\sum_{i=1}^n |a_i + b_i|^p\right)^{\frac{1}{p}} \leq \left(\sum_{i=1}^n |a_i|^p\right)^{\frac{1}{p}} + \left(\sum_{i=1}^n |b_i|^p\right)^{\frac{1}{p}}.$$

Fact A.4 (Hölder's Inequality). *Let $a_1, \ldots, a_n, b_1, \ldots, b_n$ be real numbers and let $p, q > 1$ satisfying $\frac{1}{p} + \frac{1}{q} = 1$. Then*

$$\sum_{i=1}^n |a_i b_i|^p \leq \left(\sum_{i=1}^n |a_i|^p\right)^{\frac{1}{p}} \cdot \left(\sum_{i=1}^n |b_i|^q\right)^{\frac{1}{q}}.$$

Fact A.5 (Stirlings Approximation)**.** *Let $n, m \in \mathbb{N}_+$. Then*

$$n! = \sqrt{2\pi n} \cdot \left(\frac{n}{e}\right)^n \cdot \left(1 + \Theta\left(\frac{1}{n}\right)\right).$$

Particularly, we have for the Binomial Coefficient that

$$\left(\frac{n}{m}\right)^m \leq \binom{n}{m} \leq \left(\frac{e \cdot n}{m}\right)^m,$$

where $e = 2.7182...$ is the base of the natural logarithm.

Fact A.6. *Let $n \in \mathbb{R}_+$. Then*

$$\left(1 + \frac{1}{n}\right)^n < e < \left(1 + \frac{1}{n}\right)^{n+1},$$

where $e = 2.71...$ is the base of the natural logarithm.

Fact A.7 (Jensen's Inequality)**.** *Let $f : [a, b] \to \mathbb{R}$ be a continuous function that is* convex *in the interval $[a, b]$, i.e., for all $\lambda \in (0, 1)$ and all $x, y \in [a, b]$ it holds that $f(\lambda \cdot x + (1 - \lambda) \cdot y) \leq \lambda \cdot f(x) + (1 - \lambda) \cdot f(y)$. Let $x_1, \ldots, x_n \in [a, b]$ and let $\lambda_1, \ldots, \lambda_n \in \mathbb{R}^+$. Then*

$$f\left(\frac{\sum_{i=1}^n \lambda_i \cdot x_i}{\sum_{i=1}^n \lambda_i}\right) \leq \frac{\sum_{i=1}^n \lambda_i \cdot f(x_i)}{\sum_{i=1}^n \lambda_i}.$$

Bibliography

[AKPW95] Noga Alon, Richard M. Karp, David Peleg, and Douglas West. A graph-theoretic game and its application to the k-server problem. *SIAM Journal on Computation*, 24(1):78–100, 1995.

[AN93] Arne Andersson and Stefan Nilsson. Improved behaviour of tries by adaptive branching. *Information Processing Letters*, 46(6):295–300, 1993.

[AN94] Arne Andersson and Stefan Nilsson. Faster searching in tries and quadtrees - an analysis of level compression. In *Proceesings of the 2nd European Symposium on Algorithms (ESA'94)*, volume 855 of *Lecture Notes in Computer Science*, pages 82–93. Springer-Verlag, Berlin, 1994.

[AN98] Arne Andersson and Stefan Nilsson. Implementing radixsort. *The ACM Journal of Experimental Algorithmics*, 3, 1998.

[ANRV07] Heiner Ackermann, Alantha Newman, Heiko Röglin, and Berthold Vöcking. Decision making based on approximate and smoothed pareto curves. *Theoretical Computer Science*, 378(3):253–270, 2007.

[AS92] Alberto Apostolico and Wojciech Szpankowski. Self-alignments in words and their applications. *Journal of Algorithms*, 13(3):446–467, 1992.

[AS00] Noga Alon and Joel Spencer. *The Probabilistice Method*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley, New York, NY, 2nd edition, 2000.

[AV06] David Arthur and Sergei Vassilvitskii. Worst-case and smoothed analysis of the icp algorithm, with an application to the k-means method. In *Proceedings of the 47rd Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 153–164. IEEE Computer Society, 2006.

[Awe85] Baruch Awerbuch. Complexity of network synchronization. *Journal of the ACM*, 32(4):804–823, 1985.

[BadHS04] Vikas Bansal, Friedhelm Meyer auf der Heide, and Christian Sohler. Labeling smart dust. In *Proceedings of the 12th European Symposium on Algorithms (ESA'04)*, volume 3221 of *Lecture Notes in Computer Science*, pages 77–88. Springer-Verlag, Berlin, 2004.

[Bar96] Yair Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *37th Annual Symposium on Foundations of Computer Science (FOCS'96)*, pages 184–193. IEEE Computer Society, 1996.

[Bar98] Yair Bartal. On approximating arbitrary metrics by tree metrics. In *Proceedings of the 30th Annual ACM Symposium on the Theory of Computing (STOC'98)*, pages 161–168. ACM Press, New York, NY, 1998.

[BBM03] Cyril Banderier, René Beier, and Kurt Mehlhorn. Smoothed analysis of three combinatorial problems. In *Proceedings of the 28th International Symposium on Mathematical Foundations of Computer Science(MFCS'03)*, volume 2747 of *Lecture Notes in Computer Science*, pages 198–207. Springer-Verlag, Berlin, 2003.

[BCD99] Andreas Brandstädt, Victor Chepoi, and Feodor Dragan. Distance approximating trees for chordal and dually chordal graphs. *Journal of Algorithms*, 30(1):166–184, 1999.

[BD02] Avrim Blum and John Dunagan. Smoothed analysis of the perceptron algorithm for linear programming. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'02)*, pages 905 – 914. ACM Press, New York, NY, 2002.

[BDLL04] Andreas Brandstädt, Feodor F. Dragan, Hoàng-Oanh Le, and Van Bang Le. Tree spanners on chordal graphs: complexity and algorithms. *Theoretical Computer Science*, 310(1-3):329–354, 2004.

[BE05] Ulrik Brandes and Thomas Erlebach, editors. *Network Analysis: Methodological Foundations*, volume 3418 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2005.

[Bea65] Murray A. Beauchamp. An improved index of centrality. *Behavioral Science*, 10:161–163, 1965.

[BH98] Ulrik Brandes and Dagmar Handke. NP-completeness results for minimum planar spanners. *Discrete Mathematics and Theoretical Computer Science*, 3(1):1–10, 1998.

[BKMP05] Surender Baswana, Telikepalli Kavitha, Kurt Mehlhorn, and Seth Pettie. New constructions of (α, β)-spanners and purely additive spanners. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'05)*, pages 672–681. ACM Press, New York, NY, 2005.

[BLMS+03] Luca Becchetti, Stefano Leonardi, Alberto Marchetti-Spaccamela, Guido Schäfer, and Tjark Vredeveld. Average case and smoothed competitive analysis of the multi-level feedback algorithm. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS'03)*. IEEE Computer Society, 2003.

[BLP97] Vineet Bafna, Eugene L. Lawler, and Pavel Pevzner. Approximation algorithms for multiple sequence alignment. *Theoretical Computer Science*, 182(2):233–244, 1997.

[Bra05] Richard C. Bradley. Basic properties of strong mixing conditions. A survey and some open questions. *Probability Surveys*, 2:107–144, 2005.

[BS95] Avrim Blum and Joel Spencer. Coloring random and semi-random k-colorable graphs. *Journal of Algorithms*, 19(2):204–234, 1995.

[BS97] Jon L. Bentley and Robert Sedgewick. Fast algorithms for sorting and searching strings. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'97)*, pages 360 – 369. ACM Press, New York, NY, 1997.

[BV03] René Beier and Berthold Vöcking. Random knapsack in expected polynomial time. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC'03)*, pages 232–241. ACM Press, New York, NY, 2003.

[BV06] René Beier and Berthold Vöcking. Typical properties of winners and losers in dicrete optimization. *SIAM Journal on Computing*, 35(4):855–881, 2006.

[BW90] Timothy C. Bell and Ian H. Witten. Source models for natural language. *International Journal of Man-Machine Studies*, 32(5):545–579, 1990.

[Cai94] Leizhen Cai. NP-completeness of minimum spanner problems. *Discrete Applied Mathematics*, 48(2):187–194, 1994.

[Cam78] Paolo M. Camerini. The min-max spanning tree problem and some extensions. *Information Processing Letters*, 7(1):10–14, 1978.

[CC95] Leizhen Cai and Derek G. Corneil. Tree spanners. *SIAM Journal on Discrete Mathematics*, 8(3):359–387, 1995.

[CD00] Victor Chepoi and Feodor Dragan. A note on distance approximating trees in graphs. *European Journal of Combinatorics*, 21(6):761–766, 2000.

[CFV01] Julien Clément, Philippe Flajolet, and Brigitte Vallée. Dynamical sources in information theory: a general analysis of trie structures. *Algorithmica*, 29(1-2):307–369, 2001.

[CG82] Paolo M. Camerini and Giulia Galbiati. The bounded path tree problem. *SIAM Journal on Algebraic and Discrete Methods*, 3(4):474–484, 1982.

[CG84] R. A. Cuninghame-Green. The absolute centre of a graph. *Discrete Applied Mathematics*, 7(3):275–283, 1984.

[CGM80] Paolo M. Camerini, Giulia Galbiati, and Francesco Maffioli. Complexity of spanning tree problems: Part I. *European Journal of Operational Research*, 5(5):346–352, 1980.

[CGM83] Paolo M. Camerini, Giulia Galbiati, and Francesco Maffioli. On the complexity of finding multi-constrained spanning trees. *Discrete Applied Mathematics*, 5(1):39–50, 1983.

[CGM84] Paolo M. Camerini, Giulia Galbiati, and Francesco Maffioli. The complexity of weighted multi-constrained spanning tree problems. In *Proceedings of the Colloquium on the Theory of Algorithms*, volume 44 of *Colloquia Mathematica Societatis János Bolyai*, pages 53–101, 1984.

[Che89] L. Paul Chew. There are planar graphs almost as good as the complete graph. *Journal of Computer and System Sciences*, 39(2):205–219, 1989.

[CLR01] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, 2nd edition, 2001.

[CSTV04] Brad Calder, Timothy Sherwood, Nathan Tuck, and George Varghese. Deterministic memory-efficient string matching algorithms for intrusion detection. In *Proceedings the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (IEEEINFOCOM'04)*, pages 2628–2639. IEEE Computer Society, 2004.

[DadHR+03] Valentina Damerow, Friedhelm Meyer auf der Heide, Harald Räcke, Christian Scheideler, and Christian Sohler. Smoothed motion complexity. In *Proceedings of the 11th European Symposium on Algorithms (ESA'03)*, volume 2832 of *Lecture Notes in Computer Science*, pages 161–171. Springer-Verlag, Berlin, 2003.

[DBCP97] Mikael Degermark, Andrej Brodnik, Svante Carlsson, and Stephen Pink. Small forwarding tables for fast routing lookups. In *Proceedings of the ACM SIGCOMM '97 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 3–14. ACM Press, New York, NY, 1997.

[DDGS03] Elias Dahlhaus, Peter Dankelmann, Wayne Goddard, and Henda C. Swart. MAD trees and distance-hereditary graphs. *Discrete Applied Mathematics*, 131(1):151–167, 2003.

[DE00] Peter Dankelmann and Roger Entringer. Average distance, minimum degree, and spanning trees. *Journal of Graph Theory*, 33(1):1–13, 2000.

[Dev82] Luc Devroye. A note on the average depth of tries. *Computing*, 28:367–371, 1982.

[Dev84] Luc Devroye. A probabilistic analysis of the height of tries and the complexity of triesort. *Acta Informatica*, 21(3):229–237, 1984.

[Dev92a] Luc Devroye. A note on the probabilistic analysis of patricia trees. *Random Structures and Algorithms*, 3(2):203–214, 1992.

[Dev92b] Luc Devroye. A study of trie-like structures under the density model. *Annals of Applied Probability*, 2(2):402–434, 1992.

[Dev01] Luc Devroye. Analysis of random LC tries. *Random Structures and Algorithms*, 19(3-4):359–375, 2001.

[Dev02] Luc Devroye. Laws of large numbers and tail inequalities for random tries and PATRICIA trees. *Journal of Computational and Applied Mathematics*, 142(1):27–37, 2002.

[Dev05] Luc Devroye. Universal asymptotics for random tries and PATRICIA trees. *Algorithmica*, 42:11–29, 2005.

[DH04] Dov Dvir and Gabriel Y. Handler. The absolute center of a network. *Networks*, 43(2):109–118, 2004.

[Die05] Reinhard Diestel. *Graph Theory*. Graduate Texts in Mathematics. Springer-Verlag, Berlin, 3rd edition, 2005.

[DLM96] Mauro Dell'Amico, Martine Labbé, and Francesco Maffioli. Complexity of spanning tree problems with leaf-dependent objectives. *Networks*, 27(3):175–181, 1996.

[DPK82] Narsingh Deo, G. Prabhu, and M.S. Krishnamoorthy. Algorithms for generating fundamental cycles in a graph. *ACM Transactions on Mathematical Software*, 8(1):26–42, 1982.

[DS04] Valentina Damerow and Christian Sohler. Extreme points under random noise. In *Proceedings of the 12th European Symposium on Algorithms (ESA'04)*, volume 3221 of *Lecture Notes in Computer Science*, pages 264–274. Springer-Verlag, Berlin, 2004.

[DS05] Luc Devroye and Wojciech Szpankowski. Probabilistic behavior of asymmetric level compressed tries. *Random Structures and Algorithms*, 19:185–200, 2005.

[DSR92] Luc Devroye, Wojciech Szpankowski, and Bonita Rais. A note on the height of suffix trees. *SIAM Journal on Computing*, 21(1):48–53, 1992.

[EEST05] Michael Elkin, Yuval Emek, Daniel A. Spielman, and Shang-Hua Teng. Lower-stretch spanning trees. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC'05)*, pages 494–503. ACM Press, New York, NY, May 2005.

[EKM+05a] Stefan Eckhardt, Sven Kosub, Moritz G. Maaß, Hanjo Täubig, and Sebastian Wernicke. Combinatorial network abstraction by trees and distances. In *Proceedings of the 16th International Symposium on Algorithms and Computation (ISAAC'05)*, volume 3827 of *Lecture Notes in Computer Science*, pages 1100–1109. Springer-Verlag, Berlin, 2005.

[EKM+05b] Stefan Eckhardt, Sven Kosub, Moritz G. Maaß, Hanjo Täubig, and Sebastian Wernicke. Combinatorial network abstraction by trees and distances. Technical Report TUM-I0502, Technische Universität München, Institut für Informatik, 2005.

[EKN07] Stefan Eckhardt, Sven Kosub, and Johannes Nowak. Smoothed analysis of trie height. Technical Report TUM-I0715, Technische Universiät München, Institut für Informatik, 2007.

[Eme03] Yuval Emek. Low stretch spanning trees. Master's thesis, Dept. of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel, October 2003.

[EP04a] Michael Elkin and David Peleg. $(1 + \epsilon, \beta)$-spanner constructions for general graphs. *SIAM Journal on Computation*, 33(3):608–631, 2004.

[EP04b] Yuval Emek and David Peleg. Approximating minimum max-stretch spanning trees on unweighted graphs. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'04)*, pages 261–270. ACM Press, New York, NY, January 2004.

[Epp00] David Eppstein. Spanning trees and spanners. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, pages 425–461. Elsevier, 2000.

[FD87] Da-Fei Feng and Russell F. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution*, 25(4):351–360, 1987.

[FK98] Sándor P. Fekete and Jana Kremer. Tree spanners in planar graphs. In *Graph-Theoretic Concepts in Computer Science, 24th International Workshop (WG'98)*, volume 1517 of *Lecture Notes in Computer Science*, pages 298–309. Springer-Verlag, Berlin, June 1998.

[FK01a] Uriel Feige and Joe Kilian. Heuristics for semirandom graph problems. *Journal of Computer and System Sciences*, 63(4):639–671, 2001.

[FK01b] Sándor P. Fekete and Jana Kremer. Tree spanners in planar graphs. *Discrete Applied Mathematics*, 108(1–2):85–103, 2001.

[Fla83] Philippe Flajolet. On the performance evaluation of extendible hashing and trie searching. *Acta Informatica*, 20(4):345–369, 1983.

[Fla06] Philippe Flajolet. The ubiquitous digital tree. In *Proceedings of the 23rd International Symposium on Theoretical Aspects of Computer Science (STACS'06)*, volume 3884 of *Lecture Notes in Computer Science*, pages 1–22. Springer-Verlag, Berlin, 2006.

[FLS02] Matteo Fischetti, Giuseppe Lancia, and Paolo Serafini. Exact algorithms for minimum routing cost trees. *Networks*, 39(3):161–173, 2002.

[FO82] Philippe Flajolet and Andrew Odlyzko. The average height of binary trees and other simple trees. *Journal of Computer and System Sciences*, 25(2):171–213, 1982.

[FPZW04] Arthur M. Farley, Andrzej Proskurowski, Daniel Zappala, and Kurt Windisch. Spanners and message distribution in networks. *Discrete Applied Mathematics*, 137(2):159–171, 2004.

[Fre60] E. H. Fredkin. Trie memory. *Communications of the ACM*, 3:490–500, 1960.

[FRT03] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC'03)*, pages 448–455. ACM Press, New York, NY, 2003.

[FS82] Philippe Flajolet and Jean-Marc Steyaert. A branching process arising in dynamic hashing, trie searching and polynomial factorization. In *Proceedings of the 9th International Colloquium on Automata, Languages and Programming (ICALP'82)*, volume 140 of *Lecture Notes in Computer Science*, pages 239 – 251. Springer-Verlag, Berlin, 1982.

[FS86] Philippe Flajolet and Robert Sedgewick. Digital search trees revisited. *SIAM Journal on Computing*, 15:748–767, 1986.

[FS07] Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Web edition, 9th edition, 2007.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[GKP05] Ronald L. Graham, Donald Ervin Knuth, and Oren Patashnik. *Concrete Mathematics*. Addison-Wesley Publishing Co., Reading, MA, 2nd edition, 2005.

[GT01] Michael T. Goodrich and Roberto Tamassia. *Algorithms Design: Foundations, Analysis, and Internet Examples*. John Wiley, New York, NY, 2001.

[Gus93] Daniel M. Gusfield. Efficient methods for multiple sequence alignment with guaranteed error bounds. *Bulletin of mathematical biology*, 55(1):141–154, 1993.

[Gus97] Dan Gusfield. *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge University Press, Cambridge, 1997.

[Hak64] S. Louis Hakimi. Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations Reseach*, 12(3):450–459, 1964.

[Hal02] Eran Halperin. Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs. *SIAM Journal on Computation*, 31(5):1608–1623, 2002.

[Han99] Dagmar Handke. *Graphs with Distance Guarantees*. Doctoral dissertation, Universität Konstanz, Fakultät für Mathematik und Informatik, December 1999.

[HL04] Refael Hassin and Asaf Levin. Minimum restricted diameter spanning trees. *Discrete Applied Mathematics*, 137(3):343–357, 2004.

[HLCW91] Jan-Ming Ho, D. T. Lee, Chia-Hsiang Chang, and C. K. Wong. Minimum diameter spanning trees and related problems. *SIAM Journal on Computation*, 20(5):987–997, 1991.

[Hoc96] Dorit S. Hochbaum. Approximation algorithms for geometric problems. In *Approximation Algorithms for NP-hard Problems*. PWS Publishing, 1996.

[HSP78] S. Louis Hakimi, Edward F. Schmeichel, and J. G. Pierce. On p-centers in networks. *Transportation Science*, 12(1):1–15, 1978.

[HT95] Refael Hassin and Arie Tamir. On the minimum diameter spanning tree problem. *Information Processing Letters*, 53(2):109–111, 1995.

[Hu74] T.C. Hu. Optimum communication spanning trees. *SIAM Journal on Computation*, 3(2):188–195, 1974.

[HWZ02] Steffen Heinz, Hugh E. William, and Justin Zobel. Burst tries: a fast, efficient data structure for string keys. *ACM Transactions on Information Systems*, 20(2):192–223, 2002.

[JLRK78] David S. Johnson, Jan Karel Lenstra, and Alexander H. G. Rinnooy Kan. The complexity of the network design problem. *Networks*, 8:279–285, 1978.

[JR86] Philippe Jacquet and Mireille Régnier. Trie partitioning process: limiting distributions. In *Proceedings of the 11th colloquium on trees in algebra and programming (CAAP'86)*, volume 214 of *Lecture Notes in Computer Science*, pages 196–210. Springer-Verlag, Berlin, 1986.

[JS91a] Philippe Jacquet and Wojciech Szpankowski. Analysis of digital tries with markovian dependency. *IEEE Transactions on Information Theory*, 37(5):1470–1475, 1991.

[JS91b] Philippe Jacquet and Wojciech Szpankowski. What can we learn about suffix trees from independent tries? In *Proceedings of the 2nd Workshop on Algorithms and Data Structures (WADS'91)*, number 519 in Lecture Notes in Computer Science, pages 228–239. Springer-Verlag, Berlin, 1991.

[JS94] Philippe Jacquet and Wojciech Szpankowski. Autocorrelation on words and its applications : Analysis of suffix trees by string-ruler approach. *Journal on Combinatorial Theory, Series A*, 66(2):237–269, 1994.

[JS97] Svante Janson and Wojciech Szpankowski. Analysis of an asymmetric leader election algorithm. *The Electronic Journal of Combinatorics*, 4(1), 1997.

[KH79] Oded Kariv and S. Louis Hakimi. An algorithmic approach to network location problems. I: the p-centers. *SIAM Journal on Applied Mathematics.*, 37(3):513–538, 1979.

[KLM+03] Dieter Kratsch, Hoàng-Oanh Le, Haiko Müller, Erich Prisner, and Dorothea Wagner. Additive tree spanners. *SIAM Journal on Discrete Mathematics*, 17(2):332–340, 2003.

[KNJ04] Dong-Hee Kim, Jae Dong Noh, and Hawoong Jeong. Scale-free trees: The skeletons of complex networks. *Physical Review E*, 70(046126), 2004.

[Knu97] Donald Ervin Knuth. *The Art of Computer Programming*, volume Vol. 3: Sorting and Searching. Addison-Wesley Publishing Co., Reading, MA, 3rd edition, 1997.

[Knu00] Donald Ervin Knuth. *Selected Papers on Analysis of Algorithms*. Number 105 in CSLI Lecture Notes. Cambridge University Press, Cambridge, 2000.

[Kor01] Guy Kortsarz. On the hardness of approximating spanners. *Algorithmica*, 30(3):432–450, 2001.

[KP94] Guy Kortsarz and David Peleg. Generating sparse 2-spanners. *Journal of Algorithms*, 17(2):222–236, 1994.

[LR78] Tze Leung Lai and Herbert Robbins. A class of dependent random variables and their maxima. *Probability Theory and Related Fields*, 42(2):89–111, 1978.

[LS91] Arthur L. Liestman and Thomas C. Shermer. Additive spanners for hypercubes. *Parallel Processing Letters*, 1:35–42, 1991.

[LS93] Arthur L. Liestman and Thomas C. Shermer. Additive graph spanners. *Networks*, 23(4):343–363, 1993.

[Maf73] Francesco Maffioli. On constrained diameter and medium optimal spanning trees. In *Proceedings of the 5th Conference on Optimization Techniques*, volume 4 of *Lecture Notes in Computer Science*, pages 110–117. Springer-Verlag, Berlin, May 1973.

[Mar99] Roesch Martin. Snort: Lightweight intrusion detection for networks. In *Proceedings of the 13th Conference on Systems Administration (LISA'99)*, pages 229–238. USENIX, 1999.

[Min81] Edward Minieka. A polynomial time algorithm for finding the absolute center of a network. *Networks*, 11(4):351–355, 1981.

[MLMBdC05] L. Mora-López, R. Morales-Bueno, and M. Sidrach de Cardona. Modeling time series of climatic parameters with probabilistic finite automata. *Environmental Modelling & Software*, 20(6):753–760, 2005.

[Mor68] Donald G. Morrison. PATRICIA - practical algorithm to retrieve information coded in alphanumeric. *Journal of the ACM*, 15(4):514 – 534, 1968.

[MR05] Bodo Manthey and Rüdiger Reischuk. Smoothed analysis of binary search trees. In *Proceedings of the 16th International Symposium on Algorithms and Computation (ISAAC'05), Porceedings*, volume 3827 of *Lecture Notes in Computer Science*, pages 483–492. Springer-Verlag, Berlin, 2005.

[MS85] Burkhard Monien and Ewald Speckenmeyer. Ramsey numbers and an approximation algorithm for the vertex cover problem. *Acta Informatica*, 22(1):115–123, 1985.

[NK99] Stefan Nilsson and Gunnar Karlsson. IP-address lookup using LC-tries. *IEEE Journal on Selected Areas in Communications*, 17(6):1083–1092, 1999.

[NR03] Rolf Niedermeier and Peter Rossmanith. On efficient fixed-parameter algorithms for weighted vertex cover. *Journal of Algorithms*, 47(2):63–77, 2003.

[NT02] Stefan Nilsson and M. Tikkanen. An experimental study of compression methods for dynamic tries. *Algorithmica*, 33(1):19–33, 2002.

[Pap94] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing Co., Reading, MA, 1994.

[Paz71] Azaria Paz. *Introduction to Probabilistic Automata*. Computer Science and Applied Mathematics. A Series of Monographs and Textbooks. Academic Press, 1971.

[PBMMP98] Dionisio Pérez-Brito, Nenad Mladenovic, and José A. Moreno-Pérez. A note on spanning trees for network location problems. *Yugoslav Journal of Operations Research*, 8(1):141–145, 1998.

[Pee98] Peter H. Peeters. Some new algorithms for location problems on networks. *European Journal of Operational Research*, 104(2):299–309, 1998.

[Pev92] Pavel Pevzner. Multiple alignment, communication cost, and graph matching. *SIAM Journal on Applied Mathematics.*, 52(6):1763–1779, 1992.

[Pit85] Boris Pittel. Asymptotical growth of a class of random trees. *Annals of Probability*, 13(2):414–427, 1985.

[Pit86] Boris Pittel. Paths in a random digital tree: Limiting distributions. *Advances in Applied Probability*, 18(1):139–155, 1986.

[PR01] David Peleg and Eilon Reshef. Low complexity variants of the arrow distributed directory. *Journal of Computer and System Sciences*, 63(3):474–485, 2001.

[Pri97] Erich Prisner. Distance approximating spanning trees. In *14th Annual Symposium on Theoretical Aspects of Computer Science (STACS'97)*, volume 1200 of *Lecture Notes in Computer Science*, pages 499–510. Springer-Verlag, Berlin, 1997.

[PS89] David Peleg and Alejandro A. Schäffer. Graph spanners. *Journal of Graph Theory*, 13(1):99–116, 1989.

[PU87] David Peleg and Jeffrey D. Ullman. An optimal synchronizer for the hypercube. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing (PODC'87)*, pages 77–85. ACM Press, August 1987.

[PU89] David Peleg and Jeffrey D. Ullman. An optimal synchronizer for the hypercube. *SIAM Journal on Computation*, 18(4):740–747, 1989.

[Rab63] Michael O. Rabin. Probabilistic automata. *Information and Control*, 6(3):230–245, 1963.

[Rég81] Mireille Régnier. On the average height of trees in digital search and dynamic hashing. *Information Processing Letters*, 13:64–66, 1981.

[RV07] Heiko Röglin and Berthold Vöcking. Smoothed analysis of integer programming. *Mathematical Programming*, 110(1):21–56, 2007.

[Sab66] Gert Sabidussi. The centrality index of a graph. *Psychometrica*, 31:581–603, 1966.

[Sch94] Uwe Schoening. *Theretische Informatik – kurzgefaßt.* Spektrum-Hochschultaschenbuch. Spektrum Akademischer Verlag, Heidelberg, 3rd edition, 1994.

[Sco69] Allen J. Scott. The optimal network problem: Some computational procedures. *Transportation Research*, 3(2):201–210, 1969.

[Shi92] Paul C. Shields. Entropy and prefixes. *Annals of Probability*, 20(1):403–409, 1992.

[Soa92] José Soares. Graph spanners: a survey. *Congressus Numerantium*, 89:225–238, 1992.

[SOUM05] Hiroshi Sunaga, Toshiyuki Oka, Kiyoshi Ueda, and Hiroaki Matsumura. P2P-based grid architecture for homology searching. In *Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing (P2P'05)*, pages 148 – 149. IEEE Computer Society, 2005.

[SS97] Jacque Sakarovitch and Imre Simon. *Combinatorics on Words*, chapter 6, pages 121–134. Cambridge Mathematical Library. Cambridge University Press, Cambridge, 1997.

[SS05] Guido Schäfer and Naveen Sivadasan. Topology matters: Smoothed competitiveness of metrical task systems. *Theoretical Computer Science*, 341(1-3):216–246, 2005.

[SST06] Arvind Sankar, Daniel A. Spielman, and Shang-Hua Teng. Smoothed analysis of the condition numbers and growth factors of matrices. *SIAM Journal on Matrix Analysis and Applications*, 28(2):446–476, 2006.

[ST01] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC'01)*, pages 296–305. ACM Press, New York, NY, 2001.

[ST03a] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis (motivation and discrete models). In *Proceedings of the 8th International Workshop on Algorithms and Data Structures (WADS'03)*, volume 2748 of *Lecture Notes in Computer Science*, pages 256–270. Springer-Verlag, Berlin, 2003.

[ST03b] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of termination of linear programming algorithms. *Mathematical Programming, Series B*, 97(1-2):237–404, 2003.

[ST04] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time. *Journal of the ACM*, 51(3):385 – 463, 2004.

[SV86] Miklos Santha and Umesh V. Vazirani. Generating quasi-random sequences from semi-random sources. *Journal of Computer and System Sciences*, 33(1):75–87, 1986.

[SZ03] Ranjan Sinha and Justin Zobel. Efficient trie-based sorting of large sets of strings. In *Proceedings of the 26th Australasian computer science conference*, volume 35 of *ACM International Conference Proceeding Series*, pages 11–18. ACM Press, New York, NY, 2003.

[Szp88] Wojciech Szpankowski. Some results on V-ary asymmetric tries. *Journal of Algorithms*, 9(2):224–244, 1988.

[Szp91] Wojciech Szpankowski. On the height of digital trees and related problems. *Algorithmica*, 6:256–277, 1991.

[Szp93a] Wojciech Szpankowski. Asymptotic properties of data compression and suffix trees. *IEEE Transactions on Information Theory*, 39(5):1647–1659, 1993.

[Szp93b] Wojciech Szpankowski. A generalized suffix tree and its (un)expected asymptotic behaviors. *SIAM Journal on Computing*, 22(6):1176–1198, 1993.

[Szp01] Wojciech Szpankowski. *Average Case Analysis of Algorithms on Sequences*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley, New York, NY, 2001.

[Tan96] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall, Upper Saddle River, NJ, 3rd edition, 1996.

[TZ01] Mikkel Thorup and Uri Zwick. Compact routing schemes. In *Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures (SPAA'01)*, pages 1–10. ACM Press, New York, NY, 2001.

[Val01] Brigitte Vallée. Dynamical sources in information theory: Fundamental intervals and word prefixes. *Algorithmica*, 29(1-2):269–306, 2001.

[Ver06] Roman Vershynin. Beyond hirsch conjecture: Walks on random polytopes and smoothed complexity of the simplex method. In *Proceedings of the 47rd Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 133–142. IEEE Computer Society, 2006.

[VRM+97] G. Venkatesan, Udi Rotics, M. S. Madanlal, Johann A. Makowsky, and C. Pandu Rangan. Restrictions of minimum spanner problems. *Information and Computation*, 136(2):143–164, 1997.

[WC04] Bang Ye Wu and Kun-Mao Chao. *Spanning trees and optimization problems*. Dicrete Mathematics and its Applications. Chapman & Hall/CRC, 2004.

[Wil94] Herbert S. Wilff. *Generatingfunctionology*. Academic Press, 2nd edition, 1994.

[WLC+98] Bang Ye Wu, Guiseppe Lancia, Kun-Mao Chao, R. Ravi, and Chuan Yi Tang. A polynomial time approximation scheme for minimum routing cost spanning trees. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete*

Algorithms (SODA'98), pages 21–32. ACM Press, New York, NY, January 1998.

[Won80] Richard T. Wong. Worst-case analysis of network design problem heuristics. *SIAM Journal on Algebraic and Discrete Methods*, 1(1):51–63, 1980.

Index