Saarland University

Doctoral Thesis

Tight(er) Bounds for Similarity Measures, Smoothed Approximation and Broadcasting

Author:
Marvin Künnemann

A dissertation submitted towards the title of
Doctor of Engineering (Dr.-Ing.)
of the Faculties of Natural Sciences and Technology of Saarland University

Saarbrücken, Germany, 2016

Dean:	Prof. Dr. Frank-Olaf Schreyer
Date of Colloquium:	August 3, 2016

Examination Board

Chair:	Prof. Dr. Markus Bläser
Examiners:	Prof. Dr. Benjamin Doerr
	Prof. Dr. Dr. h.c. mult. Kurt Mehlhorn
	Prof. Dr. Emo Welzl
Member of non-professorial academic staff:	Dr. Matthias Függer

SAARLAND UNIVERSITY

Abstract

Faculty of Natural Sciences and Technology I
Computer Science

Doctoral Thesis

Tight(er) Bounds for Similarity Measures, Smoothed Approximation and Broadcasting

by Marvin Künnemann

In this thesis, we prove upper and lower bounds on the complexity of sequence similarity measures, the approximability of geometric problems on realistic inputs, and the performance of randomized broadcasting protocols.

The first part approaches the question why a number of fundamental polynomial-time problems – specifically, Dynamic Time Warping, Longest Common Subsequence (LCS), and the Levenshtein distance – resists decades-long attempts to obtain polynomial improvements over their simple dynamic programming solutions. We prove that any (strongly) subquadratic algorithm for these and related sequence similarity measures would refute the Strong Exponential Time Hypothesis (SETH). Focusing particularly on LCS, we determine a tight running time bound (up to lower order factors and conditional on SETH) when the running time is expressed in terms of all input parameters that have been previously exploited in the extensive literature.

In the second part, we investigate the approximation performance of the popular 2-Opt heuristic for the Traveling Salesperson Problem using the smoothed analysis paradigm. For the Fréchet distance, we design an improved approximation algorithm for the natural input class of c-packed curves, matching a conditional lower bound.

Finally, in the third part we prove tighter performance bounds for processes that disseminate a piece of information, either as quickly as possible (rumor spreading) or as anonymously as possible (cryptogenography).

Universität des Saarlandes

Zusammenfassung

Naturwissenschaftliche-Technische Fakultät I
Fachbereich Informatik

Dissertation

**Tight(er) Bounds for Similarity Measures,
Smoothed Approximation and Broadcasting**

von Marvin Künnemann

Die vorliegende Dissertation beweist obere und untere Schranken an die Komplexität von Sequenzähnlichkeitsmaßen, an die Approximierbarkeit geometrischer Probleme auf realistischen Eingaben und an die Effektivität randomisierter Kommunikationsprotokolle.

Der erste Teil befasst sich mit der Frage, warum für eine Vielzahl fundamentaler Probleme im Polynomialzeitbereich – insbesondere für das Dynamic-Time-Warping, die längste gemeinsame Teilfolge (LCS) und die Levenshtein-Distanz – seit Jahrzehnten keine Algorithmen gefunden werden konnten, die polynomiell schneller sind als ihre einfachen Lösungen mittels dynamischer Programmierung. Wir zeigen, dass ein (im strengen Sinne) subquadratischer Algorithmus für diese und verwandte Ähnlichkeitsmaße die starke Exponentialzeithypothese (SETH) widerlegen würde. Für LCS zeigen wir eine scharfe Schranke an die optimale Laufzeit (unter der SETH und bis auf Faktoren niedrigerer Ordnung) in Abhängigkeit aller bisher untersuchten Eingabeparameter.

Im zweiten Teil untersuchen wir die Approximationsgüte der klassischen 2-Opt-Heuristik für das Problem des Handlungsreisenden anhand des Smoothed-Analysis-Paradigmas. Weiterhin entwickeln wir einen verbesserten Approximationsalgorithmus für die Fréchet-Distanz auf einer Klasse natürlicher Eingaben.

Der letzte Teil beweist neue Schranken für die Effektivität von Prozessen, die Informationen entweder so schnell wie möglich (Rumor-Spreading) oder so anonym wie möglich (Kryptogenografie) verbreiten.

Acknowledgements

First and foremost, I wish to thank everyone who contributed in any way to bringing me to research or who has supported me during my time as a PhD student. I am particularly grateful to Benjamin Doerr, my PhD advisor, for all his guidance during this time and before. I feel fortunate to have been supported to do research early on, and have been greatly inspired by these experiences. I enjoyed being encouraged to pursuit and explore my research interests also independently in various directions and greatly appreciate all the time doing research together.

I wish to thank Kurt Mehlhorn, especially for providing the stimulating research environment at MPII and all his advice and support. Furthermore, I am very grateful to Emo Welzl for agreeing to review this thesis.

I wish to thank my co-authors during my PhD and before, Karl Bringmann, Ning Chen, Radu Curticapean, Benjamin Doerr, Tobias Friedrich, Martin Hoefer, Chengyu Lin, Bodo Manthey, Peihan Miao, Thomas Sauerwald and Magnus Wahlström. In all of these collaborations, I learned a lot.

It was great working in the Algorithms and Complexity department at MPII, with so many friendly and clever colleagues. I was offered countless advice by past and present members, which I greatly appreciate.

I would like to thank all those with whom I shared scientific as well as personal discussions and, almost most importantly, regular coffee. This includes especially Karl Bringmann and Radu Curticapean, who both influenced and inspired me heavily, as well as Maximilian Dylla, Thomas Schaub, and many more.

I thank all my friends who made my time as a PhD student as pleasant as it was, particularly my flatmates who had to endure me during the more stressful periods.

Finally, my deepest thanks go to my parents, my brother and family, and Johanna. In all the time, you remained my constant support and reminder of what is important in life.

Contents

Chapter 1

Introduction

This thesis approaches three topics in the design and analysis of algorithms: conditional lower bounds for polynomial-time problems, approximation algorithms on realistic inputs and randomized communication protocols.

In the first part, we revisit classic string and curve similarity measures including Dynamic Time Warping, Longest Common Subsequence (LCS) and the Levenshtein distance. As fundamental problems with a variety of applications, predominantly in computational biology, natural language processing and signal processing, all of these problems have attracted substantial, decade-spanning interest by the research community. Yet except for small, i.e., polylogarithmic, improvements, the worst-case quadratic running time bounds obtained by simple dynamic programming algorithms remain state-of-the-art. In this thesis, we provide a justification for this long-standing barrier: Any polynomial improvement over the quadratic running time bound would refute the Strong Exponential Time Hypothesis (SETH), a fundamental conjecture about the complexity of solving the satisfiability problem. This puts the lack of significant progress for these "simple" polynomial-time problems into a more favorable perspective – it shows that a breakthrough result for the notoriously hard satisfiability problem is required to break the quadratic-time barrier for these classic sequence similarity measures. Furthermore, we cast our proofs in a framework that provides an abstract reason for the common hardness of these and related sequence similarity measures.

Focusing particularly on LCS, we extend these results in a yet finer-grained analysis and provide an extensive study of its multivariate complexity: We show that even when the running time is expressed in terms of all input parameters studied in the literature, the best known LCS algorithms are optimal up to lower order factors unless SETH fails – with the exception of a certain special case of the problem, for which we provide a new faster algorithm. Such a study of the multivariate complexity of a problem can be regarded as a step beyond general worst-case results, aiming to obtain a more realistic assessment of the hardness of a problem on "typical", realistic inputs.

In Part II, we further explore this direction towards more realistic analyses, this time rather from an algorithmic than a complexity-theoretic perspective. Here, we analyze the performance of approximation algorithms on realistic inputs. First, we perform a smoothed analysis of the popular 2-Opt heuristic for the Traveling Salesperson Problem. The running time of this simple heuristic on points in the Euclidean space perturbed by modest Gaussian noise has been known to beat its worst-case exponential running time. However, its smoothed approximation performance was unsettled. We give an upper bound of $\mathcal{O}(\log(1/\sigma))$ on the approximation performance of 2-Opt on instances perturbed by Gaussian noise with standard deviation σ. This bound smoothly interpolates between the worst-case and average-case bounds of $\mathcal{O}(\log n)$ and $\mathcal{O}(1)$, respectively. To show that this bound is almost tight, we present a lower bound proving that the approximation performance cannot be $o(\log(1/\sigma)/\log\log(1/\sigma))$. Second, we

consider the approximability of the Fréchet distance. A natural class of realistic inputs, i.e., c-packed curves, admits an $(1+\varepsilon)$-approximation in (typically subquadratic) time $\tilde{\mathcal{O}}(cn/\varepsilon)$ [DHPW12]. This positive result has recently been complemented by a lower bound [Bri14] of $\Omega((cn/\sqrt{\varepsilon})^{1-\delta})$ for constant dimension $d \geq 5$, conditional on SETH – however, a certain gap between upper and (conditional) lower bound remained. We effectively settle the complexity of this problem on c-packed curves (at least for $d \geq 5$ and up to lower order factors) by presenting a $\tilde{\mathcal{O}}(cn/\sqrt{\varepsilon})$-time algorithm for any constant dimension d.

Part III of this thesis finally considers randomized communication protocols for publicizing information in a network. We consider two cooperative settings, in both of which some piece of information initially known to one of n players is to be broadcast to the public. In the first setting, the aim is to disseminate the information (called *rumor*) to all players in the least possible number of rounds, where in each round, each informed player can directly communicate with any other player to inform him or her of the rumor. We precisely analyze the classical *randomized push protocol* that in each round lets each informed player choose a player to communicate with uniformly at random. Improving upon classical analyses [FG85; Pit87], we are able to show in particular that the expected number of rounds it takes until all players are informed is $\log_2(n) + \ln(n) \pm \mathcal{O}(1)$, where we give explicit (small) bounds on the constants hidden in the $\mathcal{O}(1)$-notation. In the second setting, the player initially owning the piece of information (this time called *secret*) aims at broadcasting the secret without revealing his or her identity. For the 2-player case, Brody et al. [Bro+14] proved that the best-possible protocol succeeds at correctly communicating a secret single bit without revealing the identity of the secret owner with probability $\textsc{succ} \in [1/3, 3/8]$. Using both theoretical tools as well as a computer-aided protocol search, we are able to show that neither of these bounds – while natural and elegant – are tight by giving the stronger bounds $\textsc{succ} \in [0.3384, 0.3672]$.

As a common theme throughout this thesis, we typically advance both upper and lower bounds on the subject of study, often yielding almost tight results: When analyzing the rumor spreading time of the push protocol, we give stochastic dominance results (in both directions) for the total time to inform all players. When analyzing the 2-Opt heuristic on perturbed inputs, we provide both upper and lower bounds on its approximation performance. In both cases, such attempts to obtain (almost) tight results belong to the basic toolbox of algorithmic analysis (except possibly for obtaining lower bounds on the smoothed approximation performance, which seems much less studied).

For all further results, the close correspondence between upper and lower bounds might possibly be more surprising: (1) To analyze the cryptogenography problem, we introduce a problem reformulation that helps in both finding better protocols (by a computer-aided search) and proving stronger hardness results (using the concavity method proposed in the previous work [Bro+14]), which are quite different tasks. (2) To design our improved approximation algorithm for the Fréchet distance, we have drawn inspiration from the reasons why the non-matching conditional lower bound of [Bri14] could not be improved. (3) Finally, to obtain our tight results on the multivariate complexity of LCS, we had to carefully work directly at the border of hardness proofs and algorithmic results – fortunately, apart from providing tight lower bounds, our approach helped to uncover a faster algorithm on a special case. We are intrigued by how elegantly, especially for problems like LCS and Fréchet distance, SETH-based lower bounds enable us to work simultaneously on faster algorithms and hardness proofs, culminating in many tight running time bounds (up to factors of lower order $n^{o(1)}$ and conditional on SETH).

This concludes the general overview of all parts of this thesis with their results. We provide more detailed introductions to each topic in the first chapter of each part.

1.1 Indication of Source

During my PhD, I was fortunate to be able to work closely with my advisor, Benjamin Doerr, who encouraged me to work on a diverse set of projects, also with other co-authors. As a result of this and the inspiring research environment of MPII, my thesis contains results in three research directions, obtained in collaborations with varying co-authors. Here, I list these collaborations and resulting publications – a complete list of publications (including results not contained in this thesis) is provided in Section 1.2.

The contents of Part I result from joint work with Karl Bringmann. The alignment gadget framework and its applications (particularly the results of Chapters 4 and 7 and Section 6.1) have previously been published in the article *Quadratic Conditional Lower Bounds for String Problems and Dynamic Time Warping* in the proceedings of *FOCS 2015* [BK15b] (with an extended online version accessible at [BK15c]). The multivariate study of the complexity of Longest Common Subsequence (particularly Chapter 5 and Section 6.2) is based on results of an unpublished manuscript titled *Multivariate Fine-Grained Complexity of Longest Common Subsequence* [BK16]. I contributed 50 % at all stages of the project.

Part II splits into (1) joint work with Bodo Manthey that appeared in the proceedings of *ICALP 2015* titled *Towards Understanding the Smoothed Approximation Ratio of the 2-Opt Heuristic* [KM15] and (2) joint work with Karl Bringmann that has previously been published in the proceedings of *ISAAC 2015* titled *Improved Approximation for Fréchet Distance on c-packed Curves Matching Conditional Lower Bounds* [BK15a] (with an extended online version accessible at [BK14]). Note that the contents of [BK15a] (and thus Chapter 11) are also included in similar form in Karl Bringmann's PhD thesis [Bri15]. To both projects, I contributed 50 % at all stages.

The contents of Part III have been obtained solely under the supervision of my PhD advisor, Benjamin Doerr. The results on randomized rumor spreading appeared in the proceedings of *ANALCO 2014* titled *Tight Analysis of Randomized Rumor Spreading in Complete Graphs* [DK14]. The results on the 2-player cryptogenography problem are accepted for publication at *ICALP 2016* [DK16a], titled *Improved Protocols and Hardness Results for the Two-Player Cryptogenography Problem* (an extended online version is accessible at [DK16b]).

1.2 Further Contributions

I list here all articles I published during (and before) my PhD, which I started in August 2012, as well as an unpublished manuscript. Note that [CK13; CK15] contain results of my Master's thesis.

References

[BK15a] Karl Bringmann and Marvin Künnemann. "Improved Approximation for Fréchet Distance on c-Packed Curves Matching Conditional Lower Bounds". In: *Proc. 26th International Symposium on Algorithms and Computation (ISAAC'15)*. 2015, pp. 517–528.

[BK15b] Karl Bringmann and Marvin Künnemann. "Quadratic Conditional Lower Bounds for String Problems and Dynamic Time Warping". In: *Proc. 56th IEEE Symposium on Foundations of Computer Science (FOCS'15)*. 2015, pp. 79–97.

[BK16] Karl Bringmann and Marvin Künnemann. *Multivariate Fine-Grained Complexity of Longest Common Subsequence*. Unpublished Manuscript. 2016.

[Che+15] Ning Chen, Martin Hoefer, Marvin Künnemann, Chengyu Lin, and Peihan Miao. "Secretary Markets with Local Information". In: *Proc. 42nd International Colloquium on Automata, Languages, and Programming (ICALP'15), Part II*. 2015, pp. 552–563.

[CK13] Radu Curticapean and Marvin Künnemann. "A Quantization Framework for Smoothed Analysis of Euclidean Optimization Problems". In: *Proc. 21st Annual European Symposium on Algorithms (ESA'13)*. Received *Best Student Paper Award*. 2013, pp. 349–360.

[CK15] Radu Curticapean and Marvin Künnemann. "A Quantization Framework for Smoothed Analysis of Euclidean Optimization Problems". In: *Algorithmica* 73(3) (2015), pp. 1–28.

[DK13a] Benjamin Doerr and Marvin Künnemann. "How the $(1 + \lambda)$ Evolutionary Algorithm Optimizes Linear Functions". In: *Proc. 15th Annual Conference on Genetic and Evolutionary Computation (GECCO'13)*. 2013, pp. 1589–1596.

[DK13b] Benjamin Doerr and Marvin Künnemann. "Royal Road Functions and the $(1 + \lambda)$ Evolutionary Algorithm: Almost no Speed-up from Larger Offspring Populations". In: *Proc. 2013 IEEE Congress on Evolutionary Computation (CEC'13)*. 2013, pp. 424–431.

[DK14] Benjamin Doerr and Marvin Künnemann. "Tight Analysis of Randomized Rumor Spreading in Complete Graphs". In: *Proc. 11th Workshop on Analytic Algorithmics and Combinatorics (ANALCO'14)*. 2014, pp. 82–91.

[DK15] Benjamin Doerr and Marvin Künnemann. "Optimizing Linear Functions with the Evolutionary Algorithm – Different Asymptotic Runtimes for Different Instances". In: *Theoretical Computer Science* 561 (2015), pp. 3–23.

[DK16a] Benjamin Doerr and Marvin Künnemann. "Improved Protocols and Hardness Results for the Two-Player Cryptogenography Problem". In: *Proc. 43rd International Colloquium on Automata, Languages, and Programming (ICALP'16)*. To appear. 2016.

[DKW10] Benjamin Doerr, Marvin Künnemann, and Magnus Wahlström. "Randomized Rounding for Routing and Covering Problems: Experiments and Improvements". In: *Experimental Algorithms (SEA'10)*. 2010, pp. 190–201.

[DKW11] Benjamin Doerr, Marvin Künnemann, and Magnus Wahlström. "Dependent Randomized Rounding: The Bipartite Case". In: *Proc. 13th Workshop on Algorithm Engineering and Experiments (ALENEX'11)*. 2011, pp. 96–106.

[Doe+09] Benjamin Doerr, Tobias Friedrich, Marvin Künnemann, and Thomas Sauerwald. "Quasirandom Rumor Spreading: An Experimental Analysis". In: *Proc. 10th Workshop on Algorithm Engineering and Experiments (ALENEX'09)*. 2009, pp. 145–153.

[Doe+11] Benjamin Doerr, Tobias Friedrich, Marvin Künnemann, and Thomas Sauerwald. “Quasirandom Rumor Spreading: An Experimental Analysis”. In: *ACM Journal of Experimental Algorithmics* 16 (2011), Article No. 3.3.

[KM15] Marvin Künnemann and Bodo Manthey. “Towards Understanding the Smoothed Approximation Ratio of the 2-Opt Heuristic”. In: *Proc. 42nd International Colloquium on Automata, Languages, and Programming (ICALP’15), Part I*. 2015, pp. 859–871.

Part I

Conditional Lower Bounds for Similarity Measures

Chapter 2

Introduction to Part I

For many classical polynomial-time problems, algorithmic research has been stagnant for decades: some classic algorithm solves the problem, e.g., in time $\tilde{\mathcal{O}}(n^2)$, yet the fastest known algorithm improves upon this running time only by polylogarithmic factors and it is unknown whether significantly faster algorithms exist. Since unconditional lower bounds for these problems seem far beyond reach for current techniques, the research community has spent growing attention to the field of *conditional lower bounds*. Here, the lack of progress towards faster algorithms for a particular problem is explained by a long-standing algorithmic barrier for another, fundamental problem. Put differently, if we are willing to assume that some problem P does not admit algorithms faster than some reasonable time barrier, then by reducing P to other problems in a certain way, we can (conditionally) rule out faster algorithms for these other problems.

The most prominent such approach is 3SUM-hardness, introduced by Gajentaan and Overmars in 1995 [GO95]: Assuming that 3SUM has no (strongly) subquadratic algorithms, a number of lower bounds, especially for problems in computational geometry could be shown. For many other problems, however, it seems impossible to find a reduction from 3SUM.

Subsequently, further popular conjectures have been used to derive conditional hardness results for problems for which 3SUM-hardness does not seem to apply (for a survey of this emerging field, see, e.g., [VW15]). In this thesis, we will focus on reductions from the Strong Exponential Time Hypothesis (SETH)[1] introduced by Impagliazzo and Paturi [IP01; IPZ01]. It asserts that no algorithm beats exhaustive search for the satisfiability problem by an exponential factor.

Hypothesis (Strong Exponential Time Hypothesis (SETH))**.** *For no $\varepsilon > 0$, k-SAT can be solved in time $\mathcal{O}(2^{(1-\varepsilon)N})$ for all $k \geq 3$.*

Note that exhaustive search takes time $2^N \cdot \text{poly}(n)$ and the best known algorithms for k-SAT have a running time of the form $\mathcal{O}(2^{(1-c/k)N})$ for some constant $c > 0$ [Pat+05]. Thus, SETH is a reasonable hypothesis and, due to lack of progress in the last decades, can be considered unlikely to fail.

The idea to use SETH to prove conditional lower bounds for polynomial-time problems dates back to 2005 [Wil05], but only in recent years more and more such conditional lower bounds have been proven, see, e.g., [ABVW15b; AVW14; AVWW14; AGW15; Abb+16b; BI15a; Bri14; PW10; RVW13; BI15b]. Two recent examples, which inspired the results in this part of the thesis, are the conditional lower bounds for Fréchet distance [Bri14] and Levenshtein distance [BI15a]. Both problems are natural *similarity measures* between two sequences (curves or strings, respectively). We study additional classic similarity measures between strings and curves and prove SETH-based lower bounds for the algorithmic task of computing these similarity measures.

[1] In fact, all our reductions will be based on the Orthogonal Vectors Hypothesis, which is implied by SETH; see Section 3.3 for the definition and a discussion.

To this end, we propose a framework based on a technical idea which we refer to as *alignment gadget*. As the first main result of this part, we provide a general quadratic conditional lower bound for all similarity measures admitting such an alignment gadget. This includes problems such as the classical Levenshtein distance, Longest Common Subsequence (LCS) and Dynamic Time Warping (DTW), explaining the lack of polynomial improvements over the dynamic-programming solutions for these fundamental problems.

While this shows that under SETH any algorithm for these problems requires time $n^{2-o(1)}$ in the worst case, there exist a number of specialized algorithms with faster running times on suitably restricted instances for these problems. Indeed, when we express an algorithm's running time in more input parameters than just the problem size n, the optimal running time might have a much more complex behavior and this is only partially addressed by the general quadratic lower bound of our framework. Therefore, in an even finer analysis, we demonstrate a novel paradigm of *multivariate fine-grained complexity* using LCS as a fundamental example: We determine, up to lower order factors, the conditional fine-grained complexity of LCS (i.e., matching upper and lower bounds), expressed in terms of all input parameters that have been algorithmically exploited in the LCS literature so far.

In the following sections, we explain these two main contributions (the alignment gadget framework as well as our study on the multivariate fine-grained complexity of LCS) in more detail.

2.1 Alignment Gadget Framework and Applications

As the first main result of this part of the thesis, we prove quadratic conditional lower bounds for problems which admit a certain technical construction called alignment gadget. We detail here the problems captured in this framework together with the specific results we obtain.

Dynamic Time Warping (DTW). Fix a metric space (M, d) and call any sequence of points in M a *curve*. Let x and y be two strings of lengths n and m, respectively, where we always assume that $n \geq m$. We may *traverse* x and y by starting in their first entries, in any time step advancing to the next entry in x or y or both, and ending in their last entries (see Chapter 3 for details). The cost of such a traversal is the sum over all points in time of the distance between the current entries. The dynamic time warping distance of x and y is the minimal cost of any traversal. This similarity measure can, e.g., readily detect whether two given signals are equal up to time accelerations or decelerations. This property, among others, makes it a very useful measure in practice, with many applications in comparing temporal data such as video and audio, e.g., for speech recognition or music processing (see, e.g., [SC78]). The best known worst-case running time is achieved by a simple dynamic programming algorithm that computes the DTW distance of x and y in time $\mathcal{O}(nm)$. To break this apparent barrier in practice, many heuristics have been designed for this problem (see, e.g., [SC07]).

An important special case that frequently arises in practice is *dynamic time warping on one-dimensional curves*. Here the metric space is $M = \mathbb{R}$ and the distance measure is $d(a, b) := |a - b|$ for any $a, b \in \mathbb{R}$. Even for this important special case the best known algorithm takes time $\mathcal{O}(nm)$. We provide a possible explanation for this situation by proving the following conditional lower bound for DTW on one-dimensional curves.

Theorem 2.1.1. DTW *on one-dimensional curves taking values in* $\{0, 1, 2, 4, 8\} \subseteq \mathbb{R}$ *cannot be solved in time* $\mathcal{O}(n^{2-\varepsilon})$ *for any constant* $\varepsilon > 0$*, unless* SETH *fails.*

This shows that strongly subquadratic algorithms for DTW can be considered unlikely to exist. Specifically, obtaining such algorithms is at least as hard as a breakthrough for satisfiability.[2]

Edit Distance. Let x and y be two strings of lengths n and m, respectively, where we always assume that $n \geq m$. We start in their first characters at positions $(1, 1)$ and traverse them up to their last characters at positions (n, m) using the following operations: If we are at positions (i, j) we may (1) delete a character in x (this costs $c_{\text{del-x}}$ and we advance to $(i + 1, j)$), (2) delete a character in y (this costs $c_{\text{del-y}}$ and we advance to $(i, j + 1)$), (3) match the current characters, which is only possible if $x[i] = y[j]$ (this costs c_{match} and we advance to $(i + 1, j + 1)$), or (4) substitute the current characters, which is only possible if $x[i] \neq y[j]$ (this costs c_{subst} and we advance to $(i + 1, j + 1)$). The minimum total cost of such a sequence of operations is called the edit distance of x and y, and we denote the problem of computing the edit distance by $\text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$. The Levenshtein distance (i.e., the classic edit distance) is $\text{Edit}(1, 1, 0, 1)$. An important special case is Longest Common Subsequence (LCS), which can be seen to be equivalent to $\text{Edit}(1, 1, 0, 2)$. One obtains more variants for other cost choices, e.g., for aligning DNA sequences a classic choice is $\text{Edit}(2, 2, -1, 1)$ [SM97].

Edit Distance admits a natural dynamic programming algorithm with running time $\mathcal{O}(nm)$, which is taught in many undergraduate algorithms courses. Since such string distance measures have ample applications in bioinformatics, natural language processing and data comparison, Levenshtein distance and LCS are well studied with a rich literature focussing on approximation algorithms (see, e.g., [AKO10]) and algorithms that perform well on special cases (see Section 2.2) – we refer to [Nav01] for an extensive survey on exact algorithms. Yet still, the best known worst-case running time (of an exact algorithm) is $\mathcal{O}(nm(\log\log n/\log n)^2 + n)$ by an algorithm due to Masek and Paterson [MP80], i.e., algorithmic improvements have stalled slightly below quadratic time. Even if we restrict the input to strings over a binary alphabet $\{0, 1\}$, no significantly better worst-case running time is known[3]. We present a possible explanation for this state of affairs by proving conditional lower bounds for Edit Distance on binary strings. Note that already for the special case of LCS, finding subquadratic algorithms has been posed as an open problem as early as 1972 [CKK72], with attempts at obtaining lower bounds following shortly afterwards [AHU76][4].

Specifically, our second framework result is a classification of the conditional complexity of $\text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$ for all operation costs $c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}}$: We identify trivial variants where the edit distance is independent of the input x and y, and only depends on n and m. In this case, it can be computed in constant time. For all remaining choices of the operation costs, we prove quadratic-time hardness, even restricted to binary strings. This includes quadratic-time hardness of LCS and Levenshtein distance on binary strings. Compared to the known lower bound for Levenshtein

[2] Note that quadratic SETH-hardness of DTW has been independently also obtained by [ABVW15b] – see Section 2.4 for a comparison of the two approaches.

[3] In this case, the running time of the Masek-Paterson algorithm decreases only slightly to $\mathcal{O}(nm/\log^2 n + n)$.

[4] Aho et al. [AHU76] prove quadratic complexity in the decision tree model, which, however, does not even capture the Masek-Paterson algorithm. On the other hand, when restricting the alphabet to constant size, the complexity drops to linear, rendering this model too powerful to derive non-trivial hardness for this case.

distance [BI15a], our result decreases the alphabet size from 4 to 2 and adds hardness of a large class of problems including LCS[5].

Theorem 2.1.2. $\text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$ *can be solved in constant time if* $c_{\text{subst}} = c_{\text{match}}$ *or* $c_{\text{del-x}} + c_{\text{del-y}} \leq \min\{c_{\text{match}}, c_{\text{subst}}\}$. *Otherwise,* $\text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$ *on binary strings cannot be solved in time* $\mathcal{O}(n^{2-\varepsilon})$ *for any constant* $\varepsilon > 0$*, unless* SETH *fails.*

As the first step of the hardness part of this theorem, for some $0 < c'_{\text{subst}} \leq 2$ that depends on $c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}$ and c_{subst}, we reduce $\text{Edit}(1, 1, 0, c'_{\text{subst}})$ (with "normalized deletion costs") to $\text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$. This reduction is what fails for the trivial cases. We then proceed to prove hardness of $\text{Edit}(1, 1, 0, c'_{\text{subst}})$ using a construction that is parameterized by c'_{subst}.

Unbalanced Inputs. Our main results are most meaningful for inputs with $n \approx m$. It is conceivable that for unbalanced inputs, i.e., $m \ll n$, faster algorithms exist, say the running time of $\mathcal{O}(nm)$ could be reduced to $\tilde{\mathcal{O}}(n + m^2)$. For DTW, we show that such an improvement is unlikely, by proving that "for any m" no algorithm with running time $\mathcal{O}((nm)^{1-\varepsilon})$ exists, assuming SETH. This is analogous to the situation for the Fréchet distance, for which such a lower bound is known as well [Bri14].

Theorem 2.1.3. *Unless* SETH *fails,* DTW *on one-dimensional curves taking values in* $\{0, 1, 2, 4, 8\}$ *cannot be solved in time* $\mathcal{O}((nm)^{1-\varepsilon})$ *for any constant* $\varepsilon > 0$*, and this even holds restricted to instances with* $n^{\alpha-o(1)} \leq m \leq n^{\alpha+o(1)}$ *for any* $0 < \alpha < 1$.

For Edit Distance, Theorem 2.1.2 implies that there is no $\mathcal{O}(m^{2-\varepsilon})$-time algorithm for any $\varepsilon > 0$ (in the worst case over all strings x, y with $|x| \leq n$ and $|y| \leq m$ for any $n \geq m$). Our reduction from SETH cannot result in unbalanced strings, and thus we are not able to prove better lower bounds than $\mathcal{O}(m^{2-\varepsilon})$. This behaviour hints at the possibility of an $\tilde{\mathcal{O}}(n + m^2)$ algorithm for Edit Distance - and indeed there is an algorithm for LCS from 1977 due to Hirschberg [Hir77] matching this time complexity. This algorithm can be generalized to Edit Distance, which we prove in Chapter 6.

Theorem 2.1.4. $\text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$ *admits an* $\tilde{\mathcal{O}}(n + m^2)$*-time algorithm.*

Thus, for unbalanced inputs, DTW and Edit Distance differ in their behavior, but using SETH we can readily explain this difference. Note that we will dive deeper into analyzing the running time in terms of more parameters than just the problem size n (here, we analyzed also the length m of the shorter string or curve), specialized to the simpler problem LCS, in Section 2.2 and Chapter 5.

Reductions from Longest Common Subsequence. Observe that any near-linear time reduction from LCS to another problem P transfers the quadratic-time lower bound of LCS to P. We think that this notion of *LCS-hardness* could be used to prove lower bounds for many string problems (not only distance measures). To support this claim, we present two simple results in this direction in Chapter 7.

A *palindromic subsequence* (also called symmetric subsequence) of a string x of length n is a subsequence z that is the same as its reverse $\text{rev}(z)$. Computing a longest palindromic subsequence is a popular exercise in undergraduate text books (e.g., [Cor+09,

[5]Note that quadratic SETH-hardness of LCS has been independently also obtained in [ABVW15b] – see Section 2.4 for a comparison of our two approaches.

Exercise 15-2]), since it can be easily solved by a reduction to LCS or adapting the dynamic programming solution of LCS, both resulting in an $\mathcal{O}(n^2)$-time algorithm. A *tandem subsequence* of a string x is a subsequence z that can be written as the concatenation $z = yy$ of a string y with itself. In contrast to longest palindromic subsequence, it is non-trivial to compute a longest tandem subsequence in time $\mathcal{O}(n^2)$ [Kos04]. We present reductions from LCS to both of these problems, which yields the following lower bounds.

Theorem 2.1.5. *On binary strings, longest palindromic subsequence and longest tandem subsequence cannot be solved in time $\mathcal{O}(n^{2-\varepsilon})$ for any constant $\varepsilon > 0$, unless* SETH *fails.*

These results show that SETH-based lower bounds via LCS are applicable to string problems that are not necessarily similarity measures.

2.2 Multivariate Complexity of LCS

Let us focus on the conceptually simplest of the framework problems: the problem of computing the longest common subsequence (LCS). Notwithstanding its quadratic-time worst-case time barrier obstructing progress in the general case, a long and successful line of research builds upon the observation that solving LCS on strings with certain structural properties is significantly simpler. This is most prominently witnessed by the UNIX `diff` utility, which quickly compares large, similar files (exploiting that the longest common subsequence in such instances differs from the input strings at only few positions). As a result, after the problem was introduced by Wagner and Fischer in 1974 [WF74], identifying and exploiting structural parameters to obtain faster algorithms has been a decades-long effort [Apo86; AG87; Epp+92; Hir77; HS77; IR09; Mye86; NKY82; Wu+90]. Parameters that have been studied in the literature are, besides the input size $n := \max\{|x|, |y|\}$, the length $m := \min\{|x|, |y|\}$ of the shorter string, the size of the alphabet Σ that x and y are defined on, the length L of a longest common subsequence of x and y, the number $\Delta = n - L$ of deleted symbols in x, the number $\delta = m - L$ of deleted symbols in y, the number of *matching* pairs M, and the number of *dominant* pairs d (see Section 5.1 for their definitions). Among the fastest currently known algorithms are an $\tilde{\mathcal{O}}(n + d)$-algorithm due to Apostolico [Apo86], an $\tilde{\mathcal{O}}(n + \delta L)$-algorithm due to Hirschberg [Hir77] and an $\tilde{\mathcal{O}}(n + \delta\Delta)$-algorithm due to Wu, Manbers, Myers, and Miller [Wu+90]. In the remainder, we refer to such algorithms, whose running time is stated in more parameters than just the problem size n, as *multivariate algorithms*. In Chapter 5, we provide a non-exhaustive survey containing the asymptotically fastest multivariate LCS algorithms.

While the conditional lower bounds of the alignment gadget framework shows that any LCS algorithm takes time $n^{2-o(1)}$ in the worst case unless SETH fails, these results have no direct implications as to whether any of the above multivariate algorithms can be significantly improved in their dependence on the input parameters for special cases of inputs. Thus, we aim to establish hardness of LCS also under general input parameterization with respect to all previously studied parameters.

The motivation for such an even more precise analysis is twofold: (1) It appears imperative to further investigate whether SETH-based lower bounds can fully justify the lack of progress since the early 1990s (apart from lower-order improvements), even when we analyze the running time as a function of all input parameters studied in the literature. (2) If a careful, systematic approach uncovers parameter settings for which no matching lower bounds can be obtained, this might reveal special cases that admit faster algorithms than currently known. In this sense, studying the multivariate complexity of

a problem can be seen as a structured search for faster multivariate algorithms. As a case in point, see Chapter 11 in which we provide an algorithmic result for the Fréchet distance that was inspired by the reasons why a SETH-based lower bound could not be strengthened to match the best known upper bound.

To obtain such fine-grained results, we systematically study special cases of LCS that arise from polynomial restrictions of any of the previously studied input parameters. Informally, we define a *parameter setting* (or polynomial restriction of the parameters) as the subset of all LCS instances where each input parameter is individually bound to a polynomial relation with the input size n, i.e., for each parameter p we fix a constant α_p and restrict the instances such that p attains a value $\Theta(n^{\alpha_p})$. An algorithm for a specific parameter setting of LCS receives as input two strings x and y guaranteed to satisfy the parameter setting and outputs (the length of) an LCS of x and y. We call a parameter setting trivial if it is satisfied by at most a finite number of instances; this happens if the restrictions on different parameters are contradictory. For each non-trivial parameter setting, we construct a family of hard instances via a reduction from satisfiability, thus obtaining a conditional lower bound. This greatly extends the construction of hard instances for the $n^{2-o(1)}$ lower bound of the alignment gadget framework. More precisely, we obtain the following results, which are developed in full formality in Chapter 5.

Results for Large Alphabets. Since we only consider exact algorithms, any algorithm for LCS takes time $\Omega(n)$. Beyond this trivial bound, for any non-trivial parameter setting we obtain a SETH-based lower bound of

$$\min\big\{d,\, \delta\Delta,\, \delta m\big\}^{1-o(1)}.$$

This bound is matched by the known algorithms with running times $\tilde{\mathcal{O}}(n+d)$, $\tilde{\mathcal{O}}(n+\delta L)$ and $\tilde{\mathcal{O}}(n+\delta\Delta)$.[6] Thus, our lower bound very well explains the lack of progress since 1990 (apart from lower-order factors), as these three algorithms were already known at that time.

Results for Constant Alphabet Size. For the alphabet size $|\Sigma|$, we do not only consider the case of a polynomial relation with n, but also the important special cases of $|\Sigma|$ being any fixed constant. We show that our conditional lower bound for polynomial alphabet size also holds for any constant $|\Sigma| \geq 3$. For $|\Sigma| = 2$, we instead obtain a SETH-based lower bound of

$$\min\big\{d,\, \delta\Delta,\, \delta M/n\big\}^{1-o(1)}.$$

This lower bound is weaker than the lower bound for $|\Sigma| \geq 3$ (it is easy to see that the term $\delta M/n$ is at most δm). Surprisingly, a stronger lower bound would refute SETH: Motivated by the difficulties to obtain the same lower bound as for $|\Sigma| \geq 3$, we discovered an algorithm with running time $\mathcal{O}(n+\delta M/n)$ for $|\Sigma| = 2$, thus matching our conditional lower bound. To the best of our knowledge, this algorithm provides the first polynomial improvement for a special case of LCS since 1990, so while its practical relevance is unclear, we succeeded in uncovering a tractable special case. Interestingly, our algorithm and lower bounds show that the multivariate fine-grained complexity of LCS differs

[6] It might not be immediate to the reader that these bounds indeed match the lower bound and do not contradict it. The parameter relations from which this observation follows are explained in detail in Chapter 5.

polynomially between $|\Sigma| = 2$ and $|\Sigma| \geq 3$. So far, the running time of the fastest known algorithms for varying alphabet size differed at most by a logarithmic factor in $|\Sigma|$.

We find it surprising that the hardness assumption SETH is not only sufficient to prove a worst-case quadratic lower bound for LCS, but extends to the *complete spectrum* of multivariate algorithms using the previously used 7 parameters, thus proving an optimal running time bound which was implicitly discovered by the computer science community within the first 25 years of research on LCS (except for the case of $\Sigma = \{0, 1\}$, for which we provide a missing algorithm).

2.3 Technical Contributions

We highlight some of our technical contributions.

Alignment Gadget Framework. We introduce a framework for proving SETH-based lower bounds for a general class of similarity measures $\boldsymbol{\delta}$. It is based on a construction that we call *alignment gadget*. Given instances $x_1, \ldots, x_n$ and $y_1, \ldots, y_m$, $m \leq n$, an alignment gadget consists of (a way to construct) two instances x, y whose similarity $\boldsymbol{\delta}(x, y)$ is closely related to $\sum_{(i,j)\in\Lambda} \boldsymbol{\delta}(x_i, y_j)$, where $\Lambda = \{(i_1, 1), \ldots, (i_m, m)\}$ is the best-possible ordered alignment of the numbers in $[m]$ to $[n]$ (for details, we refer to Chapter 4). We prove a quadratic lower bound for any similarity measure admitting such an alignment gadget (and additionally, so called *coordinate values*). This proof is a simplified version of a construction in the known lower bound for Levenshtein distance [BI15a], which is also closely related to the lower bound for Fréchet distance [Bri14].

Working with our framework has three advantages: First, it unifies three constructions that are separate proof steps in other SETH-based lower bounds [BI15a; Bri14], thus reducing the amount of work necessary to prove SETH-based lower bounds. Second, it hides the reduction from satisfiability, providing a level of abstraction that allows to ignore the details of the satisfiability problem and instead focus on the details of the problem we reduce to. This makes it possible to tackle general problems such as $\text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$, where the reduction depends on parameters of the problem, without resulting in an overly complex proof. Third, subsequent work [Abb+16a] shows that the general definition of our framework captures even more powerful variants of the satisfiability problem than merely CNF-SAT, yielding even stronger hardness (based on a weaker assumption than SETH)[7].

We present alignment gadgets for DTW, LCS and Edit Distance[8]. This part needs careful problem-specific constructions. In particular, we have to construct instances where the optimal sequence of edit distance operations has some exploitable structure, which is made difficult by the fact that we work over binary alphabet, so that in principle any two zeroes and any two ones can be matched. Interestingly, exactly this property of providing hardness already on the smallest-possible alphabet has enabled the use of our alignment gadget for LCS as a key technical tool to derive conditional hardness for practically relevant regimes for the RNA folding problem [Cha15].

Multivariate Fine-grained Complexity. Analyzing the time complexity of problems in the polynomial-time regime with respect to various input parameters has a long

[7] For a short discussion, see Section 3.3.

[8] Note that our general definition of Edit Distance includes LCS as a special case. Nevertheless, since the proof for LCS is more accessible, we chose to also include this construction for readers' convenience.

history, especially for the LCS problem, and recently received growing attention as a general paradigm [AVWW16; GMN15]. Our systematic approach of proving conditional lower bounds for all parameter settings strengthens this kind of analysis, since it allows to establish (near-)optimality and to uncover tractable parameter settings, for which improved algorithms can be found. To the best of our knowledge, our results are the first systematic conditional lower bounds for a polynomial-time problem with more than one parameter (in addition to the problem size).

The first challenge of our work is to uncover all relations between the studied parameters. Indeed, some parameter settings are contradictory because they violate a parameter relation. In this case, at most a finite number of instances satisfies the parameter setting, and LCS is trivial to solve. As we want to construct a family of instances for each non-trivial parameter setting, we have to find a complete list of parameter relations that hold up to constant factors. While some parameter relations can be found in the literature, a complete list was unknown. In particular, these relations allow us to judge for any asymptotic running times, like $\tilde{\mathcal{O}}(n+d)$ and $\tilde{\mathcal{O}}(n+\delta M/n)$, whether one is larger than the other on all instances or whether they are incomparable.

Our main challenge is to construct a family of hard instances for each non-trivial parameter setting. Here we start with the hard instances obtained by using the alignment gadget framework and carefully embed them into strings that satisfy any given parameter setting. Note that since the desired bound $\tilde{\mathcal{O}}(n+\min\{d,\delta m,\delta\Delta\})$ is a quite complex function of the parameters, our reductions are necessarily rather involved and we need to very carefully fine-tune our constructions.

One of the new ideas in our constructions is an alternative way of combining so-called (normalized) vector gadgets[9] that we use for the case of small δ. In this construction, picking any two vector gadgets in x and one in y yields an LCS without any deletions in the vector gadget of y, thus not increasing δ, which is important for keeping δ small. Picking one vector gadget in both x and y yields an LCS whose length depends on the orthogonality of the vectors. We then need to ensure that such "2vs1" and "1vs1" matchings always generate an LCS and at least one "1vs1" matchings appears in any LCS.

2.4 Notes

The contents of Chapters 4 and 7 and parts of Chapter 6 have already appeared in [BK15b] (with an extended version accessible at [BK15c]). The results of Chapter 5 and parts of Chapter 6 are based on the unpublished manuscript [BK16].

Independently of our work, quadratic SETH-hardness of LCS and DTW has been shown by Abboud, Backurs and Vassilevska Williams [ABVW15b]. Let us briefly compare our approaches. Our main technical contribution is the alignment gadget framework, which allows us to give shorter hardness proofs. The proofs of Abboud et al. are longer, in particular since they are using the lower bound for Levenshtein distance [BI15a], while our proofs are self-contained. The main technical contribution of Abboud et al., apart from careful reductions, seems to be that they reduce from a novel problem that they call Most-Orthogonal Vectors. Regarding the problem LCS, our hardness result is stronger, since we show hardness on binary strings, while Abboud et al. need alphabet size 7. Regarding DTW, we prove hardness of different special cases, as we consider DTW on

[9] Readers unfamiliar with this notion are referred to Section 3.3 for an introduction to the Orthogonal Vectors problem, whose quadratic SETH-hardness is well known. To obtain our hardness results, we reduce from Orthogonal Vectors by representing each vector by a (normalized) vector gadget and combining them to a final instance. Here, the orthogonality of two vectors is captured by the similarity of the corresponding vector gadgets.

one-dimensional curves over alphabets of size 5 (where the distance of two numbers is their absolute difference), while Abboud et al. consider DTW on strings over alphabets of size 5 (where the distance of two symbols is 1 or 0, depending on whether they are equal or not).

2.5 Organization

In Chapter 3, we introduce notation, prove several useful lemmas and discuss alternative assumptions to SETH that can be used to derive our results. We present our framework for obtaining quadratic lower bounds in Chapter 4: In Section 4.1, we define the alignment gadget and prove that any similarity measure admitting such a gadget (together with so called *coordinate values*) has a quadratic lower bound under SETH. We proceed to prove quadratic-time hardness of Dynamic Time Warping (Section 4.2), Longest Common Subsequence (Section 4.3) and Edit Distance (Section 4.4) by implementing an alignment gadget for each of these problems. Note that the construction for LCS is in fact subsumed by the alignment gadget for Edit Distance – however, since for the special case LCS the gadget (and its proof) is simpler and might be more accessible, we chose to provide it for the reader's convenience.

In Chapter 5, we focus on Longest Common Subsequence and study its *multivariate* complexity, going beyond the general quadratic worst-case SETH-hardness of $n^{2-o(1)}$. Since this a quite complex task, this chapter takes up a large fraction of this thesis.

Algorithmic results accompanying the lower bounds (showing that the obtained bounds are indeed tight up to lower order terms) are given in Chapter 6. In Chapter 7, we prove hardness of longest palindromic subsequence and longest tandem subsequence by reductions from LCS. Finally, we conclude Part I of this thesis with open problems and an outlook in Chapter 8.

Acknowledgments

The author wishes to thank Pawel Gawrychowski for valuable pointers and suggesting to study LTS, the anonymous reviewers of our FOCS 2015 submission for their helpful comments and Karl Bringmann for the fruitful and exciting collaboration on the results in this part.

Chapter 3

Preliminaries

In this chapter, we prepare notation, problem definitions and basic concepts used throughout Part I. In particular, we give a list of algorithmic problems considered in this part in Section 3.2 and discuss the hardness hypotheses used to derive our conditional lower bounds in Section 3.3. Section 3.4 briefly describes the model of computation we assume throughout this part and Section 3.5 introduces basic facts to ease later analysis of Edit Distance and its special case LCS.

3.1 Notation and Conventions

We write $[n] := \{1, \dots, n\}$. For a sequence (i.e., string or curve) x, we denote its length by $|x|$, write $x[i]$ for its i-th position, and denote the subsequence from position i to position j by $x[i..j]$. We call its entries *symbols* or sometimes, if it is a string, *characters*. For a sequence x of length n, let $\mathrm{rev}(x) := x[n]\, x[n-1] \dots x[1]$ denote its reversed sequence. For two sequences x and y, we denote their concatenation by $x \circ y = xy$ and define, for any $\ell \geq 0$, the ℓ-fold repetition of x by $x^\ell := \bigcirc_{i=1}^{\ell} x$.

If string x is defined over alphabet Σ, we denote the number of occurrences of symbol $\sigma \in \Sigma$ in x by $\#_\sigma(x)$. We say that z is a subsequence of x of length ℓ, if $|z| = \ell$ and there are $1 \leq i_1 < \dots < i_\ell \leq |x|$ such that $x[i_k] = z[k]$ for all $1 \leq k \leq \ell$.

To state running times, we use $\tilde{\mathcal{O}}(\cdot)$ to hide polylogarithmic factors in n. For clarity of presentation, we sometimes state running times of the form $\mathcal{O}(f(n) \cdot g(n))$, where $f(n)$ possibly attains non-positive values (e.g, $f(n) = \log(n/m)$ with $n \geq m$); such time bounds are to be read as $\mathcal{O}(\max\{f(n), 1\} \cdot g(n))$. Whenever we write a chain of equalities $f(n) = \mathcal{O}(g(n)) = \mathcal{O}(h(n))$, the intended meaning is $f(n) = \mathcal{O}(g(n))$ and $g(n) = \mathcal{O}(h(n))$.

3.2 Problems in Part I

A *traversal* of two sequences x and y of length n and m, respectively, is a sequence of pairs $((a_1, b_1), \dots, (a_t, b_t))$ with $t \in \mathbb{N}$ satisfying (1) $(a_1, b_1) = (1, 1)$, (2) $(a_t, b_t) = (n, m)$, and (3) (a_{i+1}, b_{i+1}) is either of $(a_i + 1, b_i)$, $(a_i, b_i + 1)$, or $(a_i + 1, b_i + 1)$ for all $1 \leq i < t$.

Edit Distance. Let x, y be strings over an alphabet Σ of length n, m ($n \geq m$), respectively. For a traversal $T = ((a_1, b_1), \dots, (a_t, b_t))$ of x, y we say that its i-th operation, $1 \leq i < t$, is (1) a *deletion in x* if $(a_{i+1}, b_{i+1}) = (a_i + 1, b_i)$, (2) a *deletion in y* if $(a_{i+1}, b_{i+1}) = (a_i, b_i + 1)$, (3) a *matching* if $(a_{i+1}, b_{i+1}) = (a_i + 1, b_i + 1)$ and $x[a_i] = y[b_i]$, or (4) a *substitution* if $(a_{i+1}, b_{i+1}) = (a_i + 1, b_i + 1)$ and $x[a_i] \neq y[b_i]$. These four operations incur costs of $c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}$, and c_{subst}, respectively. We always assume that these costs are rational constants, so that we can ignore representation issues. The cost $\boldsymbol{\delta}_{\text{Edit}}(T)$ of a traversal T is the total cost of all its operations. The edit distance $\boldsymbol{\delta}_{\text{Edit}}(x, y)$ is the

minimal cost of any traversal of x, y. We write $\text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$ for the problem of computing the edit distance of two given strings with costs $c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}$, and c_{subst}. We write $\text{Edit}(c_{\text{subst}})$ as a shorthand for $\text{Edit}(1, 1, 0, c_{\text{subst}})$. Note that for these problems, the costs of all four operations are constant, i.e., they stay fixed with growing n, m. We will mostly consider Edit Distance over binary strings, i.e., we set $\Sigma = \{0, 1\}$.

We obtain the **Levenshtein distance** by setting $c_{\text{match}} = 0$ and $c_{\text{del-x}} = c_{\text{del-y}} = c_{\text{subst}} = 1$. By setting $c_{\text{match}} = 0$, $c_{\text{del-x}} = c_{\text{del-y}} = 1$, and $c_{\text{subst}} = 2$, we obtain effectively the Longest Common Subsequence problem, see below. One obtains more variants for other cost choices, e.g., for aligning DNA sequences a classic choice is $c_{\text{match}} = -1$, $c_{\text{subst}} = 1$, and $c_{\text{del-x}} = c_{\text{del-y}} = 2$, see, e.g., [SM97].

Longest Common Subsequence (LCS). For strings x,y and z, we say that z is a longest common subsequence (LCS) of x and y if it is a subsequence of both x and y that has maximal length $|z|$. By $\text{LCS}(x, y)$, we denote an arbitrary LCS of x and y and write $L(x, y) := |\text{LCS}(x, y)|$ for its length. It is easy to see that computing $L(x, y)$ is equivalent to computing $\text{Edit}(2)$, since under the operation costs $c_{\text{match}} = 0$, $c_{\text{del-x}} = c_{\text{del-y}} = 1$ and $c_{\text{subst}} = 2$ of $\text{Edit}(2)$, we have $\boldsymbol{\delta}_{\text{Edit}}(x, y) = |x| + |y| - 2L(x, y) =: \boldsymbol{\delta}_{\text{LCS}}(x, y)$.

Dynamic Time Warping (DTW). Let (M, d) be any metric space. Let x, y be curves, i.e., sequences over M of length n, m ($n \geq m$), respectively. The cost $\boldsymbol{\delta}_{\text{DTW}}(T)$ of a traversal $T = ((a_1, b_1), \ldots, (a_t, b_t))$ is $\sum_{i=1}^{t} d(x[a_i], y[b_i])$. The dynamic time warping distance $\boldsymbol{\delta}_{\text{DTW}}(x, y)$ is the minimal cost of any traversal of x and y. We obtain the special case of *dynamic time warping on one-dimensional curves* by setting $M = \mathbb{R}$ and $d(a, b) := |a - b|$ for any $a, b \in \mathbb{R}$.

Longest Palindromic Subsequence (LPS). For a string x, a palindromic subsequence (also known as symmetric subsequence) is a subsequence z of x such that z equals its reverse $\text{rev}(z)$. We denote the length of the longest palindromic subsequence (LPS) of x by $P(x)$.

Longest Tandem Subsequence (LTS). For a string x, we call a subsequence z of x a tandem sequence, if z can be written as the concatenation $z = yy$ of a string y with itself. We let $T(x)$ denote the length of the longest tandem subsequence (LTS) of x.

3.3 Hardness Hypotheses

Recall the Strong Exponential Time Hypothesis (SETH), introduced by Impagliazzo and Paturi [IP01; IPZ01], that asserts that satisfiability has no algorithms that are much faster than exhaustive search. It forms the basis of many conditional lower bounds for NP-hard as well as polynomial-time problems, see, e.g., [VW15; LMS11].

Hypothesis 3.3.1 (Strong Exponential Time Hypothesis (SETH))**.** *For no $\varepsilon > 0$, k-SAT can be solved in time $\mathcal{O}((2 - \varepsilon)^n)$ for all $k \geq 3$.*

Effectively all known SETH-based lower bounds for polynomial-time problems use reductions via the *Orthogonal Vectors problem* (OV): Given sets $\mathcal{A}, \mathcal{B} \subseteq \{0, 1\}^D$ of size $|\mathcal{A}| = n$, $|\mathcal{B}| = m$, determine whether there exist $a \in \mathcal{A}$, $b \in \mathcal{B}$ with $\sum_{i=1}^{D} a[i] \cdot b[i] = 0$ (which we denote by $\langle a, b \rangle = 0$). Simple algorithms solve OV in time $\mathcal{O}(2^D(n + m))$ and $\mathcal{O}(nmD)$. The fastest known algorithm for $D = c(n) \log n$ runs in time $n^{2-1/\mathcal{O}(\log c(n))}$

(when $n = m$) [AWY15], which is only slightly subquadratic for $D \gg \log n$. This has led to the following reasonable hypothesis.

Hypothesis 3.3.2 (Orthogonal Vectors Hypothesis (OVH))**.** *For no $\varepsilon > 0$,* OV *restricted to $n = |\mathcal{A}| = |\mathcal{B}|$ and $D = n^{o(1)}$ can be solved in time $\mathcal{O}(n^{2-\varepsilon})$.*

A well-known reduction by Williams [Wil05] shows that SETH implies OVH. Thus, OVH is the weaker assumption and any OVH-based lower bound also implies a SETH-based lower bound. The lower bounds in this thesis do not only hold assuming SETH, but even assuming the weaker OVH.

Lemma 3.3.3 ([Wil05])**.** SETH *implies* OVH*.*

Proof sketch. The statement follows from a simple reduction that introduced the so called split-and-list technique: Let ϕ be an arbitrary k-CNF formula with clauses $C_1, \ldots, C_M$ and Boolean variables $x_1, \ldots, x_N$ and assume for simplicity that N is even. Let L contain all partial assignments $\ell : [N/2] \to \{0,1\}$ to the variables $x_1, \ldots, x_{N/2}$ and similarly, let R contain all partial assignments $r : [N/2] \to \{0,1\}$ to the variables $x_{N/2+1}, \ldots, x_N$. Then for each assignment $\ell_i \in L$, we define a vector $a_i \in \{0,1\}^M$, where for all $k \in [M]$, we set $a_i[k] = 0$ if and only if the partial assignment ℓ_i of the variables $x_1, \ldots, x_{N/2}$ already satisfies the clause C_k, i.e., there is some variable x_j with $j \in [N/2]$ such that C_k contains the literal x_j if $\ell_i(j) = 1$ or the negated literal $\overline{x_j}$ if $\ell_i(j) = 0$. Symmetrically, we define a vector b_j for each partial assignment $r_j \in R$. Defining $\mathcal{A}$ as the set of all constructed a_i and $\mathcal{B}$ as the set of all constructed b_j, we obtain an OV instance, finishing the definition of the reduction.

It is easy to see that the obtained OV instance $\mathcal{A}, \mathcal{B}$ contains an orthogonal pair a_i, b_j if and only if ϕ is satisfiable: There is a trivial one-to-one correspondence of assignments $s : [N] \to \{0,1\}$ and pairs of partial assignments (ℓ_i, r_j). The corresponding vectors a_i, b_j are orthogonal if and only if (ℓ_i, r_j) (and hence s) is a satisfying assignment, since $\sum_{k=1}^{M} a_i[k] \cdot b_j[k] = 0$ holds if and only if for all k, at least one of $a_i[k]$ and $b_j[k]$ are zero, which captures exactly the fact that each clause C_k has to be satisfied either by some variable(s) among $x_1, \ldots, x_{N/2}$ or by some variable(s) among $x_{N/2}, \ldots, x_N$ (or by some variables among both of the sets).

Note that $n := |\mathcal{A}| = |\mathcal{B}| = 2^{N/2}$ and hence $D = M \leq N^k = \mathcal{O}(\log^k n) = n^{o(1)}$. Since the reduction runs in linear time $\mathcal{O}((|\mathcal{A}| + |\mathcal{B}|)D) = \mathcal{O}(2^{N/2+o(1)})$, we conclude that an $\mathcal{O}(n^{2-\varepsilon})$-time algorithm for any constant $\varepsilon > 0$ would yield an algorithm for k-SAT with running time $\mathcal{O}(2^{N(1-\varepsilon/2)+o(1)})$. Since k was chosen arbitrarily, this would refute SETH. □

We remark that using the sparsification lemma [IPZ01], it is easy to show that SETH also implies a stronger Orthogonal Vectors Hypothesis which asserts that there is no $\varepsilon > 0$ such that OV with $D \leq c \log n$ can be solved in time $\mathcal{O}(n^{2-\varepsilon})$ for all $c > 0$. However, we will not use this stronger assumption.

For convenience, we also work with the following variant of OVH. It provides an equivalent formulation for giving conditional lower bounds of the form $\Omega((|\mathcal{A}| \cdot |\mathcal{B}|)^{1-\varepsilon})$ also when $|\mathcal{A}|, |\mathcal{B}|$ have an arbitrary fixed polynomial dependence on a number n (note that here, n is not necessarily the problem size $(|\mathcal{A}| + |\mathcal{B}|) \cdot D$; however, the problem size has a fixed polynomial dependence on n).

Hypothesis 3.3.4 (Unbalanced Orthogonal Vectors Hypothesis (UOVH))**.** *For any $\varepsilon > 0$, $\alpha, \beta \in (0,1]$, and computable functions $f(n) = n^{\alpha - o(1)}, g(n) = n^{\beta - o(1)}$, the following problem cannot be solved in time $\mathcal{O}(n^{\alpha+\beta-\varepsilon})$: Given a number n, solve a given* OV *instance with $D = n^{o(1)}$, $|\mathcal{A}| = f(n)$ and $|\mathcal{B}| = g(n)$.*

Lemma 3.3.5 (Essentially folklore)**.** UOVH *is equivalent to* OVH.

Proof. Clearly, UOVH implies OVH (using $\alpha = \beta = 1, f(n) = g(n) = n$). For the other direction, assume that UOVH fails and let $\alpha, \beta \in (0, 1]$, $f(n) = n^{\alpha - o(1)}$, and $g(n) = n^{\beta - o(1)}$ be such that OV with $D = n^{o(1)}$ and $|\mathcal{A}| = f(n)$ and $|\mathcal{B}| = g(n)$ can be solved in time $\mathcal{O}(n^{\alpha+\beta-\varepsilon})$ for some constant $\varepsilon > 0$. Consider an arbitrary OV instance $\mathcal{A}, \mathcal{B} \subseteq \{0,1\}^D$ with $|\mathcal{A}| = |\mathcal{B}| = n$ and $D = n^{o(1)}$. We partition $\mathcal{A}$ into $s := \lceil \frac{n}{f(n)} \rceil$ sets $\mathcal{A}_1, \ldots, \mathcal{A}_s$ of size $f(n)$ and $\mathcal{B}$ into $t := \lceil \frac{n}{g(n)} \rceil$ sets $\mathcal{B}_1, \ldots, \mathcal{B}_t$ of size $g(n)$ (note that the last set of such a partition might have strictly less elements, but can safely be filled up using all-ones vectors). By assumption, we can solve each OV instance $\mathcal{A}_i, \mathcal{B}_j$ in time $\mathcal{O}(n^{\alpha+\beta-\varepsilon})$. Since there exist $a \in \mathcal{A}, b \in \mathcal{B}$ with $\langle a, b \rangle = 0$ if and only if there exist $a \in \mathcal{A}_i, b \in \mathcal{B}_j$ with $\langle a_i, b_j \rangle = 0$ for some $i \in [s], j \in [t]$, we can decide the instance $\mathcal{A}, \mathcal{B}$ by sequentially deciding the $s \cdot t = \mathcal{O}(n^{2-(\alpha+\beta)+o(1)})$ OV instances $\mathcal{A}_i, \mathcal{B}_j$. This takes total time $\mathcal{O}(s \cdot t \cdot n^{\alpha+\beta-\varepsilon}) = \mathcal{O}(n^{2-\varepsilon'})$ for any $\varepsilon' < \varepsilon$, which contradicts OVH and thus proves the claim. □

We remark that OVH is implicit in many other SETH-based lower bounds. A weaker version of it (called Most-Orthogonal Vectors) has been used in [ABVW15b] to derive their results.

Finally, let us briefly discuss recent work on stronger and weaker variants of SETH that investigates their implications in the regime of polynomial-time problems. Abboud et al. [Abb+16a] reduce the satisfiability problem on (non-deterministic) branching programs to the problems captured in our framework to obtain stronger hardness results. Intuitively, a non-deterministic branching program is a directed graph consisting of a set of L layers of width W, where each node represents one of n Boolean variables $x_1, \ldots, x_n$ and each edge is labeled with 0 or 1. The label ℓ of an edge corresponds to a constraint $x_i = \ell$, where x_i is the label of the node this edge leaves. There are two distinguished nodes, a start node in the first layer and an accept node in the last layer. The satisfiability problem asks whether there is a Boolean assignment to the variables $x_1, \ldots, x_n$ such that a constraint-respecting path from the start node to the accept node exists. Abboud et al. show that a strongly subquadratic algorithm for any problem admitting an alignment gadget would imply an $\mathcal{O}((2-\varepsilon)^n)$-time algorithm (for some constant $\varepsilon > 0$) for the satisfiability problem on non-deterministic branching programs of constant width W and length $L = 2^{o(n)}$. Note that this class of branching programs captures even NC circuits, which are capable of, e.g., computing determinants or basic cryptographic primitives. This bases the conditional lower bounds of our framework on a much weaker assumption than SETH.

In the opposite direction, Carmosino et al. [Car+16] investigate a stronger, non-deterministic variant of SETH and prove that it cannot be refuted without proving new non-trivial circuit lower bounds. Translated to the problems considered in this part, their results imply that even finding strongly subquadratic *co-nondeterministic* algorithms for LCS, DTW and the Levenshtein distance requires substantial progress in circuit complexity.

3.4 Models of Computation

Unless stated otherwise, we assume the *unit-cost Word RAM* model of computation, which we briefly describe here (for a more detailed introduction, see, e.g., [Hag98]). In this model, the memory consists of M words of word size w and all elementary operations (i.e., basic arithmetic and logic operations on a word of w bits, as well as dereferencing a

memory cell) can be executed in constant time, where random access to the memory words is allowed. We assume that $M \geq n$ (to store the input, where n is the problem size) and $w = \Theta(\log n)$, since $w \geq \log M$ is needed to ensure that each memory word can indeed be addressed and the corresponding upper bound prohibits, e.g, arithmetics on superpolynomially large integers in unit cost per operation.

A more restrictive model of computation (used in Section 6.1) is the abstract *pointer machine* [Tar79; BG92]. Intuitively, it disallows pointer arithmetic by classifying memory cells (words) into data cells or pointer cells, and enabling computation only on data cells, not pointer cells. Thus, instead of having random access, this model resembles the power of indirect accessing by following pointers.

We remark here that the reductions we provide in this thesis are linear-time computable using only very elementary computational primitives and that the hardness results (and linear running time bounds) transfer also to less powerful models of computation than the Word RAM. The above choice of models is more relevant for the algorithmic results stated in Chapter 5 and Chapter 6, where it may affect logarithmic factors in the running times.

3.5 Basic Facts for Edit Distance and LCS

In this section, we collect technical tools for the analysis of Edit Distance and LCS to ease the analysis in later chapters. In particular, throughout this section we analyze the variant $\mathrm{Edit}(c_{\mathrm{subst}}) = \mathrm{Edit}(1, 1, 0, c_{\mathrm{subst}})$ with $0 < c_{\mathrm{subst}} \leq 2$, in which deletions in both strings incur unit cost, matches are free of cost and each substitution costs c_{subst}. This is justified by our classification result in Section 4.4.1 which reduces every non-trivial variant of the edit distance to this case. Recall that LCS is equivalent to the special case Edit(2).

Fact 3.5.1 (Optimal Partitions)**.** *Let x and $y_1, \ldots, y_k$ be arbitrary strings. Set $y = y_1 \ldots y_k$. Then we have*

$$\boldsymbol{\delta}_{\mathrm{Edit}}(x, y) = \min_{x(y_1),\ldots,x(y_k)} \sum_{j=1}^{k} \boldsymbol{\delta}_{\mathrm{Edit}}(x(y_j), y_j),$$

where $x(y_1), \ldots, x(y_k)$ ranges over all ordered partitions *of x into k substrings, i.e., $x(y_1) = x[i_0 + 1..i_1], x(y_2) = x[i_1 + 1..i_2], \ldots, x(y_k) = x[i_{k-1} + 1..i_k]$ for any $0 = i_0 \leq i_1 \leq \ldots \leq i_k = |x|$.*

Proof. For any ordered partition, the substrings $x(y_j)$ are disjoint and ordered along x, so we can concatenate (optimal) traversals of $(x(y_j), y_j)$, $j \in [k]$, to form a traversal of (x, y). This shows $\boldsymbol{\delta}_{\mathrm{Edit}}(x, y) \leq \sum_{j=1}^{k} \boldsymbol{\delta}_{\mathrm{Edit}}(x(y_j), y_j)$.

Now let T be an optimal traversal of (x, y). Let J_j be the indices in x that appear in a matching or substitution operation with symbols in y_j. Note that these sets are ordered, in the sense that for any $i \in J_j$ and $i' \in J_{j'}$ with $j < j'$ we have $i < i'$. This allows to find an ordered partition $x(y_1), \ldots, x(y_k)$ of x such that $x(y_j)$ contains the indices J_j for any j. Let us denote the total cost of the substitutions involving y_j by s_j. Since traversal T deletes $|y_j| - |J_j|$ symbols in y_j and $|x(y_j)| - |J_j|$ symbols in $x(y_j)$, we have $\boldsymbol{\delta}_{\mathrm{Edit}}(T) = \sum_{j=1}^{k} |y_j| + |x(y_j)| - 2|J_j| + s_j$. Clearly, we can construct a traversal of $(x(y_j), y_j)$ that follows the matchings and substitutions in J_j and deletes all other symbols, showing $\boldsymbol{\delta}_{\mathrm{Edit}}(x(y_j), y_j) \leq |y_j| + |x(y_j)| - 2|J_j| + s_j$. By optimality of T, we obtain $\boldsymbol{\delta}_{\mathrm{Edit}}(x, y) \geq \sum_{i=1}^{k} \boldsymbol{\delta}_{\mathrm{Edit}}(x(y_j), y_j)$. □

Fact 3.5.2. *Let x, y, z be arbitrary strings and $\ell, k \in \mathbb{N}_0$. Then we have*

(i) $\boldsymbol{\delta}_{\mathrm{Edit}}(1^k x, 1^k y) = \boldsymbol{\delta}_{\mathrm{Edit}}(x, y)$,

(ii) $\boldsymbol{\delta}_{\mathrm{Edit}}(x, y) \geq \big||x| - |y|\big|$, *and*

(iii) $|\boldsymbol{\delta}_{\mathrm{Edit}}(xz, y) - \boldsymbol{\delta}_{\mathrm{Edit}}(x, y)| \leq |z|$.

We obtain symmetric statements of (i) by replacing all 1's by 0's and by reversing all involved strings.

Proof. We show (i) for $k = 1$, then the general statement follows by induction. Consider an optimal traversal T of $1x, 1y$. If both "1"s are deleted in T, then we can instead match them and improve T, contradicting optimality. If exactly one "1" is matched or substituted, then the other "1" is deleted, so we may instead match the two "1"s without increasing cost. Thus, without loss of generality an optimal traversal of $(1x, 1y)$ matches the two "1"s.

For (ii), note that matchings and substitutions touch as many symbols in x as in y. Hence, there have to be at least $|x| - |y|$ deletions in x and at least $|y| - |x|$ deletions in y.

For (iii), taking an optimal traversal of (x, y) and appending $|z|$ deletions of the symbols in z shows that $\boldsymbol{\delta}_{\mathrm{Edit}}(xz, y) \leq \boldsymbol{\delta}_{\mathrm{Edit}}(x, y) + |z|$. For the other direction, consider an optimal traversal T of (xz, y). Replace any matching or substitution of a symbol in z with a symbol $y[j]$ in y by a deletion of $y[j]$. Additionally, remove every deletion of a symbol in z. This results in a traversal T' of (x, y) with cost at most $\boldsymbol{\delta}_{\mathrm{Edit}}(xz, y) + |z|$, as we introduced at most $|z|$ deletions in y. This proves the desired inequality $\boldsymbol{\delta}_{\mathrm{Edit}}(x, y) \leq \boldsymbol{\delta}_{\mathrm{Edit}}(xz, y) + |z|$. □

For the special case of LCS, we use the following similar observations.

Fact 3.5.3. *Let z, w be arbitrary strings and $\ell, k \in \mathbb{N}_0$. Then we have*

(i) $\boldsymbol{\delta}_{\mathrm{LCS}}(1^k z, 1^k w) = \boldsymbol{\delta}_{\mathrm{LCS}}(z, w)$,

(ii) $\boldsymbol{\delta}_{\mathrm{LCS}}(1^k z, w) \geq \boldsymbol{\delta}_{\mathrm{LCS}}(z, w) - k$, *and*

(iii) $\boldsymbol{\delta}_{\mathrm{LCS}}(0^\ell z, 1^k w) \geq \min\{k, \boldsymbol{\delta}_{\mathrm{LCS}}(z, 1^k w) + \ell\}$.

We obtain symmetric statements by flipping all bits and by reversing all involved strings.

Proof. Statement (i) is a special case of Fact 3.5.2(i). Statement (ii) follows from Fact 3.5.2(iii).

For (iii), fix a LCS s of $(0^\ell z, 1^k w)$. If s starts with a "0", then it does not contain the leading 1^k of the second argument, leaving at least k symbols unmatched, so that $\boldsymbol{\delta}_{\mathrm{LCS}}(0^\ell z, 1^k w) \geq k$. Otherwise, if s starts with a "1", then it does not contain the leading 0^ℓ of the first argument, so that $L(0^\ell z, 1^k w) = L(z, 1^k w)$. Then we have $\boldsymbol{\delta}_{\mathrm{LCS}}(0^\ell z, 1^k w) = |0^\ell z| + |1^k w| - 2L(0^\ell z, 1^k w) \geq |z| + |1^k w| - 2L(z, 1^k w) = \boldsymbol{\delta}_{\mathrm{LCS}}(z, 1^k w)$. □

Chapter 4

Framework

In this chapter, we provide a general way to derive quadratic conditional lower bounds from OVH/SETH, inspired by the approach of [BI15a; Bri14]. In Section 4.1, we define the *alignment gadget*, which provides the basis of the lower bound, and show how to reduce Orthogonal Vectors to computing any similarity measure admitting such an alignment gadget (together with so called *coordinate values*). We then give quadratic conditional lower bounds for Dynamic Time Warping (Section 4.2), Longest Common Subsequence (Section 4.3) and Edit Distance (Section 4.4) by proving alignment gadgets for each of these problems.

4.1 Framework Proof

We consider a similarity (or distance) measure $\boldsymbol{\delta} : \mathcal{I} \times \mathcal{I} \to \mathbb{N}_0$, where $\mathcal{I}$ denotes the set of inputs, e.g., all binary strings or all one-dimensional curves. By a reduction from Orthogonal Vectors, we prove that this similarity measure cannot be computed in strongly subquadratic time unless SETH fails if $\boldsymbol{\delta}$ admits a gadget that allows us to realize, in a certain sense, alignments of inputs $x_1, \ldots, x_n \in \mathcal{I}$ and $y_1, \ldots, y_m \in \mathcal{I}$. To formally state the requirement, we start by introducing the following notions.

Types. For each similarity measure, we define an abstract *type* of a sequence $x \in \mathcal{I}$. This type simplifies the construction of the gadget we aim to construct. Typically, we choose the type to be the length of the sequence and the sum of its entries, i.e., $\text{type}(x) := (|x|, \sum_i x[i])$ (where for binary strings, $\sum_k x[k]$ is to be interpreted as the number of ones in x). This definition can be customized to work for the similarity measure under consideration.[1] We define $\mathcal{I}_t := \{x \in \mathcal{I} \mid \text{type}(x) = t\}$ as the set of inputs of type t.

Alignments. Let $n \geq m$. A *(partial) alignment* is a set $\Lambda = \{(i_1, j_1), \ldots, (i_k, j_k)\}$ with $0 \leq k \leq m$ such that $1 \leq i_1 < \ldots < i_k \leq n$ and $1 \leq j_1 < \ldots < j_k \leq m$. We say that $(i, j) \in \Lambda$ are *aligned*. Any $i \in [n]$ or $j \in [m]$ that is not contained in any pair in Λ is called *unaligned*. We denote the set of all partial alignments (with respect to n, m) by $\boldsymbol{\Lambda}_{n,m}$.

We call the partial alignment $\{(\Delta + 1, 1), \ldots, (\Delta + m, m)\}$, with $0 \leq \Delta \leq n - m$, a *structured alignment*. We denote the set of all structured alignments by $\mathcal{S}_{n,m}$.

For any $x_1, \ldots, x_n \in \mathcal{I}$ and $y_1, \ldots, y_m \in \mathcal{I}$, we define the *cost* of alignment $\Lambda \in \boldsymbol{\Lambda}_{n,m}$ by

$$\mathbf{c}(\Lambda) = \mathbf{c}^{x_1,\ldots,x_n}_{y_1,\ldots,y_m}(\Lambda) := \sum_{(i,j)\in\Lambda} \boldsymbol{\delta}(x_i, y_j) + (m - |\Lambda|) \max_{i,j} \boldsymbol{\delta}(x_i, y_j).$$

[1] For DTW, we indeed augment the type definition by the maximum entry within the sequence.

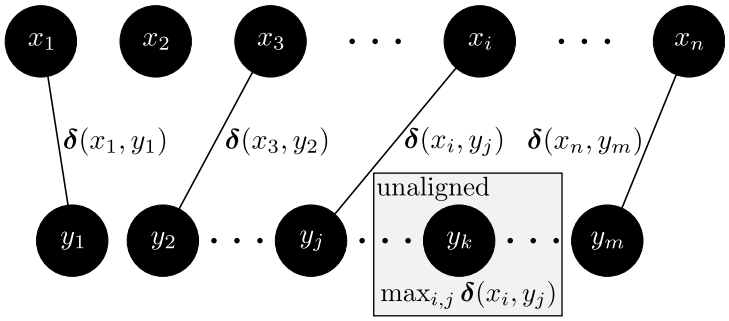


FIGURE 4.1: Cost $\mathbf{c}(\Lambda)$ of a partial alignment $\Lambda \in \mathbf{\Lambda}_{n,m}$.

In other words, for any $j \in [m]$ which is aligned to some i we "pay" the distance $\boldsymbol{\delta}(x_i, y_j)$, while for any unaligned j we "pay" the maximal distance of any $(x_{i'}, y_{j'})$ (note that there are $m - |\Lambda|$ unaligned $j \in [m]$). This means that we incur punishment for any unaligned j (see Figure 4.1 and 4.2 for an illustrations of alignments and their costs).

Alignment Gadget. We start with some intuition. Consider the problem of computing the value $\min_{\Lambda \in \mathcal{S}_{n,m}} \mathbf{c}(\Lambda)$. This can be solved in time $\mathcal{O}(nm)$ if each $\boldsymbol{\delta}(x_i, y_j)$ can be evaluated in constant time, since $|\mathcal{S}_{n,m}| = \mathcal{O}(n)$ and evaluating $\mathbf{c}(\Lambda)$ amounts to computing m values $\boldsymbol{\delta}(x_i, y_j)$. Moreover, intuitively it should not be possible to compute this value in strongly subquadratic time. We will show that in some sense it is even hard to compute, in strongly subquadratic time, *any value* v with

$$\min_{\Lambda \in \mathbf{\Lambda}_{n,m}} \mathbf{c}(\Lambda) \leq v \leq \min_{\Lambda \in \mathcal{S}_{n,m}} \mathbf{c}(\Lambda). \tag{4.1}$$

Now, an alignment gadget is simply a pair of instances (x, y) such that from $\boldsymbol{\delta}(x, y)$ we can infer[2] a value v as above. The main reason to relax our goal from computing $\min_{\Lambda \in \mathcal{S}_{n,m}} \mathbf{c}(\Lambda)$ to satisfying (4.1) is that this makes constructing alignment gadgets much easier. Note that for the alignment gadget (x, y) computing $\boldsymbol{\delta}(x, y)$ is as hard as computing $\min_{\Lambda \in \mathcal{S}_{n,m}} \mathbf{c}(\Lambda)$ (in an approximate sense as given by (4.1)), which we argued above should take quadratic time. This informal discussion motivates the following definition.

Definition 4.1.1. *The similarity measure $\boldsymbol{\delta}$ admits an* alignment gadget, *if the following conditions hold: Given instances $x_1, \ldots, x_n \in \mathcal{I}_{t_X}$, $y_1, \ldots, y_m \in \mathcal{I}_{t_Y}$ with $m \leq n$ and types $t_X = (\ell_X, s_X), t_Y = (\ell_Y, s_Y)$, we can construct new instances $x = \mathrm{GA}_X^{m,t_Y}(x_1, \ldots, x_n)$ and $y = \mathrm{GA}_Y^{n,t_X}(y_1, \ldots, y_m)$ and $C \in \mathbb{Z}$ such that*

$$\min_{\Lambda \in \mathbf{\Lambda}_{n,m}} \mathbf{c}(\Lambda) \leq \boldsymbol{\delta}(x, y) - C \leq \min_{\Lambda \in \mathcal{S}_{n,m}} \mathbf{c}(\Lambda). \tag{4.2}$$

Moreover, $\mathrm{type}(x)$ *and* $\mathrm{type}(y)$ *only depend on n, m, t_X, and t_Y. Finally, this construction runs in time $\mathcal{O}((n+m)(\ell_X + \ell_Y))$.*

If the construction additionally fulfills $|x| = \mathcal{O}(n(\ell_X + \ell_Y))$ and $|y| = \mathcal{O}(m(\ell_X + \ell_Y))$, we say that $\boldsymbol{\delta}$ admits an unbalanced alignment gadget*.*

Note that the types serve the purpose of simplifying the algorithmic problem in the above definition by restricting the inputs to equi-typed objects. If we can construct

[2]For us, "infer" will simply mean that $v = \boldsymbol{\delta}(x, y) - C$ for an appropriate C. This could be easily generalized to more general correspondences, if needed for a particular application.

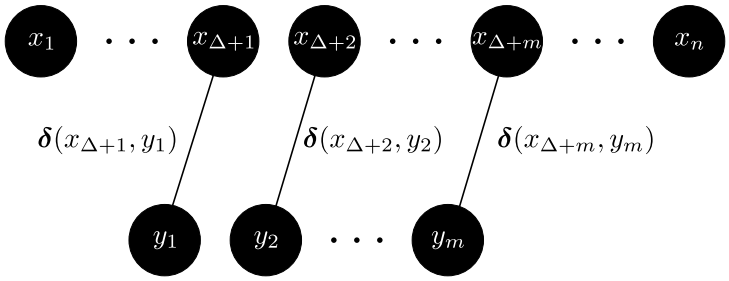


FIGURE 4.2: Cost $\mathbf{c}(\Lambda)$ of a structured alignment $\Lambda \in \mathcal{S}_{n,m}$.

suitable x and y for *arbitrary* inputs $x_1, \dots, x_n$ and $y_1, \dots, y_m$, then we may completely disregard types.

Definition 4.1.2. *The similarity measure $\boldsymbol{\delta}$ admits* coordinate values*, if there exist $\mathbf{0}_{\mathrm{X}}, \mathbf{0}_{\mathrm{Y}}, \mathbf{1}_{\mathrm{X}}, \mathbf{1}_{\mathrm{Y}} \in \mathcal{I}$ satisfying*

$$\boldsymbol{\delta}(\mathbf{1}_{\mathrm{X}}, \mathbf{1}_{\mathrm{Y}}) > \boldsymbol{\delta}(\mathbf{0}_{\mathrm{X}}, \mathbf{1}_{\mathrm{Y}}) = \boldsymbol{\delta}(\mathbf{0}_{\mathrm{X}}, \mathbf{0}_{\mathrm{Y}}) = \boldsymbol{\delta}(\mathbf{1}_{\mathrm{X}}, \mathbf{0}_{\mathrm{Y}}),$$

and moreover, $\mathrm{type}(\mathbf{0}_{\mathrm{X}}) = \mathrm{type}(\mathbf{1}_{\mathrm{X}})$ *and* $\mathrm{type}(\mathbf{0}_{\mathrm{Y}}) = \mathrm{type}(\mathbf{1}_{\mathrm{Y}})$.

The alignment gadget and the coordinate values as defined above provide the basis for our framework, which we can finally formalize.

Theorem 4.1.3. *Let $\boldsymbol{\delta}$ be a similarity measure admitting an alignment gadget as well as coordinate values and consider the problem of computing $\boldsymbol{\delta}(x, y)$ with $|x| \leq n$, $|y| \leq m$, and $m \leq n$. For no $\varepsilon > 0$, this problem can be solved in time $\mathcal{O}(m^{2-\varepsilon})$ unless OVH fails. If $\boldsymbol{\delta}$ even admits an unbalanced alignment gadget, then for no $\varepsilon > 0$, this problem can be solved in time $\mathcal{O}((nm)^{1-\varepsilon})$, unless (U)OVH fails. Both statements hold restricted to $n^{\alpha-o(1)} \leq m \leq n^{\alpha+o(1)}$ for any $0 < \alpha \leq 1$.*

In the remainder of this section, we present a reduction from OV to the problem of computing $\boldsymbol{\delta}$, thus proving Theorem 4.1.3. This uses constructions and arguments similar to [Bri14; BI15a]. Consider an instance $a_1, \dots, a_n \in \{0,1\}^d$ and $b_1, \dots, b_m \in \{0,1\}^d$ of OV, $n \geq m$. We construct $x, y \in \mathcal{I}$ and $\rho \in \mathbb{N}_0$ such that $\boldsymbol{\delta}(x, y) \leq \rho$ if and only if there are $i \in [n]$ and $j \in [m]$ with $\langle a_i, b_j \rangle = 0$. To this end, let $a_i[k]$ denote the k-th component of a_i. For all $i \in [n]$ and $j \in [m]$, we construct *coordinate gadgets* as follows,

$$\mathrm{CG}(a_i, k) := \begin{cases} \mathbf{0}_{\mathrm{X}} & \text{if } a_i[k] = 0 \\ \mathbf{1}_{\mathrm{X}} & \text{if } a_i[k] = 1 \end{cases} \quad 1 \leq k \leq d, \qquad \mathrm{CG}(a_i, d+1) := \mathbf{0}_{\mathrm{X}},$$

$$\mathrm{CG}(b_j, k) := \begin{cases} \mathbf{0}_{\mathrm{Y}} & \text{if } b_j[k] = 0 \\ \mathbf{1}_{\mathrm{Y}} & \text{if } b_j[k] = 1 \end{cases} \quad 1 \leq k \leq d, \qquad \mathrm{CG}(b_j, d+1) := \mathbf{1}_{\mathrm{Y}}.$$

Note that $\mathrm{type}(\mathrm{CG}(a_i, 1)) = \cdots = \mathrm{type}(\mathrm{CG}(a_i, d+1)) =: t_{\mathrm{X}}$ and $\mathrm{type}(\mathrm{CG}(b_j, 1)) = \cdots = \mathrm{type}(\mathrm{CG}(b_j, d+1)) =: t_{\mathrm{Y}}$ by definition of coordinate values. This allows us to use the alignment gadget to obtain the following *vector gadgets*

$$\begin{aligned} \mathrm{VG}(a_i) &:= \mathrm{GA}_{\mathrm{X}}^{d+1,t_{\mathrm{Y}}}(\mathrm{CG}(a_i, 1), \dots, \mathrm{CG}(a_i, d+1)), \\ \mathrm{VG}(b_j) &:= \mathrm{GA}_{\mathrm{Y}}^{d+1,t_{\mathrm{X}}}(\mathrm{CG}(b_j, 1), \dots, \mathrm{CG}(b_j, d+1)), \\ S &:= \mathrm{GA}_{\mathrm{X}}^{d+1,t_{\mathrm{Y}}}(\underbrace{\mathbf{0}_{\mathrm{X}}, \dots, \mathbf{0}_{\mathrm{X}}, \mathbf{1}_{\mathrm{X}}}_{d+1}). \end{aligned}$$

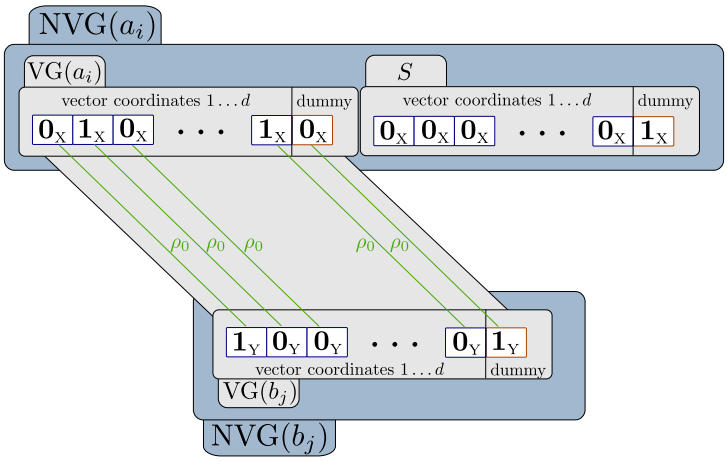


FIGURE 4.3: Schematic illustration of the coordinate, vector, and normalized vector gadgets, case $\langle a_i, b_j \rangle = 0$: Aligning VG($b_j$) with VG($a_i$) achieves alignment cost $(d+1)\rho_0$.

Note that $\text{type}(\text{VG}(a_1)) = \ldots = \text{type}(\text{VG}(a_n)) = \text{type}(S) =: t'_X$ and $\text{type}(\text{VG}(b_1)) = \ldots = \text{type}(\text{VG}(b_m)) =: t'_Y$, because the type of the output of the alignment gadget only depends on the number of input elements and their type, which are all t_X or all t_Y, respectively. We introduce *normalized vector gadgets* as follows

$$\begin{aligned} \text{NVG}(a_i) &:= \text{GA}_X^{1,t'_Y}(S, \text{VG}(a_i)), \\ \text{NVG}(b_j) &:= \text{GA}_Y^{2,t'_X}(\text{VG}(b_j)). \end{aligned}$$

Note that $\text{type}(\text{NVG}(a_1)) = \ldots = \text{type}(\text{NVG}(a_n)) =: t''_X$ and $\text{type}(\text{NVG}(b_1)) = \ldots = \text{type}(\text{NVG}(b_m)) =: t''_Y$. We finally obtain x and y by setting

$$\begin{aligned} x &:= \text{GA}_X^{m,t''_Y}(\text{NVG}(a_1), \ldots, \text{NVG}(a_n), \text{NVG}(a_1), \ldots, \text{NVG}(a_n)), \\ y &:= \text{GA}_Y^{2n,t''_X}(\text{NVG}(b_1), \ldots, \text{NVG}(b_m)). \end{aligned}$$

We denote by C, C', C'' the value C in the three invocations of Property (4.2) of the alignment gadget.

Observe that x and y have length $\mathcal{O}((n+m)d)$ and can be constructed in time $\mathcal{O}((n+m)d)$ by applying the algorithm implicit in Definition 4.1.1 three times. Moreover, if $\boldsymbol{\delta}$ admits an *unbalanced* alignment gadget, then we have $|x| = \mathcal{O}(nd)$ and $|y| = \mathcal{O}(md)$. It remains to show that if we know $\boldsymbol{\delta}(x, y)$ then we can decide the given OV instance in constant time, i.e., correctness of our construction, which we do below. This finishes our reduction from OV to the problem of computing $\boldsymbol{\delta}$. To obtain Theorem 4.1.3, let $0 < \alpha \leq 1$ and assume that $\boldsymbol{\delta}(x', y')$ can be computed in time $\mathcal{O}(M^{2-\varepsilon})$ whenever $|x'| \leq N$, $|y'| \leq M$, and $N^{\alpha - o(1)} \leq M \leq N^{\alpha + o(1)}$. Then in particular for $n = m$ we can compute $\boldsymbol{\delta}(x, y)$ in time $\mathcal{O}(\min\{|x|, |y|\}^{2-\varepsilon} + |x| + |y|) = \mathcal{O}(((n+m)d)^{2-\varepsilon}) = \mathcal{O}((nd)^{2-\varepsilon})$, contradicting OVH. In case of an unbalanced alignment gadget, assume that $\boldsymbol{\delta}(x', y')$ can be computed in time $\mathcal{O}((NM)^{1-\varepsilon})$ whenever $|x'| \leq N$, $|y'| \leq M$, and $N^{\alpha - o(1)} \leq M \leq N^{\alpha + o(1)}$. Then for $m = \Theta(n^\alpha)$ and $d \leq n^{o(1)}$ we can compute $\boldsymbol{\delta}(x, y)$ in time $\mathcal{O}((|x||y|)^{1-\varepsilon} + |x| + |y|) = \mathcal{O}(((nd)(md))^{1-\varepsilon} + (n+m)d) = \mathcal{O}((nm)^{1-\varepsilon/2})$, contradicting UOVH. This proves Theorem 4.1.3.

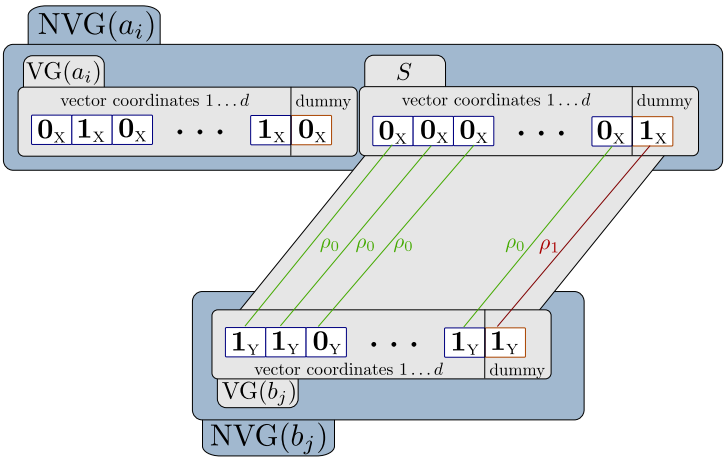


FIGURE 4.4: Schematic illustration of the coordinate, vector, and normalized vector gadgets, case $\langle a_i, b_j \rangle \neq 0$: Aligning $\mathrm{VG}(b_j)$ with S achieves an alignment cost of $d\rho_0 + \rho_1$.

Correctness. We now prove correctness of our construction and refer to Figure 4.3 and 4.4 for an intuition for coordinate, vector, and normalized vector gadgets. Let $\rho_0 := \boldsymbol{\delta}(\mathbf{0}_X, \mathbf{0}_Y) = \boldsymbol{\delta}(\mathbf{0}_X, \mathbf{1}_Y) = \boldsymbol{\delta}(\mathbf{1}_X, \mathbf{0}_Y)$ and $\rho_1 := \boldsymbol{\delta}(\mathbf{1}_X, \mathbf{1}_Y)$. Recall that $\rho_0 < \rho_1$.

Claim 4.1.4. *For any $i \in [n]$, $j \in [m]$, if $\langle a_i, b_j \rangle = 0$, then $\boldsymbol{\delta}(\mathrm{VG}(a_i), \mathrm{VG}(b_j)) = C + (d+1)\rho_0$. Otherwise, $\boldsymbol{\delta}(\mathrm{VG}(a_i), \mathrm{VG}(b_j)) \geq C + d\rho_0 + \rho_1$. Moreover, $\boldsymbol{\delta}(S, \mathrm{VG}(b_j)) = C + d\rho_0 + \rho_1$.*

Proof. If $\langle a_i, b_j \rangle = 0$, then the structured alignment $\Lambda = \{(1,1), \dots, (d+1, d+1)\}$ has a cost of $\mathbf{c}(\Lambda) = \sum_{k=1}^{d+1} \boldsymbol{\delta}(\mathrm{CG}(a_i, k), \mathrm{CG}(b_j, k)) = (d+1)\rho_0$, since in each component k at least one value is $\mathbf{0}_X$ or $\mathbf{0}_Y$, incurring a cost of ρ_0 (indeed, even in position $k = d{+}1$, we have $\mathrm{CG}(a_i, d+1) = \mathbf{0}_X$). By definition of alignment gadgets, we obtain $\boldsymbol{\delta}(\mathrm{VG}(a_i), \mathrm{VG}(b_j)) - C \leq (d+1)\rho_0$. Moreover, since the cost $\mathbf{c}(\Lambda)$ of any alignment $\Lambda \in \boldsymbol{\Lambda}_{d+1,d+1}$ consists of $d+1$ summands of the form $\boldsymbol{\delta}(u_X, u_Y)$ with $u_X \in \{\mathbf{0}_X, \mathbf{1}_X\}, u_Y \in \{\mathbf{0}_Y, \mathbf{1}_Y\}$, we also have $\boldsymbol{\delta}(\mathrm{VG}(a_i), \mathrm{VG}(b_j)) - C \geq (d+1)\rho_0$.

If $\langle a_i, b_j \rangle \neq 0$, then consider any $\Lambda \in \boldsymbol{\Lambda}_{n,m}$. If $|\Lambda| = d+1$ then $\Lambda = \{(1,1), \dots, (d+1, d{+}1)\}$, and this alignment incurs a cost of at least $d\rho_0{+}\rho_1$, since in at least one position k we have $\mathrm{CG}(a_i, k) = \mathbf{1}_X$ and $\mathrm{CG}(b_j, k) = \mathbf{1}_Y$. Otherwise, if $|\Lambda| < d{+}1$, then $\mathbf{c}(\Lambda)$ consists of $d+1$ summands of the form $\boldsymbol{\delta}(u_X, u_Y)$ with $u_X \in \{\mathbf{0}_X, \mathbf{1}_X\}, u_Y \in \{\mathbf{0}_Y, \mathbf{1}_Y\}$, and at least one of these summands is the punishment term $\max_{k,\ell} \boldsymbol{\delta}(\mathrm{CG}(a_i, k), \mathrm{CG}(b_j, \ell))$ because $|\Lambda| < d+1$. Since $\langle a_i, b_j \rangle = 1$, the punishment term is ρ_1 and we obtain $\mathbf{c}(\Lambda) \geq d\rho_0 + \rho_1$. By definition of alignment gadgets, we have $\boldsymbol{\delta}(\mathrm{VG}(a_i), \mathrm{VG}(b_j)) - C \geq d\rho_0 + \rho_1$.

We argue similarly for $\boldsymbol{\delta}(S, \mathrm{VG}(b_j))$: The alignment $\{(1,1), \dots, (d{+}1, d{+}1)\}$ incurs a cost of $d\rho_0{+}\rho_1$, since the $(d{+}1)$-st components of S and $\mathrm{VG}(b_j)$ are $\mathbf{1}_X$ and $\mathrm{VG}(b_j, d{+}1) = \mathbf{1}_Y$, respectively, while all other components of S are $\mathbf{0}_X$. Moreover, any alignment with $|\Lambda| < d+1$ incurs a punishment term, so that it incurs cost of at least $d\rho_0 + \rho_1$. □

Claim 4.1.5. *For any $i \in [n]$, $j \in [m]$, if $\langle a_i, b_j \rangle = 0$ then $\boldsymbol{\delta}(\mathrm{NVG}(a_i), \mathrm{NVG}(b_j)) = C + C' + (d+1)\rho_0 =: \rho_0'$. Otherwise, $\boldsymbol{\delta}(\mathrm{NVG}(a_i), \mathrm{NVG}(b_j)) = C + C' + d\rho_0 + \rho_1 =: \rho_1'$.*

Proof. Note that $\{(1,1)\}$, $\{(2,1)\}$, and $\emptyset$ are the only alignments in $\boldsymbol{\Lambda}_{2,1}$, which corresponds to aligning $(S, \mathrm{VG}(b_j))$ or $(\mathrm{VG}(a_i), \mathrm{VG}(b_j))$ or nothing. Moreover, the structured alignments are $\{(1,1)\}$ and $\{(2,1)\}$. Observe that the cost of the alignment $\emptyset$ is simply

the maximum of the other two alignments. By Claim 4.1.4, if $\langle a_i, b_j \rangle = 0$ then the minimal cost is $C + (d+1)\rho_0$, attained by alignment $\{(2,1)\}$. Otherwise, the minimal cost is $C + d\rho_0 + \rho_1$, attained by alignment $\{(1,1)\}$. By definition of alignment gadgets, this yields that $\boldsymbol{\delta}(\mathrm{NVG}(a_i), \mathrm{NVG}(b_j)) - C'$ is equal to $C + (d+1)\rho_0$ or $C + d\rho_0 + \rho_1$, respectively. □

The claim shows that $\boldsymbol{\delta}(\mathrm{NVG}(a_i), \mathrm{NVG}(b_j))$ attains one of only two values, depending on whether $\langle a_i, b_j \rangle = 0$.

Claim 4.1.6. *If there is no $i \in [n], j \in [m]$ with $\langle a_i, b_j \rangle = 0$, then $\boldsymbol{\delta}(x,y) \geq C'' + m\rho_1'$. Otherwise, $\boldsymbol{\delta}(x,y) \leq C'' + (m-1)\rho_1' + \rho_0'$.*

Proof. If $\langle a_i, b_j \rangle \neq 0$ for all i, j, then the previous claim yields $\boldsymbol{\delta}(\mathrm{NVG}(a_i), \mathrm{NVG}(b_j)) \geq \rho_1'$ for all i, j. Since the cost of any alignment consists of m summands of the form $\boldsymbol{\delta}(\mathrm{NVG}(a_i), \mathrm{NVG}(b_j))$ for some i, j, the cost of any alignment is at least $m\rho_1'$. By definition of alignment gadgets, we obtain $\boldsymbol{\delta}(x,y) - C'' \geq m\rho_1'$.

If $\langle a_i, b_j \rangle = 0$ for some i, j, then consider the structured alignment $\Lambda = \{(\Delta + 1, 1), \ldots, (\Delta + m, m)\}$ with $\Delta := i - j$ if $i \geq j$, or $\Delta := n + i - j$ if $i < j$. Its cost consists of m summands, one of which being $\boldsymbol{\delta}(\mathrm{NVG}(a_i), \mathrm{NVG}(b_j)) = \rho_0'$ while all others are at most ρ_1'. Hence, the cost of Λ is at most $(m-1)\rho_1' + \rho_0'$ and by definition of alignment gadgets, we obtain $\boldsymbol{\delta}(x,y) - C'' \leq (m-1)\rho_1' + \rho_0'$. □

By setting $\rho := C'' + (m-1)\rho_1' + \rho_0'$, we have found a threshold such that $\boldsymbol{\delta}(x,y) \leq \rho$ if and only if there is a pair (i,j) with $\langle a_i, b_j \rangle = 0$. Thus, computing $\boldsymbol{\delta}(x,y)$ allows to decide the given OV instance. This finishes the proof of Theorem 4.1.3.

4.2 Dynamic Time Warping

We present coordinate values and an *unbalanced* alignment gadget for DTW on one-dimensional curves taking values in $\mathbb{N}_0$, i.e., we consider the set of inputs $\mathcal{I} := \bigcup_{k \geq 0} \mathbb{N}_0^k$. We define the type of a sequence $x \in \mathcal{I}$ as $\mathrm{type}(x) := (|x|, \sum_i x[i], \max_i x[i])$.

Lemma 4.2.1. DTW *admits coordinate values by setting*

$$\mathbf{1}_{\mathrm{X}} := 1100,\ \mathbf{0}_{\mathrm{X}} := 0110,\ \mathbf{1}_{\mathrm{Y}} := 0011,\ \mathbf{0}_{\mathrm{Y}} := 1010.$$

Proof. All four sequences have the same length, sum of all entries and maximum entry, so they have equal type. Short calculations show that $4 = \boldsymbol{\delta}_{\mathrm{DTW}}(\mathbf{1}_{\mathrm{X}}, \mathbf{1}_{\mathrm{Y}}) > \boldsymbol{\delta}_{\mathrm{DTW}}(\mathbf{0}_{\mathrm{X}}, \mathbf{1}_{\mathrm{Y}}) = \boldsymbol{\delta}_{\mathrm{DTW}}(\mathbf{0}_{\mathrm{X}}, \mathbf{0}_{\mathrm{Y}}) = \boldsymbol{\delta}_{\mathrm{DTW}}(\mathbf{1}_{\mathrm{X}}, \mathbf{0}_{\mathrm{Y}}) = 1$. □

Definition 4.2.2. *Consider instances $x_1, \ldots, x_n \in \mathcal{I}_{t_{\mathrm{X}}}$ and $y_1, \ldots, y_m \in \mathcal{I}_{t_{\mathrm{Y}}}$ with $n \geq m$ and types $t_{\mathrm{X}} = (\ell_{\mathrm{X}}, s_{\mathrm{X}}, z_{\mathrm{X}}), t_{\mathrm{Y}} = (\ell_{\mathrm{Y}}, s_{\mathrm{Y}}, z_{\mathrm{Y}})$. We define $M := 2z$, where $z := \max\{z_{\mathrm{X}}, z_{\mathrm{Y}}\}$ is the largest value contained in any of the one-dimensional curves $x_1, \ldots, x_n, y_1, \ldots, y_m$, and we set $\kappa := 3(\ell_{\mathrm{X}} + \ell_{\mathrm{Y}})$. We construct*

$$\begin{aligned} \mathrm{GA}_{\mathrm{X}}^{m,t_{\mathrm{Y}}}(x_1, \ldots, x_n) &:= M^\kappa\, x_1\, M^\kappa\, x_2\, M^\kappa \ldots M^\kappa\, x_n\, M^\kappa, \\ \mathrm{GA}_{\mathrm{Y}}^{n,t_{\mathrm{X}}}(y_1, \ldots, y_m) &:= M^\kappa\, y_1\, M^\kappa\, y_2\, M^\kappa \ldots M^\kappa\, y_m\, M^\kappa, \end{aligned}$$

where M^κ is to be understood as a sequence with κ times the entry M.

Lemma 4.2.3. *Definition 4.2.2 realizes an unbalanced alignment gadget for dynamic time warping.*

Thus, Theorem 4.1.3 is applicable, ruling out $\mathcal{O}((nm)^{1-\varepsilon})$-algorithms for DTW on one-dimensional curves over $\mathbb{N}_0$ (under OVH and SETH). To restrict the alphabet further, note that our coordinate values use the alphabet $\{0,1\} \subseteq \mathbb{N}_0$ and each invocation of the alignment gadget introduces a new symbol which is twice as large as the largest value seen so far. Since in the proof of Theorem 4.1.3 we use alignment gadgets thrice, we introduce the symbols 2, 4, and 8. In total, we prove quadratic-time hardness of DTW on one-dimensional curves taking values in $\{0,1,2,4,8\} \subseteq \mathbb{N}_0$. This proves Theorems 2.1.1 and 2.1.3.

Proof of Lemma 4.2.3. Observe that the instance given by $x := \mathrm{GA}_{\mathrm{X}}^{m,t_{\mathrm{Y}}}(x_1,\ldots,x_n)$ and $y := \mathrm{GA}_{\mathrm{Y}}^{n,t_{\mathrm{X}}}(y_1,\ldots,y_m)$ can be computed in time $\mathcal{O}((n+m)(\ell_{\mathrm{X}}+\ell_{\mathrm{Y}}))$, yielding sequences of length $\mathcal{O}(n(\ell_{\mathrm{X}}+\ell_{\mathrm{Y}}))$ and $\mathcal{O}(m(\ell_{\mathrm{X}}+\ell_{\mathrm{Y}}))$, respectively. Moreover, $\mathrm{type}(x)$ and $\mathrm{type}(y)$ only depend on $t_{\mathrm{X}}, t_{\mathrm{Y}}, n, m$. It remains to show the inequalities (4.2) of Definition 4.1.1, for which we set $C := (n-m)(\ell_{\mathrm{X}}M - s_{\mathrm{X}})$.

We start with the following useful observations.

Claim 4.2.4. *Let $\ell \geq 1$ and $a, a', b, b' \in \mathbb{N}_0$. For any $i \in [n], j \in [m]$, we have*

(i) $\boldsymbol{\delta}_{\mathrm{DTW}}(x_i, M^\ell) \geq \boldsymbol{\delta}_{\mathrm{DTW}}(x_i, M) = \ell_{\mathrm{X}}M - s_{\mathrm{X}} \geq \ell_{\mathrm{X}}M/2$ *and the analogous statement* $\boldsymbol{\delta}_{\mathrm{DTW}}(M^\ell, y_j) \geq \boldsymbol{\delta}_{\mathrm{DTW}}(M, y_j) = \ell_{\mathrm{Y}}M - s_{\mathrm{Y}} \geq \ell_{\mathrm{Y}}M/2$,

(ii) $\boldsymbol{\delta}_{\mathrm{DTW}}(x_i, y_j) < (\ell_{\mathrm{X}} + \ell_{\mathrm{Y}})M/2$,

(iii) $\boldsymbol{\delta}_{\mathrm{DTW}}(x', M^\kappa) \geq \kappa M/2$ *and* $\boldsymbol{\delta}_{\mathrm{DTW}}(M^\kappa, y') \geq \kappa M/2$ *for any substrings x' of x_i and y' of y_j*,

(iv) $\boldsymbol{\delta}_{\mathrm{DTW}}(M^a x_i M^{a'}, M^b y_j M^{b'}) \geq \boldsymbol{\delta}_{\mathrm{DTW}}(x_i, y_j)$.

Proof. For (i), observe that each symbol of x_i can only be traversed together with the symbol M and hence,

$$\boldsymbol{\delta}_{\mathrm{DTW}}(x_i, M^\ell) \geq \boldsymbol{\delta}_{\mathrm{DTW}}(x_i, M) = \sum_{k=1}^{\ell_{\mathrm{X}}} |M - x_i[k]| = \ell_{\mathrm{X}}M - \sum_{k=1}^{\ell_{\mathrm{X}}} x_i[k] = \ell_{\mathrm{X}}M - s_{\mathrm{X}}.$$

Since $x_i[k] \leq z = M/2$, we have $s_{\mathrm{X}} \leq \ell_{\mathrm{X}}M/2$. The statement for y_j is symmetric.

For (ii) and (iii), note that all symbols in x' are in $[0,z]$. Hence, we obtain that $\boldsymbol{\delta}_{\mathrm{DTW}}(x_i, y_j) \leq \max\{|x_i|, |y_j|\} \cdot z < (\ell_{\mathrm{X}}+\ell_{\mathrm{Y}})M/2$. Likewise, $\boldsymbol{\delta}_{\mathrm{DTW}}(x', M^\kappa) \geq \kappa(M-z) = \kappa M/2$. The inequality for y_j follows symmetrically.

To prove (iv), consider an optimal traversal T of $M^a x_i M^{a'}$ and $M^b y_j M^{b'}$. We construct a traversal T' of x_i and y_j that has no larger cost. If T does not already traverse $x_i[1]$ together with $y_j[1]$, then at some step in T either a symbol in x_i is traversed together with a symbol of the prefix M^b or a symbol in y_j is traversed together with a symbol of the prefix M^a. Let us assume the first case, since the second is symmetric. A contiguous part T^H of T consists of traversing a prefix x' of x_i together with all symbols in M^b, incurring a cost of at least $|x'|M/2$. Let T^R be the remaining part of T after T^H. We construct a traversal T'' of $x_i M^{a'}$ and $y_j M^{b'}$ as follows. We first traverse x' together with $y_j[1]$ and then follow T^R, which is possible since T^R starts at $y_j[1]$. Since traversing x' together with $y_j[1]$ incurs a cost of at most $|x'|z = |x'|M/2$, which is smaller than the cost of T^H, the cost of our constructed traversal T'' is no larger than the cost of T. Symmetrically, we eliminate the suffixes $M^{a'}$ and $M^{b'}$ and construct a traversal T' of x_i and y_j of cost no larger than T. □

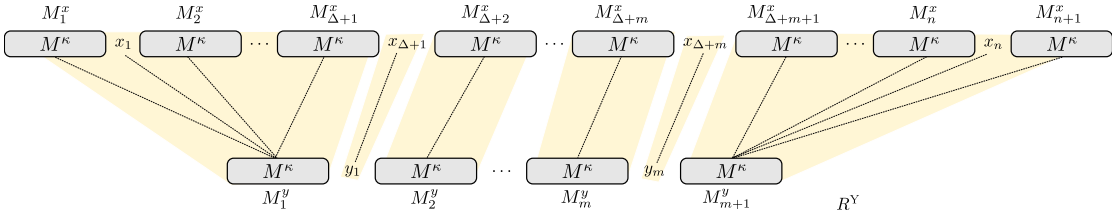


FIGURE 4.5: Optimal traversal corresponding to structured alignment $\Lambda = \{(\Delta + j, j) \mid j \in [m]\} \in \mathcal{S}_{n,m}$.

We first verify that

$$\boldsymbol{\delta}_{\mathrm{DTW}}(x, y) \leq (n - m)(\ell_{\mathrm{X}} M - s_{\mathrm{X}}) + \min_{\Lambda \in \mathcal{S}_{n,m}} \sum_{(i,j) \in \Lambda} \boldsymbol{\delta}_{\mathrm{DTW}}(x_i, y_j),$$

by designing a traversal (illustrated in Figure 4.5) that achieves this bound. Let $\Lambda \in \mathcal{S}_{n,m}$ be the alignment minimizing the expression, and note that $\Lambda = \{(\Delta + 1, 1), \ldots, (\Delta + m, m)\}$ for some $0 \leq \Delta \leq n - m$. We first traverse $M^\kappa x_1 M^\kappa \ldots M^\kappa x_\Delta$ together with the first symbol of y, M, which contributes a cost of $\sum_{i=1}^{\Delta} \boldsymbol{\delta}_{\mathrm{DTW}}(x_i, M) = \Delta(\ell_{\mathrm{X}} M - s_{\mathrm{X}})$ by Claim 4.2.4(i). For $i = 1, \ldots, m$ we repeat the following: We traverse $M^\kappa x_{\Delta+i}$ together with $M^\kappa y_i$ by traversing M^κ and M^κ simultaneously, and x_i and y_i in a locally optimal manner; this incurs a cost of $\boldsymbol{\delta}_{\mathrm{DTW}}(x_{\Delta+i}, y_i)$ for each i. Finally, we traverse the last block M^κ in y with the current block M^κ in x, and then traverse the remainder $x_{\Delta+m+1} M^\kappa \ldots M^\kappa x_n M^\kappa$ of x together with the last symbol of y, M. The total cost amounts to $\Delta(\ell_{\mathrm{X}} M - s_{\mathrm{X}}) + \sum_{i=1}^{m} \boldsymbol{\delta}_{\mathrm{DTW}}(x_{\Delta+i}, y_i) + (n - \Delta - m)(\ell_{\mathrm{X}} M - s_{\mathrm{X}}) = (n - m)(\ell_{\mathrm{X}} M - s_{\mathrm{X}}) + \sum_{(i,j) \in \Lambda} \boldsymbol{\delta}_{\mathrm{DTW}}(x_i, y_j)$.

In the remainder of the proof, we verify that

$$\boldsymbol{\delta}_{\mathrm{DTW}}(x, y) \geq (n - m)(\ell_{\mathrm{X}} M - s_{\mathrm{X}}) + \min_{\Lambda \in \boldsymbol{\Lambda}_{n,m}} \Big[\sum_{(i,j) \in \Lambda} \boldsymbol{\delta}_{\mathrm{DTW}}(x_i, y_j) + (m - |\Lambda|) \max_{i,j} \boldsymbol{\delta}_{\mathrm{DTW}}(x_i, y_j) \Big].$$

Let $T^* = ((a_1^*, b_1^*), \ldots, (a_t^*, b_t^*))$ be an optimal traversal of (x, y) (see Chapter 3 for the definition of traversals). Substrings x' of x and y' of y are *paired* if for some index i in x' and some index j in y' we have $(i, j) = (a_{t'}^*, b_{t'}^*)$ for some $1 \leq t' \leq t$.

We call the i-th occurrence of M^κ in x the *i-th M-block* M_i^x of x, and similarly for y. Let $X := \{M_i^x \mid i \in [n + 1]\}$, $Y := \{M_j^y \mid j \in [m + 1]\}$ be the sets of all M-blocks of x and y, respectively. We define a bipartite graph G_M with vertex set $X \cup Y$, where M-blocks M_i^x and M_j^y are connected by an edge if and only if they are paired. We show the following properties of G_M.

Claim 4.2.5 (Planarity)**.** *For any paired M_i^x, M_j^y and paired $M_{i'}^x, M_{j'}^y$ we have $i \leq i'$ and $j \leq j'$ (or $i \geq i'$ and $j \geq j'$).*

Proof. By monotonicity of traversals, for $k \leq k'$ we have $a_k^* \leq a_{k'}^*$ and $b_k^* \leq b_{k'}^*$. Thus, if $x[a_k^*]$ is in M_i^x and $x[a_{k'}^*]$ is in $M_{i'}^x$, then $i \leq i'$. Similarly, if $y[b_k^*]$ is in M_j^y and $y[b_{k'}^*]$ is in $M_{j'}^y$, then $j \leq j'$. Hence, for any paired M_i^x, M_j^y and $M_{i'}^x, M_{j'}^y$ we have $i \leq i'$, $j \leq j'$ or $i \geq i'$, $j \geq j'$. □

Claim 4.2.6. *G_M has no isolated vertices.*

Proof. Assume that some M-block M_i^x is not paired with any M-block of y, and let i be maximal with this property. Note that $i < n+1$, as the last M-block of x is always paired with the last M-block of y. Then there is some $j \in [m]$ such that M_i^x is paired with y_j, but M_i^x is not paired with any part of y outside y_j. By maximality of i and planarity, M_{j+1}^y is paired with x_i or M_{i+1}^x, as otherwise M_{i+1}^x is not paired with any $M_{j'}^y$.

We can find a cheaper traversal as follows. Consider the first time t_1 at which the traversal T^* is simultaneously at the first symbol of M_i^x and any symbol of y_j (this exists since M_i^x is paired to y_j, but to no part of y outside y_j), and any time t_2 at which T^* is at M_{j+1}^y and x_i or at M_{j+1}^y and M_{i+1}^x. Between t_1 and t_2, T^* has a cost of at least $\boldsymbol{\delta}_{\mathrm{DTW}}(y', M^\kappa)$, where y' is any substring of y_j. By Claim 4.2.4(iii), this is at least $\kappa M/2$. We replace this part of T^* by traversing (i) the remainder of y_j with the first symbol of M_i^x, (ii) M_i^x with the necessary part of M_{j+1}^y, and (iii) the necessary part of x_i and M_{i+1}^x with the current symbol in y, M. By Claim 4.2.4(i), this incurs a cost of at most $\boldsymbol{\delta}_{\mathrm{DTW}}(x_i, M) + \boldsymbol{\delta}_{\mathrm{DTW}}(y_j, M) = \ell_{\mathrm{X}} M - s_{\mathrm{X}} + \ell_{\mathrm{Y}} M - s_{\mathrm{Y}} \leq (\ell_{\mathrm{X}} + \ell_{\mathrm{Y}})M$. By our choice of $\kappa = 3(\ell_{\mathrm{X}} + \ell_{\mathrm{Y}})$, we improve the cost of the traversal, contradicting optimality of T^*. This shows that no vertex in X is isolated, we argue similarly for vertices in Y. □

Claim 4.2.7. *G_M contains no path of length 3.*

Proof. Assume that G_M contains a path $M_i^x - M_j^y - M_{i'}^x - M_{j'}^y$. Without loss of generality we assume $i < i'$, the case $i > i'$ is symmetric. By planarity, we have $j < j'$. Since G_M has no isolated vertices and by planarity, every $M_{i''}^x$ with $i \leq i'' \leq i'$ is paired with M_j^y, so we can assume that $i' = i+1$ (after replacing i with $i'-1$). Similarly, we can assume $j' = j+1$, and the path is $M_i^x - M_j^y - M_{i+1}^x - M_{j+1}^y$.

We can find a cheaper traversal as follows. Consider any time t_1 at which the traversal T^* is simultaneously at M_i^x and M_j^y (this exists since M_i^x and M_j^y are paired), and consider any time t_2 at which T^* is simultaneously at M_{i+1}^x and M_{j+1}^y. Between t_1 and t_2, T^* traverses x_i with (parts of) M_j^y, and y_j with (parts of) M_{i+1}^x, which by Claim 4.2.4(i) incurs a cost of at least $\boldsymbol{\delta}_{\mathrm{DTW}}(x_i, M) + \boldsymbol{\delta}_{\mathrm{DTW}}(M, y_j) \geq (\ell_{\mathrm{X}} + \ell_{\mathrm{Y}})M/2$. We replace this part of T^* by traversing (i) the remaining parts of M_i^x and M_j^y, (ii) x_i and y_j (in a locally optimal way), and (iii) the necessary parts of M_{i+1}^x and M_{j+1}^y. This incurs a cost of $\boldsymbol{\delta}_{\mathrm{DTW}}(x_i, y_j) < (\ell_{\mathrm{X}} + \ell_{\mathrm{Y}})M/2$ (by Claim 4.2.4(ii)), which contradicts optimality of T^*. □

By the above two claims, G_M is a disjoint union of stars. By planarity and since G_M has no isolated vertices, the leaves of any star in G_M have to be consecutive M-blocks. Hence, we can write the components of G_M as $C_1, \ldots, C_s$ and $C'_1, \ldots, C'_{s'}$ with

$$C_k = \{M_{i_k}^x\} \cup \{M_{j_k}^y, M_{j_k+1}^y, \ldots, M_{j_k+d_k-1}^y\},$$
$$C'_k = \{M_{j'_k}^y\} \cup \{M_{i'_k}^x, M_{i'_k+1}^x, \ldots, M_{i'_k+d'_k-1}^x\}.$$

Claim 4.2.8. *We have $\sum_{k=1}^{s} d_k = m - s' + 1$ and $\sum_{k=1}^{s'} d'_k = n - s + 1$.*

Proof. Since the components $C_1, \ldots, C_s$ and $C'_1, \ldots, C'_{s'}$ partition G_M restricted to Y, we have $s' + \sum_{k=1}^{s} d_k = \sum_{k=1}^{s'} |C'_k \cap Y| + \sum_{k=1}^{s} |C_k \cap Y| = |Y| = m+1$. The second claim follows analogously. □

We construct an alignment by aligning all those x_i, y_j that lie between two consecutive components of G_M. More formally, we define an alignment Λ by aligning $(i_k - 1, j_k - 1)$ (for all $k \in [s]$ with $i_k, j_k > 1$) and aligning $(i'_k - 1, j'_k - 1)$ (for all $k \in [s']$ with $i'_k, j'_k > 1$). Since G_M has no isolated vertices, Λ is a valid alignment. We have $|\Lambda| = s + s' - 1$, since

only the leftmost component of G_M has $i_k = 1$, $j_k = 1$, $i'_k = 1$, or $j'_k = 1$, and all other components give rise to exactly one aligned pair.

Let us calculate the cost of T^*. Each y_j that lies between the leaves of a star C_k in G_M (i.e., $j_k \le j < j_k + d_k$) has to be traversed together with (parts of) $M^x_{i_k}$. By Claim 4.2.4(i), this incurs a cost of at least $\boldsymbol{\delta}_{\mathrm{DTW}}(M, y_j) = \ell_{\mathrm{Y}} M - s_{\mathrm{Y}}$. Likewise, each x_i that lies between the leaves of a star C'_k incurs a cost of at least $\ell_{\mathrm{X}} M - s_{\mathrm{X}}$. For any $(i,j) \in \Lambda$, x_i is traversed together with a substring of $M^\kappa y_j M^\kappa$, and y_j is traversed together with a substring of $M^\kappa x_i M^\kappa$. Hence, there are $a, a', b, b' \ge 0$ such that we traverse $M^a x_i M^{a'}$ together with $M^b y_j M^{b'}$. By Claim 4.2.4(iv), this incurs a cost of at least $\boldsymbol{\delta}_{\mathrm{DTW}}(x_i, y_j)$. In total, the cost of the optimal traversal T^* is

$$\boldsymbol{\delta}_{\mathrm{DTW}}(x,y) \ge \sum_{k=1}^{s} (d_k - 1)(\ell_{\mathrm{Y}} M - s_{\mathrm{Y}}) + \sum_{k=1}^{s'} (d'_k - 1)(\ell_{\mathrm{X}} M - s_{\mathrm{X}}) + \sum_{(i,j)\in\Lambda} \boldsymbol{\delta}_{\mathrm{DTW}}(x_i, y_j).$$

By Claim 4.2.8, we have $\sum_{k=1}^{s}(d_k - 1) = m - (s + s' - 1) = m - |\Lambda|$. Similarly, $\sum_{k=1}^{s'}(d'_k - 1) = n - |\Lambda| = (n-m) + (m - |\Lambda|)$. Additionally bounding $\ell_{\mathrm{Y}} M - s_{\mathrm{Y}} + \ell_{\mathrm{X}} M - s_{\mathrm{X}} \ge (\ell_{\mathrm{X}} + \ell_{\mathrm{Y}})M/2 > \max_{i,j} \boldsymbol{\delta}_{\mathrm{DTW}}(x_i, y_j)$, we obtain the desired inequality

$$\boldsymbol{\delta}_{\mathrm{DTW}}(x,y) \ge (m - |\Lambda|) \max_{i,j} \boldsymbol{\delta}_{\mathrm{DTW}}(x_i, y_j) + (n-m)(\ell_{\mathrm{X}} M - s_{\mathrm{X}}) + \sum_{(i,j)\in\Lambda} \boldsymbol{\delta}_{\mathrm{DTW}}(x_i, y_j). \qquad \square$$

4.3 Longest Common Subsequence

In this section, we present an alignment gadget (and thus a hardness proof) for LCS, which is simpler than for the more general problem $\mathrm{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$ in Section 4.4. Recall that we denote by $\mathrm{LCS}(x,y)$ a longest common subsequence of x and y, and by $L(x,y)$ the length of any longest common subsequence of x and y. Note that LCS is a maximization problem, but Definition 4.1.1 implicitly assumes a minimization problem, so we instead consider the number of unmatched symbols $\boldsymbol{\delta}_{\mathrm{LCS}}(x,y) := |x| + |y| - 2 \cdot L(x,y)$ for binary strings x, y. This is equivalent to $\mathrm{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$ for $c_{\text{del-x}} = c_{\text{del-y}} = 1$, $c_{\text{match}} = 0$, and $c_{\text{subst}} = 2$.

We present an alignment gadget and coordinate values for LCS over binary strings, i.e., we consider the set of inputs $\mathcal{I} := \bigcup_{k \ge 0} \{0,1\}^k$. Here, the type of a binary string x is defined as $\mathrm{type}(x) := (\ell, s)$, where $\ell = |x|$ and $s = \sum_i x_i$ denotes the number of ones in x.

Lemma 4.3.1. LCS *admits coordinate values by setting*

$$\mathbf{1}_{\mathrm{X}} := 11100,\ \mathbf{0}_{\mathrm{X}} := 10011,\ \mathbf{1}_{\mathrm{Y}} := 00111,\ \mathbf{0}_{\mathrm{Y}} := 11001.$$

Proof. All four strings have the same length and the same number of ones, so they have equal type. Short calculations show that $\mathrm{LCS}(\mathbf{1}_{\mathrm{X}}, \mathbf{1}_{\mathrm{Y}}) = 111$, $\mathrm{LCS}(\mathbf{1}_{\mathrm{X}}, \mathbf{0}_{\mathrm{Y}}) = 1100$, $\mathrm{LCS}(\mathbf{0}_{\mathrm{X}}, \mathbf{1}_{\mathrm{Y}}) = 0011$, and $\mathrm{LCS}(\mathbf{0}_{\mathrm{X}}, \mathbf{0}_{\mathrm{Y}}) = 1001$. Thus, $4 = \boldsymbol{\delta}_{\mathrm{LCS}}(\mathbf{1}_{\mathrm{X}}, \mathbf{1}_{\mathrm{Y}}) > \boldsymbol{\delta}_{\mathrm{LCS}}(\mathbf{1}_{\mathrm{X}}, \mathbf{0}_{\mathrm{Y}}) = \boldsymbol{\delta}_{\mathrm{LCS}}(\mathbf{0}_{\mathrm{X}}, \mathbf{1}_{\mathrm{Y}}) = \boldsymbol{\delta}_{\mathrm{LCS}}(\mathbf{0}_{\mathrm{X}}, \mathbf{1}_{\mathrm{Y}}) = 2$. $\square$

Definition 4.3.2. *Consider instances $x_1, \ldots, x_n \in \mathcal{I}_{t_{\mathrm{X}}}$ and $y_1, \ldots, y_m \in \mathcal{I}_{t_{\mathrm{Y}}}$ with $n \ge m$ and types $t_{\mathrm{X}} = (\ell_{\mathrm{X}}, s_{\mathrm{X}}), t_{\mathrm{Y}} = (\ell_{\mathrm{Y}}, s_{\mathrm{Y}})$. Set $\gamma_1 := \ell_{\mathrm{X}} + \ell_{\mathrm{Y}}, \gamma_2 := 6(\ell_{\mathrm{X}} + \ell_{\mathrm{Y}}), \gamma_3 := 10(\ell_{\mathrm{X}} + \ell_{\mathrm{Y}}) + 2s_{\mathrm{X}} - \ell_{\mathrm{X}}, \gamma_4 := 13(\ell_{\mathrm{X}} + \ell_{\mathrm{Y}})$. We* guard *the input strings by blocks of zeroes and*

ones, setting $\mathrm{G}(z) := 1^{\gamma_2}0^{\gamma_1}z0^{\gamma_1}1^{\gamma_2}$. *We define the alignment gagdet as*

$$\begin{aligned} x :=&\quad \mathrm{G}(x_1)\,0^{\gamma_3}\,\mathrm{G}(x_2)\,0^{\gamma_3}\,\ldots\,\mathrm{G}(x_{n-1})\,0^{\gamma_3}\,\mathrm{G}(x_n),\\ y :=&\; 0^{n\gamma_4}\,\mathrm{G}(y_1)\,0^{\gamma_3}\,\mathrm{G}(y_2)\,0^{\gamma_3}\,\ldots\,\mathrm{G}(y_{m-1})\,0^{\gamma_3}\,\mathrm{G}(y_m)\,0^{n\gamma_4}. \end{aligned}$$

Lemma 4.3.3. *Definition 4.3.2 realizes an alignment gadget for* LCS.

Thus, Theorem 4.1.3 is applicable, implying a lower bound of $\mathcal{O}(m^{2-\varepsilon})$ for LCS. We remark that our construction is not an *unbalanced* alignment gadget, as the length of y grows linearly in n, not necessarily in $m \leq n$. Thus, we do not obtain a conditional lower bound of $\mathcal{O}((nm)^{1-\varepsilon})$ (for $m \approx n^\alpha$ for any $0 < \alpha < 1$). Indeed, in Chapter 6, we will see that this is impossible without refuting SETH.

Proof of Lemma 4.3.3. Observe that indeed x only depends on m, t_{Y}, and $x_1, \ldots, x_n$, and type(x) only depends on n, m, t_{X}, and t_{Y}, and similarly for y. Moreover, x and y can clearly be constructed in time $\mathcal{O}((n+m)(\ell_{\mathrm{X}} + \ell_{\mathrm{Y}}))$.

It remains to prove that by setting $C := 2n\gamma_4$ we have

$$\min_{\Lambda \in \mathbf{\Lambda}_{n,m}} \mathbf{c}(\Lambda) \leq \boldsymbol{\delta}_{\mathrm{LCS}}(x, y) - C \leq \min_{\Lambda \in \mathcal{S}_{n,m}} \mathbf{c}(\Lambda). \tag{4.3}$$

For convenience, we collect here three useful observations. Recall that for a string z and indices $a \leq b$, we denote the substring from $z[a]$ to $z[b]$ by $z[a..b]$.

Fact 4.3.4. *Let x and $z_1, \ldots, z_k$ be binary strings. Set $z = z_1 \ldots z_n$. Then we have*

$$\boldsymbol{\delta}_{\mathrm{LCS}}(x, z) = \min_{x(z_1),\ldots,x(z_k)} \sum_{j=1}^{k} \boldsymbol{\delta}_{\mathrm{LCS}}(x(z_j), z_j),$$

where $x(z_1), \ldots, x(z_k)$ range over all ordered partitions *of x into k substrings, i.e., $x(z_1) = x[i_0 + 1..i_1], x(z_2) = x[i_1 + 1..i_2], \ldots, x(z_k) = x[i_{k-1} + 1..i_k]$ for any $0 = i_0 \leq i_1 \leq \ldots \leq i_k = |x|$.*

Proof. This claim is a specialization of Fact 3.5.1 to the case of LCS. □

The following additional fact has been proven in Section 3.5 as well.

Fact 3.5.3. *Let z, w be arbitrary strings and $\ell, k \in \mathbb{N}_0$. Then we have*

(i) $\boldsymbol{\delta}_{\mathrm{LCS}}(1^k z, 1^k w) = \boldsymbol{\delta}_{\mathrm{LCS}}(z, w)$,

(ii) $\boldsymbol{\delta}_{\mathrm{LCS}}(1^k z, w) \geq \boldsymbol{\delta}_{\mathrm{LCS}}(z, w) - k$, *and*

(iii) $\boldsymbol{\delta}_{\mathrm{LCS}}(0^\ell z, 1^k w) \geq \min\{k, \boldsymbol{\delta}_{\mathrm{LCS}}(z, 1^k w) + \ell\}$.

We obtain symmetric statements by flipping all bits and by reversing all involved strings.

Finally, we use the following lemma to handle prefixes (and suffixes) of x and y.

Claim 4.3.5. *Let $\ell \geq 0$. For any prefix x' of x we have $\boldsymbol{\delta}_{\mathrm{LCS}}(x', 0^\ell) \geq \ell$. Moreover, if x' is of the form $\mathrm{G}(x_1)0^{\gamma_3} \ldots \mathrm{G}(x_i)0^{\gamma_3}$ for some $0 \leq i < n$ and $\ell \geq i \cdot (2\gamma_2 + s_{\mathrm{X}})$, then $\boldsymbol{\delta}_{\mathrm{LCS}}(x', 0^\ell) = \ell$. Symmetric statements hold for any suffix of x.*

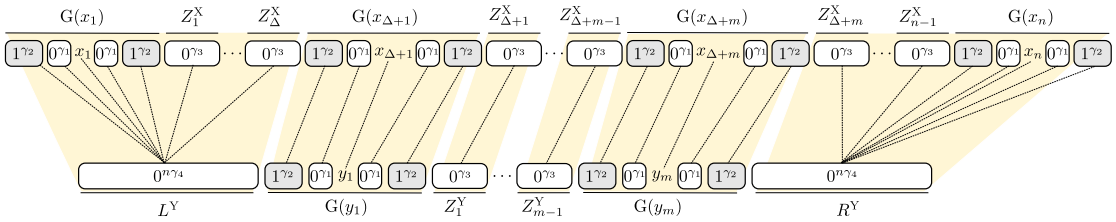


FIGURE 4.6: Optimal traversal corresponding to structured alignment $\Lambda = \{(\Delta + j, j) \mid j \in [m]\} \in \mathcal{S}_{n,m}$.

Proof. We first show that for any $i \in [n]$ the string $\mathrm{G}(x_i)0^{\gamma_3}$ contains as many ones as zeroes, and any prefix of $\mathrm{G}(x_i)0^{\gamma_3}$ contains at least as many ones as zeroes. To this end, note that each x_i has length ℓ_{X} and contains s_{X} ones, so that the number of ones of $\mathrm{G}(x_i)0^{\gamma_3}$ is $2\gamma_2 + s_{\mathrm{X}}$, while the number of zeroes is $\ell_{\mathrm{X}} - s_{\mathrm{X}} + 2\gamma_1 + \gamma_3$, and we chose γ_3 such that both values are equal. For a prefix, note that $\mathrm{G}(x_i)$ starts with γ_2 ones. Since each $\mathrm{G}(x_i)$ contains $2\gamma_1 + \ell_{\mathrm{X}} - s_{\mathrm{X}} \leq \gamma_2$ zeroes, any prefix of $\mathrm{G}(x_i)$ has as most as many zeroes as ones. Thus, we would have to advance to 0^{γ_3} to see more zeroes than ones, however, even $\mathrm{G}(x_i)0^{\gamma_3}$ does not contain more zeroes than ones.

Hence, any prefix x' of x contains at least as many ones as zeroes, implying $L(x', 0^\ell) \leq |x'|/2$. This yields $\boldsymbol{\delta}_{\mathrm{LCS}}(x', 0^\ell) = |x'| + |0^\ell| - 2L(x', 0^\ell) \geq \ell$. If x' is of the form $\mathrm{G}(x_1)0^{\gamma_3} \dots \mathrm{G}(x_i)0^{\gamma_3}$ and sufficiently many zeroes are available in 0^ℓ, then we have equality. □

Let us give names to the substrings consisting only of zeroes in x and y. In x, we denote the 0^{γ_3}-block after $\mathrm{G}(x_i)$ by Z_i^{X}, $i \in [n-1]$. In y, we denote the 0^{γ_3}-block after $\mathrm{G}(y_j)$ by Z_j^{Y}, $j \in [m-1]$. Moreover, we denote the prefix $0^{n\gamma_4}$ by L^{Y} and the suffix $0^{n\gamma_4}$ by R^{Y}.

We now show the upper bound of (4.3), i.e., $\boldsymbol{\delta}_{\mathrm{LCS}}(x, y) \leq 2n\gamma_4 + \min_{\Lambda \in \mathcal{S}_{n,m}} \mathbf{c}(\Lambda)$. Consider a structured alignment $\Lambda = \{(\Delta + 1, 1), \dots, (\Delta + m, m)\} \in \mathcal{S}_{n,m}$. We construct an ordered partition of x as in Fact 4.3.4 by setting (see Figure 4.6)

$$\begin{aligned} x(\mathrm{G}(y_j)) &:= \mathrm{G}(x_{\Delta+j}) \qquad \text{for } j \in [m], \\ x(Z_j^{\mathrm{Y}}) &:= Z_{\Delta+j}^{\mathrm{X}} \qquad \text{for } j \in [m-1], \\ x(L^{\mathrm{Y}}) &:= \mathrm{G}(x_1)Z_1^{\mathrm{X}} \dots \mathrm{G}(x_\Delta)Z_\Delta^{\mathrm{X}}, \\ x(R^{\mathrm{Y}}) &:= Z_{\Delta+m}^{\mathrm{X}}\mathrm{G}(x_{\Delta+m+1}) \dots Z_{n-1}^{\mathrm{X}}\mathrm{G}(x_n). \end{aligned}$$

Note that these strings indeed partition x and y, respectively. Thus, Fact 4.3.4 yields

$$\begin{aligned} \boldsymbol{\delta}_{\mathrm{LCS}}(x, y) \leq \boldsymbol{\delta}_{\mathrm{LCS}}(x(L^{\mathrm{Y}}), L^{\mathrm{Y}}) + \boldsymbol{\delta}_{\mathrm{LCS}}(x(R^{\mathrm{Y}}), R^{\mathrm{Y}}) \\ + \sum_{j=1}^{m} \boldsymbol{\delta}_{\mathrm{LCS}}(\mathrm{G}(x_{\Delta+j}), \mathrm{G}(y_j)) + \sum_{j=1}^{m-1} \boldsymbol{\delta}_{\mathrm{LCS}}(Z_{\Delta+j}^{\mathrm{X}}, Z_j^{\mathrm{Y}}). \end{aligned}$$

Since $L^{\mathrm{Y}} = 0^{n\gamma_4}$ and $x(L^{\mathrm{Y}})$ is a prefix of x of the correct form, by Claim 4.3.5 we have $\boldsymbol{\delta}_{\mathrm{LCS}}(x(L^{\mathrm{Y}}), L^{\mathrm{Y}}) = n\gamma_4$ (note that γ_4 is chosen sufficiently large to make Claim 4.3.5 applicable). Similarly, we obtain $\boldsymbol{\delta}_{\mathrm{LCS}}(x(R^{\mathrm{Y}}), R^{\mathrm{Y}}) = n\gamma_4$. Since $Z_i^{\mathrm{X}} = Z_j^{\mathrm{Y}} = 0^{\gamma_3}$, we have $\boldsymbol{\delta}_{\mathrm{LCS}}(Z_{\Delta+j}^{\mathrm{X}}, Z_j^{\mathrm{Y}}) = 0$. Finally, by matching the guarding ones and zeroes of $\mathrm{G}(x_{\Delta+j}) = 1^{\gamma_2}0^{\gamma_1}x_{\Delta+j}0^{\gamma_1}1^{\gamma_2}$ and $\mathrm{G}(y_j) = 1^{\gamma_2}0^{\gamma_1}y_j0^{\gamma_1}1^{\gamma_2}$ we obtain that $\boldsymbol{\delta}_{\mathrm{LCS}}(\mathrm{G}(x_{\Delta+j}), \mathrm{G}(y_j)) \leq$

$\boldsymbol{\delta}_{\mathrm{LCS}}(x_{\Delta+j}, y_j)$. Hence we have

$$\boldsymbol{\delta}_{\mathrm{LCS}}(x, y) \leq 2n\gamma_4 + \sum_{(i,j)\in\Lambda} \boldsymbol{\delta}_{\mathrm{LCS}}(x_i, y_j).$$

As $\Lambda \in \mathcal{S}_{n,m}$ was arbitrary, we proved $\boldsymbol{\delta}_{\mathrm{LCS}}(x, y) \leq 2n\gamma_4 + \min_{\Lambda\in\mathcal{S}_{n,m}} \mathbf{c}(\Lambda)$, as desired.

It remains to prove the lower bound of (4.3), i.e., $\boldsymbol{\delta}_{\mathrm{LCS}}(x, y) \geq 2n\gamma_4 + \min_{\Lambda\in\boldsymbol{\Lambda}_{n,m}} \mathbf{c}(\Lambda)$. As in Fact 4.3.4, let $x(L^{\mathrm{Y}})$, $x(\mathrm{G}(y_j))$ for $j \in [m]$, $x(Z_j^{\mathrm{Y}})$ for $j \in [m-1]$, $x(R^{\mathrm{Y}})$ be an ordered partition of x such that

$$\boldsymbol{\delta}_{\mathrm{LCS}}(x, y) = \boldsymbol{\delta}_{\mathrm{LCS}}(x(L^{\mathrm{Y}}), L^{\mathrm{Y}}) + \boldsymbol{\delta}_{\mathrm{LCS}}(x(R^{\mathrm{Y}}), R^{\mathrm{Y}}) + \sum_{j=1}^{m} \boldsymbol{\delta}_{\mathrm{LCS}}(x(\mathrm{G}(y_j)), \mathrm{G}(y_j)) + \sum_{j=1}^{m-1} \boldsymbol{\delta}_{\mathrm{LCS}}(x(Z_j^{\mathrm{Y}}), Z_j^{\mathrm{Y}}).$$

Clearly, we can bound $\boldsymbol{\delta}_{\mathrm{LCS}}(x(Z_j^{\mathrm{Y}}), Z_j^{\mathrm{Y}}) \geq 0$. Since $L^{\mathrm{Y}} = 0^{n\gamma_4}$ and $x(L^{\mathrm{Y}})$ is a prefix of x, by Claim 4.3.5 we have $\boldsymbol{\delta}_{\mathrm{LCS}}(x(L^{\mathrm{Y}}), L^{\mathrm{Y}}) \geq n\gamma_4$, and similarly we get $\boldsymbol{\delta}_{\mathrm{LCS}}(x(R^{\mathrm{Y}}), R^{\mathrm{Y}}) \geq n\gamma_4$. It remains to construct an alignment $\Lambda \in \boldsymbol{\Lambda}_{n,m}$ satisfying

$$\mathbf{c}(\Lambda) \leq \sum_{j=1}^{m} \boldsymbol{\delta}_{\mathrm{LCS}}(x(\mathrm{G}(y_j)), \mathrm{G}(y_j)), \tag{4.4}$$

then together we have shown the desired inequality $\boldsymbol{\delta}_{\mathrm{LCS}}(x, y) \geq 2n\gamma_4 + \min_{\Lambda\in\boldsymbol{\Lambda}_{n,m}} \mathbf{c}(\Lambda)$.

Let us construct such an alignment Λ. For any $j \in [m]$, if $x(\mathrm{G}(y_j))$ contains more than half of some $x_{i'}$ (which is part of $\mathrm{G}(x_{i'})$), then let i be the leftmost such index and align i and j. Note that the set Λ of all these aligned pairs (i, j) is a valid alignment in $\boldsymbol{\Lambda}_{n,m}$, since no x_i or y_j can be aligned more than once.

Since by definition, we have $\mathbf{c}(\Lambda) = \sum_{(i,j)\in\Lambda} \boldsymbol{\delta}_{\mathrm{LCS}}(x_i, y_j) + (m - |\Lambda|) \max_{i,j} \boldsymbol{\delta}_{\mathrm{LCS}}(x_i, y_j)$ and since $\max_{i,j} \boldsymbol{\delta}_{\mathrm{LCS}}(x_i, y_j) \leq \max_{i,j}(|x_i| + |y_j|) = \ell_{\mathrm{X}} + \ell_{\mathrm{Y}}$, in order to show (4.4) it suffices to prove the following two claims.

Claim 4.3.6. *For any aligned pair $(i, j) \in \Lambda$, it holds that $\boldsymbol{\delta}_{\mathrm{LCS}}(x(\mathrm{G}(y_j)), \mathrm{G}(y_j)) \geq \boldsymbol{\delta}_{\mathrm{LCS}}(x_i, y_j)$.*

Proof. Recall that $x(\mathrm{G}(y_j))$ contains more than half of x_i. First consider the case that $x(\mathrm{G}(y_j))$ touches not only $\mathrm{G}(x_i)$ but also $\mathrm{G}(x_{i'})$ for some $i' \neq i$. As between x_i and $\mathrm{G}(x_{i'})$ there is at least one block of zeroes 0^{γ_3} and half of the guarding of $\mathrm{G}(x_i)$ (i.e., $1^{\gamma_2}0^{\gamma_1}$ or $0^{\gamma_1}1^{\gamma_2}$), we obtain $|x(\mathrm{G}(y_j))| \geq |0^{\gamma_3}| + |1^{\gamma_2}0^{\gamma_1}| = \gamma_3 + \gamma_2 + \gamma_1$. Thus, any matching of $x(\mathrm{G}(y_j))$ and $\mathrm{G}(y_j)$ leaves at least $|x(\mathrm{G}(y_j))| - |\mathrm{G}(y_j)| \geq (\gamma_3 + \gamma_2 + \gamma_1) - (2\gamma_2 + 2\gamma_1 + \ell_{\mathrm{Y}}) = \gamma_3 - \gamma_2 - \gamma_1 - \ell_{\mathrm{Y}} \geq \ell_{\mathrm{X}} + \ell_{\mathrm{Y}}$ unmatched symbols, implying $\boldsymbol{\delta}_{\mathrm{LCS}}(x(\mathrm{G}(y_j)), \mathrm{G}(y_j)) \geq \ell_{\mathrm{X}} + \ell_{\mathrm{Y}} \geq \boldsymbol{\delta}_{\mathrm{LCS}}(x_i, y_j)$.

Now consider the remaining case, where $x(\mathrm{G}(y_j))$ touches no other $\mathrm{G}(x_{i'})$. In this case, $x(\mathrm{G}(y_j))$ is a substring of $0^{\gamma_3}\mathrm{G}(x_i)0^{\gamma_3}$, i.e., we can write $x(\mathrm{G}(y_j))$ as $0^{h_L}z0^{h_R}$, where z is a substring of $\mathrm{G}(x_i)$. Since $\mathrm{G}(y_j)$ starts with γ_2 ones, by Fact 3.5.3(iii) we have $\boldsymbol{\delta}_{\mathrm{LCS}}(x(\mathrm{G}(y_j)), \mathrm{G}(y_j)) \geq \min\{\gamma_2, \boldsymbol{\delta}_{\mathrm{LCS}}(z0^{h_R}, \mathrm{G}(y_j))\}$. Since $\gamma_2 \geq \ell_{\mathrm{X}} + \ell_{\mathrm{Y}} \geq \boldsymbol{\delta}_{\mathrm{LCS}}(x_i, y_j)$, it suffices to bound $\boldsymbol{\delta}_{\mathrm{LCS}}(z0^{h_R}, \mathrm{G}(y_j))$ from below. By a symmetric argument, we eliminate the block 0^{h_R} and only have to bound $\boldsymbol{\delta}_{\mathrm{LCS}}(z, \mathrm{G}(y_j))$ from below. We can assume that $|z| > |\mathrm{G}(y_j)| - \gamma_2$, since otherwise $\boldsymbol{\delta}_{\mathrm{LCS}}(z, \mathrm{G}(y_j)) \geq \gamma_2 \geq \ell_{\mathrm{X}} + \ell_{\mathrm{Y}} \geq \boldsymbol{\delta}_{\mathrm{LCS}}(x_i, y_j)$. Thus, we have $\mathrm{G}(y_j) = 1^{\gamma_2}0^{\gamma_1}y_j0^{\gamma_1}1^{\gamma_2}$ and can write z as $1^{r_L}0^{\gamma_1}x_i0^{\gamma_1}1^{r_R}$ with $r_L, r_R > 0$. By Fact 3.5.3(i) we have $\boldsymbol{\delta}_{\mathrm{LCS}}(z, \mathrm{G}(y_j)) = \boldsymbol{\delta}_{\mathrm{LCS}}(0^{\gamma_1}x_i0^{\gamma_1}1^{r_R}, 1^{\gamma_2 - r_L}0^{\gamma_1}y_j0^{\gamma_1}1^{\gamma_2})$. By Fact 3.5.3(iii), this yields $\boldsymbol{\delta}_{\mathrm{LCS}}(z, \mathrm{G}(y_j)) \geq \min\{\gamma_1, \boldsymbol{\delta}_{\mathrm{LCS}}(0^{\gamma_1}x_i0^{\gamma_1}1^{r_R}, 0^{\gamma_1}y_j0^{\gamma_1}1^{\gamma_2})\}$, and

since $\gamma_1 \geq \ell_X + \ell_Y \geq \boldsymbol{\delta}_{\mathrm{LCS}}(x_i, y_j)$ it suffices to bound the latter term. By a symmetric argument we eliminate the ones on the right side, and it suffices to bound $\boldsymbol{\delta}_{\mathrm{LCS}}(0^{\gamma_1}x_i0^{\gamma_1}, 0^{\gamma_1}y_j0^{\gamma_1})$. Using Fact 3.5.3(i) twice, this is equal to $\boldsymbol{\delta}_{\mathrm{LCS}}(x_i, y_j)$. Hence, we have shown the desired inequality $\boldsymbol{\delta}_{\mathrm{LCS}}(x(\mathrm{G}(y_j)), \mathrm{G}(y_j)) \geq \boldsymbol{\delta}_{\mathrm{LCS}}(x_i, y_j)$. □

Claim 4.3.7. *If j is unaligned in Λ, then $\boldsymbol{\delta}_{\mathrm{LCS}}(x(\mathrm{G}(y_j)), \mathrm{G}(y_j)) \geq \ell_X + \ell_Y$.*

Proof. Since $x(\mathrm{G}(y_j))$ contains less than half of any x_i, examining the structure of x we see that $x(\mathrm{G}(y_j))$ is a substring[3] of $P := x_i0^{\gamma_1}1^{\gamma_2}0^{\gamma_3}1^{\gamma_2}0^{\gamma_1}x_{i+1}$ for some $1 \leq i < n$, where at most half of x_i and x_{i+1} can be part of $x(\mathrm{G}(y_j))$. If $x(\mathrm{G}(y_j))$ contains ones to the left and to the right of 0^{γ_3} in P, then $x(\mathrm{G}(y_j))$ contains at least γ_3 zeroes. Since $\mathrm{G}(y_j)$ contains $2\gamma_1 + \ell_Y - s_Y \leq 2\gamma_1 + \ell_Y$ zeroes, at most $2\gamma_1 + \ell_Y$ zeroes of $x(\mathrm{G}(y_j))$ can be matched, leaving at least $\gamma_3 - 2\gamma_1 - \ell_Y$ unmatched zeroes. Thus, $\boldsymbol{\delta}_{\mathrm{LCS}}(x(\mathrm{G}(y_j)), \mathrm{G}(y_j)) \geq \gamma_3 - 2\gamma_1 - \ell_Y \geq \ell_X + \ell_Y$. Otherwise, if $x(\mathrm{G}(y_j))$ contains only ones to the left of 0^{γ_3} in P (or only to the right), then $x(\mathrm{G}(y_j))$ contains at most $\gamma_2 + \ell_X$ ones. Thus, among the $2\gamma_2 + s_Y \geq 2\gamma_2$ ones of $\mathrm{G}(y_j)$ at least $\gamma_2 - \ell_X$ ones remain unmatched, implying $\boldsymbol{\delta}_{\mathrm{LCS}}(x(\mathrm{G}(y_j)), \mathrm{G}(y_j)) \geq \gamma_2 - \ell_X \geq \ell_X + \ell_Y$. □

This finishes the proof of Lemma 4.3.3. □

Remark 4.3.8. Our construction of the alignment gadget provides some liberty to adapt its parameters. E.g., prepending or appending further zeroes to y does not change the length of the LCS, i.e., $L(x, 0^\beta y 0^{\beta'}) = L(x, y)$ for all $\beta, \beta' \geq 0$. This property might be useful for future applications of the alignment gadget. In particular, in Chapter 5, we exploit this property to give our hardness proofs for binary alphabets.

4.4 Edit Distance

To classify the complexity of all cost choices for the edit distance, we first identify the trivial cases of $\mathrm{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$ that can be solved in constant time. For all other cases, we present a reduction from $\mathrm{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$ to $\mathrm{Edit}(c'_{\text{subst}}) = \mathrm{Edit}(1, 1, 0, c'_{\text{subst}})$ and vice versa, over binary alphabet, in Section 4.4.1. Then in Section 4.4.2, we prove a conditional lower bound of $\mathcal{O}(m^{2-\varepsilon})$ for $\mathrm{Edit}(c_{\text{subst}})$ by applying our alignment gadget framework.

4.4.1 Equivalences of Edit Distance Variants

All of our reductions between variants of the edit distance are of the following form. Consider two arbitrary cost variants $E_1 = \mathrm{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$ and $E_2 = \mathrm{Edit}(c'_{\text{del-x}}, c'_{\text{del-y}}, c'_{\text{match}}, c'_{\text{subst}})$ and denote the cost of any traversal T with respect to E_i by $\boldsymbol{\delta}_{E_i}(T)$. We say that E_1 and E_2 are *equivalent*, if there are constants α, β such that for any traversal T we have $\boldsymbol{\delta}_{E_1}(T) = \alpha \cdot \boldsymbol{\delta}_{E_2}(T) + \beta$. Then the complexity of computing E_1 and E_2 is asymptotically equal.

Lemma 4.4.1. $\mathrm{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$ *can be solved in constant time if* $c_{\text{subst}} = c_{\text{match}}$ *or* $c_{\text{del-x}} + c_{\text{del-y}} \leq \min\{c_{\text{match}}, c_{\text{subst}}\}$. *Otherwise,* $\mathrm{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$ *on binary strings is equivalent to* $\mathrm{Edit}(c'_{\text{subst}})$ *on binary strings for some* $0 < c'_{\text{subst}} \leq 2$.

[3] Actually $x(\mathrm{G}(y_j))$ could also be a substring of $1^{\gamma_2}0^{\gamma_1}x_1$ or of $x_n0^{\gamma_1}1^{\gamma_2}$. We treat these border cases by setting $x_0 := x_1$ and $x_{n+1} := x_n$ and letting from now on $0 \leq i \leq n$.

Note that by this lemma, hardness for general rational cost parameters follows by proving hardness of $\mathrm{Edit}(c'_{\mathrm{subst}})$ for $0 < c'_{\mathrm{subst}} \leq 2$. We remark that a characterization as in Lemma 4.4.1 (for the case of $c_{\mathrm{match}} = 0$) has also been obtained in [RBN97].

Proof of Lemma 4.4.1. Let x, y be strings of length n, m. By symmetry, we may assume that $n \geq m$. Observe that we can write the cost of any traversal T with respect to $\mathrm{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\mathrm{match}}, c_{\mathrm{subst}})$ as

$$\boldsymbol{\delta}_{\mathrm{Edit}}(T) = A \cdot c_{\mathrm{match}} + B \cdot c_{\mathrm{subst}} + C \cdot (c_{\text{del-x}} + c_{\text{del-y}}) + (n - m) \cdot c_{\text{del-x}},$$

for some $A, B, C \geq 0$ with $A + B + C = m$, since matchings and substitutions touch as many symbols in x as in y, so that we need exactly $n - m$ more deletions in x than deletions in y.

If $c_{\text{del-x}} + c_{\text{del-y}} \leq \min\{c_{\mathrm{match}}, c_{\mathrm{subst}}\}$, then we can replace any matching or substitution by a deletion in x and a deletion in y without increasing the cost. Thus, an optimal traversal has $C = m$ and minimal cost $n \cdot c_{\text{del-x}} + m \cdot c_{\text{del-y}}$, which can be computed in constant time. Similarly, if $c_{\mathrm{match}} = c_{\mathrm{subst}}$, then the minimal cost is independent of the symbols in x and y. We may arbitrarily set $A + B$ and C subject to $A + B + C = m$ and $A + B, C \geq 0$; the resulting minimal cost is $m \cdot \min\{c_{\mathrm{match}}, c_{\text{del-x}} + c_{\text{del-y}}\} + (n - m)c_{\text{del-x}}$, which again can be computed in constant time.

Now assume that $c_{\mathrm{match}} \neq c_{\mathrm{subst}}$ and $c_{\text{del-x}} + c_{\text{del-y}} > \min\{c_{\mathrm{match}}, c_{\mathrm{subst}}\}$. Restricting our attention to binary strings, by flipping all symbols in y but not in x we can swap the costs of matching and substitution. Thus, we may assume that $c_{\mathrm{subst}} > c_{\mathrm{match}}$ (and $c_{\text{del-x}} + c_{\text{del-y}} > c_{\mathrm{match}}$). We set

$$c'_{\mathrm{subst}} := \alpha(c_{\mathrm{subst}} - c_{\mathrm{match}}), \qquad \text{where} \quad \alpha := \tfrac{2}{c_{\text{del-x}} + c_{\text{del-y}} - c_{\mathrm{match}}}.$$

One can easily verify that for any traversal T with cost $\boldsymbol{\delta}_{\mathrm{Edit}}(T) = A \cdot c_{\mathrm{match}} + B \cdot c_{\mathrm{subst}} + C \cdot (c_{\text{del-x}} + c_{\text{del-y}}) + (n - m) \cdot c_{\text{del-x}}$ (with respect to $\mathrm{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\mathrm{match}}, c_{\mathrm{subst}})$) we have

$$\alpha\boldsymbol{\delta}_{\mathrm{Edit}}(T) - \alpha m \cdot c_{\mathrm{match}} + (n - m)(1 - \alpha c_{\text{del-x}}) = B \cdot c'_{\mathrm{subst}} + C \cdot 2 + (n - m).$$

Observe that the latter is the cost of T with respect to $\mathrm{Edit}(c'_{\mathrm{subst}})$. Hence, this proves that $\mathrm{Edit}(c'_{\mathrm{subst}})$ is equivalent to $\mathrm{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\mathrm{match}}, c_{\mathrm{subst}})$. Finally, note that $c'_{\mathrm{subst}} > 0$. If $c'_{\mathrm{subst}} > 2$, then we can replace it by 2 without changing the cost of the optimal traversal, since we can replace any substitution (of cost 2) by a deletion and an insertion (both of cost 1). This yields $0 < c'_{\mathrm{subst}} \leq 2$. □

Finally, particularly to allow the algorithm given in Section 6.1 to handle also non-integer cost choices, we provide a simple reduction from rational cost choices to integer cost choices.

Fact 4.4.2. $\mathrm{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\mathrm{match}}, c_{\mathrm{subst}})$ *(with rational cost parameters* $c_{\text{del-x}}$*,* $c_{\text{del-y}}$*,* c_{match} *and* c_{subst}*) is equivalent to* $\mathrm{Edit}(c'_{\text{del-x}}, c'_{\text{del-y}}, c'_{\mathrm{match}}, c'_{\mathrm{subst}})$ *for some positive integers* $c'_{\text{del-x}}$*,* $c'_{\text{del-y}}$*,* c'_{match} *and* c'_{subst}*.*

Proof. Since we always assume all operation costs to be rationals, without loss of generality $c_{\text{del-x}}, c_{\text{del-y}}, c_{\mathrm{match}}$ and c_{subst} have a common denominator D. We obtain positive integral operation costs by setting $c'_{\text{del-x}} := Dc_{\text{del-x}} + M$, $c'_{\text{del-y}} := Dc_{\text{del-y}} + M$, $c'_{\mathrm{match}} := Dc_{\mathrm{match}} + 2M$, $c'_{\mathrm{subst}} := Dc_{\mathrm{subst}} + 2M$ for a sufficiently large integer M. Both

variants are equivalent, since $\boldsymbol{\delta}_{\mathrm{Edit}}(T)$ is changed to

$$D\boldsymbol{\delta}_{\mathrm{Edit}}(T) + m \cdot 2M + (n-m) \cdot M. \qquad \square$$

4.4.2 Hardness Proof

In this section, we study the edit distance with matching cost 0, deletion and insertion cost 1, and substitution cost $0 < c_{\mathrm{subst}} \leq 2$. As for LCS, we consider the set of inputs $\mathcal{I} := \bigcup_{k\geq 0}\{0,1\}^k$ and define the type of a binary string $x \in \mathcal{I}$ as $\mathrm{type}(x) := (|x|, \sum_i x[i])$.

Lemma 4.4.3. $\mathrm{Edit}(c_{\mathrm{subst}})$ *admits coordinate values by setting*

$$\mathbf{1}_{\mathrm{X}} := 11100,\ \mathbf{0}_{\mathrm{X}} := 10011,\ \mathbf{1}_{\mathrm{Y}} := 00111,\ \mathbf{0}_{\mathrm{Y}} := 11001.$$

Proof. All four strings have the same length and the same number of ones, so they have equal type. Using Fact 3.5.2(i), we have $\boldsymbol{\delta}_{\mathrm{Edit}}(\mathbf{0}_{\mathrm{X}}, \mathbf{0}_{\mathrm{Y}}) = \boldsymbol{\delta}_{\mathrm{Edit}}(10011, 11001) = \boldsymbol{\delta}_{\mathrm{Edit}}(0011, 1001) = \boldsymbol{\delta}_{\mathrm{Edit}}(001, 100)$. Depending on the value of c_{subst}, the optimal traversal of $(001, 100)$ is either to delete both ones or to substitute the first and last symbols. This yields $\boldsymbol{\delta}_{\mathrm{Edit}}(001, 100) = \min\{2, 2c_{\mathrm{subst}}\}$. Similarly, we obtain $\boldsymbol{\delta}_{\mathrm{Edit}}(\mathbf{1}_{\mathrm{X}}, \mathbf{0}_{\mathrm{Y}}) = \boldsymbol{\delta}_{\mathrm{Edit}}(\mathbf{0}_{\mathrm{X}}, \mathbf{1}_{\mathrm{Y}}) = \boldsymbol{\delta}_{\mathrm{Edit}}(\mathbf{0}_{\mathrm{X}}, \mathbf{0}_{\mathrm{Y}}) = \min\{2, 2c_{\mathrm{subst}}\}$ and $\boldsymbol{\delta}_{\mathrm{Edit}}(\mathbf{1}_{\mathrm{X}}, \mathbf{1}_{\mathrm{Y}}) = \boldsymbol{\delta}_{\mathrm{Edit}}(11100, 00111) = \min\{4, 4c_{\mathrm{subst}}\}$. Hence, $\boldsymbol{\delta}_{\mathrm{Edit}}(\mathbf{1}_{\mathrm{X}}, \mathbf{1}_{\mathrm{Y}}) > \boldsymbol{\delta}_{\mathrm{Edit}}(\mathbf{1}_{\mathrm{X}}, \mathbf{0}_{\mathrm{Y}}) = \boldsymbol{\delta}_{\mathrm{Edit}}(\mathbf{0}_{\mathrm{X}}, \mathbf{1}_{\mathrm{Y}}) = \boldsymbol{\delta}_{\mathrm{Edit}}(\mathbf{0}_{\mathrm{X}}, \mathbf{0}_{\mathrm{Y}})$. $\square$

Definition 4.4.4. *Consider instances $x_1, \ldots, x_n \in \mathcal{I}_{t_{\mathrm{X}}}$ and $y_1, \ldots, y_m \in \mathcal{I}_{t_{\mathrm{Y}}}$ with $n \geq m$ and types $t_{\mathrm{X}} = (\ell_{\mathrm{X}}, s_{\mathrm{X}}), t_{\mathrm{Y}} = (\ell_{\mathrm{Y}}, s_{\mathrm{Y}})$. We define the parameters $\rho := 2\lceil 1/c_{\mathrm{subst}} \rceil$, $\gamma_1 := 10\rho(\ell_{\mathrm{X}} + \ell_{\mathrm{Y}})$, $\gamma_2 := 6\rho\gamma_1 + 5s_{\mathrm{X}} - \ell_{\mathrm{X}}$, and $\gamma_3 := 2\gamma_2$ (since c_{subst} is constant, these parameters are $\Theta(\ell_{\mathrm{X}} + \ell_{\mathrm{Y}})$).*

To guard *a string by blocks of zeroes and ones, we set* $\mathrm{G}(z) := (1^{\gamma_1}0^{\gamma_1})^{\rho} z (0^{\gamma_1}1^{\gamma_1})^{\rho}$. *Now the alignment gagdet is*

$$\begin{aligned} x &:= \quad \mathrm{G}(x_1)\, 0^{\gamma_2}\, \mathrm{G}(x_2)\, 0^{\gamma_2} \ \ldots\ \mathrm{G}(x_{n-1})\, 0^{\gamma_2}\, \mathrm{G}(x_n), \\ y &:= 0^{n\gamma_3}\, \mathrm{G}(y_1)\, 0^{\gamma_2}\, \mathrm{G}(y_2)\, 0^{\gamma_2} \ \ldots\ \mathrm{G}(y_{m-1})\, 0^{\gamma_2}\, \mathrm{G}(y_m)\, 0^{n\gamma_3}. \end{aligned}$$

Let us provide some intuition on the more complex guarding $\mathrm{G}(z)$ (compared to the simpler guarding for LCS): Consider a block $B = (1^{\gamma}0^{\gamma})^{\rho}$. Clearly, B can be completely matched to B, resulting in a cost of 0. Consider a slight perturbation B' of B by prepending Δ ones and deleting the last Δ zeroes. Then the edit distance of B and B' is at most 2Δ, since we may delete the prepended ones in B' and the additional zeroes at the end of B. Another upper bound for the edit distance of B and B' is $2\rho \cdot \Delta c_{\mathrm{subst}}$, since we may match the first γ ones, then substitute the next Δ symbols, then match the next $\gamma - \Delta$ zeroes, and so on. By choosing $\rho := 2\lceil 1/c_{\mathrm{subst}} \rceil$, the traversal using substitutions is more expensive, and indeed we prove that then the edit distance is at least 2Δ. This provides a building block which avoids substitutions and where slight perturbations are severely punished. Thus, our guarding $\mathrm{G}(z) = (1^{\gamma_1}0^{\gamma_1})^{\rho} z (0^{\gamma_1}1^{\gamma_1})^{\rho}$ ensures that an optimal traversal of $\mathrm{G}(x)$ and $\mathrm{G}(y)$ aligns x and y, and this also holds after small perturbations.

Lemma 4.4.5. *For any $0 < c_{\mathrm{subst}} \leq 2$, Definition 4.4.4 realizes an alignment gadget for* $\mathrm{Edit}(c_{\mathrm{subst}})$.

Thus, Theorem 4.1.3 is applicable, implying a lower bound of $\mathcal{O}(m^{2-\varepsilon})$ for $\mathrm{Edit}(c_{\mathrm{subst}})$. Combining this with Lemma 4.4.1 proves Theorem 2.1.2. We remark that our construction

is not an *unbalanced* alignment gadget, as the length of y grows linearly in n, not necessarily in m. Thus, we do not obtain a conditional lower bound of $\mathcal{O}((nm)^{1-\varepsilon})$ (i.e., not for $m \approx n^\alpha$ for all $0 < \alpha < 1$), which in fact is ruled out by the algorithmic result of Theorem 2.1.4.

To prepare the proof of Lemma 4.4.5, we recall the following observation proven in Chapter 3.

Fact 3.5.2. *Let x, y, z be arbitrary strings and $\ell, k \in \mathbb{N}_0$. Then we have*

(i) $\boldsymbol{\delta}_{\mathrm{Edit}}(1^k x, 1^k y) = \boldsymbol{\delta}_{\mathrm{Edit}}(x, y)$,

(ii) $\boldsymbol{\delta}_{\mathrm{Edit}}(x, y) \geq \big||x| - |y|\big|$, *and*

(iii) $|\boldsymbol{\delta}_{\mathrm{Edit}}(xz, y) - \boldsymbol{\delta}_{\mathrm{Edit}}(x, y)| \leq |z|$.

We obtain symmetric statements of (i) by replacing all 1's by 0's and by reversing all involved strings.

Furthermore, we need the following technical lemma.

Fact 4.4.6. *Let $\ell, m, r \geq 0$. Then for any $x \in \{0^\ell 1^m 0^r, 1^{m-\ell-r}, 1^{m-\ell}0^r, 0^\ell 1^{m-r}\}$ we have $\boldsymbol{\delta}_{\mathrm{Edit}}(x, 1^m) \geq |\ell - r| + c_{\mathrm{subst}} \cdot \min\{\ell, r\}$.*

Proof. Fact 3.5.2(ii) yields $\boldsymbol{\delta}_{\mathrm{Edit}}(0^\ell 1^m 0^r, 1^m), \boldsymbol{\delta}_{\mathrm{Edit}}(1^{m-\ell-r}, 1^m) \geq \ell + r \geq |\ell - r| + c_{\mathrm{subst}} \cdot \min\{\ell, r\}$, since $c_{\mathrm{subst}} \leq 2$. For $x = 0^\ell 1^{m-r}$, consider any optimal traversal T. If T substitutes s zeroes and deletes the remaining $\ell - s$ zeroes, then $\boldsymbol{\delta}_{\mathrm{Edit}}(0^\ell 1^{m-r}, 1^m) = c_{\mathrm{subst}} \cdot s + (\ell - s) + \boldsymbol{\delta}_{\mathrm{Edit}}(1^{m-r}, 1^{m-s})$. By Fact 3.5.2(i), $\boldsymbol{\delta}_{\mathrm{Edit}}(1^{m-r}, 1^{m-s}) = \boldsymbol{\delta}_{\mathrm{Edit}}(\epsilon, 1^{|r-s|}) = |r - s|$, where ϵ is the empty string. Hence, $\boldsymbol{\delta}_{\mathrm{Edit}}(0^\ell 1^{m-r}, 1^m) = \min_{0 \leq s \leq \ell}\{c_{\mathrm{subst}} \cdot s + \ell - s + |r - s|\}$. A short case analysis shows that this term is minimized for $s = \min\{\ell, r\}$, where it evaluates to $c_{\mathrm{subst}} \cdot \min\{\ell, r\} + \ell + r - 2\min\{\ell, r\} = c_{\mathrm{subst}} \cdot \min\{\ell, r\} + |\ell - r|$. The case $x = 1^{m-\ell}0^r$ is symmetric. □

Recall that for a string y and indices $a \leq b$, we denote the substring from $y[a]$ to $y[b]$ by $y[a..b]$, and let us restate the following fact from Chapter 3.

Fact 3.5.1 (Optimal Partitions)**.** *Let x and $y_1, \ldots, y_k$ be arbitrary strings. Set $y = y_1 \ldots y_k$. Then we have*

$$\boldsymbol{\delta}_{\mathrm{Edit}}(x, y) = \min_{x(y_1), \ldots, x(y_k)} \sum_{j=1}^{k} \boldsymbol{\delta}_{\mathrm{Edit}}(x(y_j), y_j),$$

where $x(y_1), \ldots, x(y_k)$ ranges over all ordered partitions *of x into k substrings, i.e., $x(y_1) = x[i_0 + 1..i_1], x(y_2) = x[i_1 + 1..i_2], \ldots, x(y_k) = x[i_{k-1} + 1..i_k]$ for any $0 = i_0 \leq i_1 \leq \ldots \leq i_k = |x|$.*

Proof of Lemma 4.4.5. From now on let x, y be as in Definition 4.4.4. Observe that indeed x only depends on m, t_{Y}, and $x_1, \ldots, x_n$, and type(x) only depends on n, m, t_{X}, and t_{Y}, and similarly for y. Moreover, x and y can clearly be constructed in time $\mathcal{O}((n + m)(\ell_{\mathrm{X}} + \ell_{\mathrm{Y}}))$, where $\ell_{\mathrm{X}} = |x_1| = \ldots = |x_n|$ and $\ell_{\mathrm{Y}} = |y_1| = \ldots = |y_m|$.

It remains to prove that for some C, we have

$$\min_{\Lambda \in \boldsymbol{\Lambda}_{n,m}} \mathbf{c}(\Lambda) \leq \boldsymbol{\delta}_{\mathrm{Edit}}(x, y) - C \leq \min_{\Lambda \in \mathcal{S}_{n,m}} \mathbf{c}(\Lambda). \tag{4.5}$$

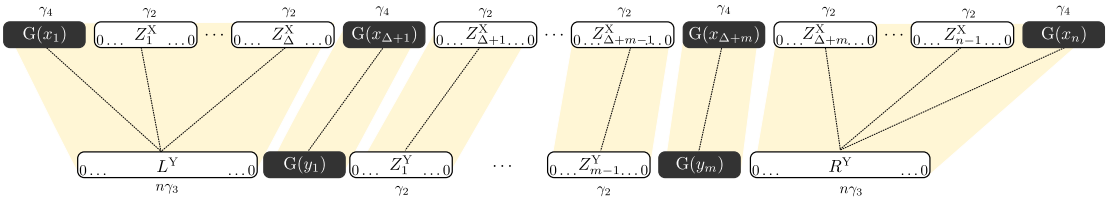


FIGURE 4.7: Optimal traversal corresponding to structured alignment $\Lambda = \{(\Delta + j, j) \mid j \in [m]\} \in \mathcal{S}_{n,m}$.

We set

$$C := 2n\gamma_3 - \beta(n-m)(\gamma_4 + \gamma_2),$$

where

$$\beta := 1 - c_{\text{subst}}/5 \quad \text{and} \quad \gamma_4 := 4\rho\gamma_1 + \ell_{\text{X}}.$$

Note that γ_4 is the length of $\text{G}(x_i)$.

Let us give names to the substrings consisting only of zeroes in x and y. In x, we denote the 0^{γ_2}-block after $\text{G}(x_i)$ by Z_i^{X}, $i \in [n-1]$. In y, we denote the 0^{γ_2}-block after $\text{G}(y_j)$ by Z_j^{Y}, $j \in [m-1]$. Moreover, we denote the prefix $0^{n\gamma_3}$ by L^{Y} and the suffix $0^{n\gamma_3}$ by R^{Y}.

We first prove the crucial property that for any prefix x' of x the distance $\boldsymbol{\delta}_{\text{Edit}}(x', L^{\text{Y}})$ is essentially $|L^{\text{Y}}| - \beta|x'| = n\gamma_3 - \beta|x'|$. This is due to a careful choice of the parameters γ_1, γ_2, ρ.

Claim 4.4.7. *For any prefix x' of x we have $\boldsymbol{\delta}_{\text{Edit}}(x', L^{\text{Y}}) \geq |L^{\text{Y}}| - \beta|x'| = n\gamma_3 - \beta|x'|$, with equality if x' is of the form $\text{G}(x_1)0^{\gamma_2} \dots \text{G}(x_i)0^{\gamma_2}$ for any $0 \leq i < n$. Symmetric statements hold for $\boldsymbol{\delta}_{\text{Edit}}(x'', R^{\text{Y}})$ where x'' is any suffix of x.*

Proof. The parameter γ_3 is chosen such that $|x'| \leq |x| \leq |L^{\text{Y}}|$: Indeed, $|x| \leq n(4\rho\gamma_1 + \ell_{\text{X}} + \gamma_2) \leq n \cdot 2\gamma_2 \leq n\gamma_3 = |L^{\text{Y}}|$. Observe that all zeroes of x' can be matched to zeroes of L^{Y}, while all ones of x' have to be substituted. The remaining zeroes of L^{Y} have to be deleted. Denoting the number of ones in x' by ℓ, we obtain $\boldsymbol{\delta}_{\text{Edit}}(x', L^{\text{Y}}) = \ell \cdot c_{\text{subst}} + (|L^{\text{Y}}| - |x'|)$. We will show $\ell \geq |x'|/5$, with equality if x' has the special form as in the statement. In other words, the *relative number of ones* $\ell/|x'|$ is at least $1/5$, with equality if x' has the special form. This implies $\boldsymbol{\delta}_{\text{Edit}}(x', L^{\text{Y}}) \geq n\gamma_3 - \beta|x'|$, with equality if x' has the special form.

Note that each x_i has length ℓ_{X} and contains s_{X} ones, and consequently, $\text{G}(x_i)Z_i^{\text{X}} = (1^{\gamma_1}0^{\gamma_1})^{\rho}x_i(0^{\gamma_1}1^{\gamma_1})^{\rho}0^{\gamma_2}$ contains $2\rho\gamma_1 + (\ell_{\text{X}} - s_{\text{X}}) + \gamma_2$ zeroes and $2\rho\gamma_1 + s_{\text{X}}$ ones. The parameter γ_2 is chosen so that the number of zeroes is four times the number of ones, implying that the relative number of ones is $1/5$. Note that any prefix of $(1^{\gamma_1}0^{\gamma_1})^{\rho}$ has relative number of ones at least $1/2$. Since $x_i0^{\gamma_1}$ has less than $2\gamma_1$ zeroes and $|(1^{\gamma_1}0^{\gamma_1})^{\rho}| \geq 2\gamma_1$, any prefix of $(1^{\gamma_1}0^{\gamma_1})^{\rho}x_i0^{\gamma_1}$ has relative number of ones at least $1/4$. Since any prefix of $1^{\gamma_1}(0^{\gamma_1}1^{\gamma_1})^{\rho-1}$ has relative number of ones at least $1/2$, any prefix of $(1^{\gamma_1}0^{\gamma_1})^{\rho}x_i(0^{\gamma_1}1^{\gamma_1})^{\rho}$ has relative number of ones at least $1/4$. The relative number of ones decreases by adding any prefix of 0^{γ_2}, however, for the final string $(1^{\gamma_1}0^{\gamma_1})^{\rho}x_i(0^{\gamma_1}1^{\gamma_1})^{\rho}0^{\gamma_2}$, we already argued that the relative number of ones is $1/5$. This shows that the relative number of ones of any prefix of x is at least $1/5$. □

We now show the upper bound of (4.5), i.e., we prove that $\boldsymbol{\delta}_{\text{Edit}}(x, y) \leq C + \min_{\Lambda \in \mathcal{S}_{n,m}} \sum_{(i,j)\in\Lambda} \boldsymbol{\delta}_{\text{Edit}}(x_i, y_j)$. To this end, consider a structured alignment $\Lambda =$

$\{(\Delta+1,1),\dots,(\Delta+m,m)\} \in \mathcal{S}_{n,m}$. We construct an ordered partition of x as in Fact 3.5.1 by setting, as illustrated in Figure 4.7,

$$\begin{aligned} x(\mathrm{G}(y_j)) &:= \mathrm{G}(x_{\Delta+j}) \qquad \text{for } j \in [m],\\ x(Z^{\mathrm{Y}}_j) &:= Z^{\mathrm{X}}_{\Delta+j} \qquad \text{for } j \in [m-1],\\ x(L^{\mathrm{Y}}) &:= \mathrm{G}(x_1)Z^{\mathrm{X}}_1 \dots \mathrm{G}(x_\Delta)Z^{\mathrm{X}}_\Delta,\\ x(R^{\mathrm{Y}}) &:= Z^{\mathrm{X}}_{\Delta+m}\mathrm{G}(x_{\Delta+m+1}) \dots Z^{\mathrm{X}}_{n-1}\mathrm{G}(x_n). \end{aligned}$$

Note that indeed these strings partition x and y, respectively. Thus, Fact 3.5.1 yields

$$\boldsymbol{\delta}_{\mathrm{Edit}}(x,y) \le \boldsymbol{\delta}_{\mathrm{Edit}}(x(L^{\mathrm{Y}}),L^{\mathrm{Y}}) + \boldsymbol{\delta}_{\mathrm{Edit}}(x(R^{\mathrm{Y}}),R^{\mathrm{Y}}) + \sum_{j=1}^{m} \boldsymbol{\delta}_{\mathrm{Edit}}(\mathrm{G}(x_{\Delta+j}),\mathrm{G}(y_j)) + \sum_{j=1}^{m-1} \boldsymbol{\delta}_{\mathrm{Edit}}(Z^{\mathrm{X}}_{\Delta+j},Z^{\mathrm{Y}}_j).$$

Since $x(L^{\mathrm{Y}})$ is a prefix of x of the correct form, by Claim 4.4.7 we have $\boldsymbol{\delta}_{\mathrm{Edit}}(x(L^{\mathrm{Y}}),L^{\mathrm{Y}}) = n\gamma_3 - \beta|x(L^{\mathrm{Y}})|$. Symmetrically, we obtain $\boldsymbol{\delta}_{\mathrm{Edit}}(x(R^{\mathrm{Y}}),R^{\mathrm{Y}}) = n\gamma_3 - \beta|x(R^{\mathrm{Y}})|$. Note that $|\mathrm{G}(x_i)Z^{\mathrm{X}}_i| = \gamma_4+\gamma_2$, so that $|x(L^{\mathrm{Y}})| + |x(R^{\mathrm{Y}})| = (n-m)(\gamma_4+\gamma_2)$. Moreover, as $Z^{\mathrm{X}}_i = Z^{\mathrm{Y}}_j = 0^{\gamma_2}$ we have $\boldsymbol{\delta}_{\mathrm{Edit}}(Z^{\mathrm{X}}_{\Delta+j},Z^{\mathrm{Y}}_j) = 0$. Finally, by matching all guarding zeroes and ones of $\mathrm{G}(x_{\Delta+j}) = (1^{\gamma_1}0^{\gamma_1})^\rho x_{\Delta+j}(0^{\gamma_1}1^{\gamma_1})^\rho$ and $\mathrm{G}(y_j) = (1^{\gamma_1}0^{\gamma_1})^\rho y_j(0^{\gamma_1}1^{\gamma_1})^\rho$ we conclude $\boldsymbol{\delta}_{\mathrm{Edit}}(\mathrm{G}(x_{\Delta+j}),\mathrm{G}(y_j)) \le \boldsymbol{\delta}_{\mathrm{Edit}}(x_{\Delta+j},y_j)$. This yields

$$\boldsymbol{\delta}_{\mathrm{Edit}}(x,y) \le 2n\gamma_3 - \beta(n-m)(\gamma_4+\gamma_2) + \sum_{j=1}^{m} \boldsymbol{\delta}_{\mathrm{Edit}}(x_{\Delta+j},y_j) = C + \sum_{(i,j)\in\Lambda} \boldsymbol{\delta}_{\mathrm{Edit}}(x_i,y_j).$$

As $\Lambda \in \mathcal{S}_{n,m}$ was arbitrary, the desired inequality follows.

It remains to prove the lower bound of (4.5), i.e., $\boldsymbol{\delta}_{\mathrm{Edit}}(x,y) \ge C + \min_{\Lambda\in\boldsymbol{\Lambda}_{n,m}} \mathbf{c}(\Lambda)$. As in Fact 3.5.1, let $x(L^{\mathrm{Y}})$, $x(\mathrm{G}(y_j))$ for $j\in[m]$, $x(Z^{\mathrm{Y}}_j)$ for $j\in[m-1]$, $x(R^{\mathrm{Y}})$ be an ordered partition of x such that

$$\boldsymbol{\delta}_{\mathrm{Edit}}(x,y) = \boldsymbol{\delta}_{\mathrm{Edit}}(x(L^{\mathrm{Y}}),L^{\mathrm{Y}}) + \boldsymbol{\delta}_{\mathrm{Edit}}(x(R^{\mathrm{Y}}),R^{\mathrm{Y}}) + \sum_{j=1}^{m} \boldsymbol{\delta}_{\mathrm{Edit}}(x(\mathrm{G}(y_j)),\mathrm{G}(y_j)) + \sum_{j=1}^{m-1} \boldsymbol{\delta}_{\mathrm{Edit}}(x(Z^{\mathrm{Y}}_j),Z^{\mathrm{Y}}_j).$$

We define an alignment Λ as follows. If there is some i such that x_i is contained in $x(\mathrm{G}(y_j))$, then align j with any such i. Otherwise leave j unaligned.

Claim 4.4.8. *We have*

$$\boldsymbol{\delta}_{\mathrm{Edit}}(x(\mathrm{G}(y_j)),\mathrm{G}(y_j)) \ge \beta(\gamma_4 - |x(\mathrm{G}(y_j))|) + \begin{cases} \boldsymbol{\delta}_{\mathrm{Edit}}(x_i,y_j) & \text{if } j \text{ is aligned to } i,\\ \max_{i,j'} \boldsymbol{\delta}_{\mathrm{Edit}}(x_i,y_{j'}) & \text{if } j \text{ is unaligned.} \end{cases}$$

Proof. If $|x(\mathrm{G}(y_j))| \ge \gamma_2$, it holds that $|x(\mathrm{G}(y_j))| \ge \gamma_2 \ge \gamma_4 + 2(\ell_{\mathrm{X}}+\ell_{\mathrm{Y}}) \ge \gamma_4 + 2\max_{i,j'} \boldsymbol{\delta}_{\mathrm{Edit}}(x_i,y_{j'})$ and by $\beta > 1/2$ the right hand side of the claim is at most 0, hence the claim holds trivially. Otherwise $x(\mathrm{G}(y_j))$ is shorter than any $Z^{\mathrm{X}}_i = 0^{\gamma_2}$, implying that $x(\mathrm{G}(y_j))$ is a substring of $0^{\gamma_2}\mathrm{G}(x_i)0^{\gamma_2}$ for some $i\in[n]$.

We write $\mathrm{G}(y_j)$ as $z_{-2\rho}\, z_{-2\rho+1} \dots z_{2\rho-1}\, z_{2\rho}$, where $z_{-2k} = z_{2k} = 1^{\gamma_1}$, $z_{-2k+1} = z_{2k-1} = 0^{\gamma_1}$, and $z_0 = y_j$ (for all $1 \le k \le \rho$). As in Fact 3.5.1, we split up $x(\mathrm{G}(y_j))$ into $x(z_k)$, $-2\rho \le k \le 2\rho$, such that $\boldsymbol{\delta}_{\mathrm{Edit}}(x(\mathrm{G}(y_j)),\mathrm{G}(y_j)) = \sum_{k=-2\rho}^{2\rho} \boldsymbol{\delta}_{\mathrm{Edit}}(x(z_k),z_k)$.

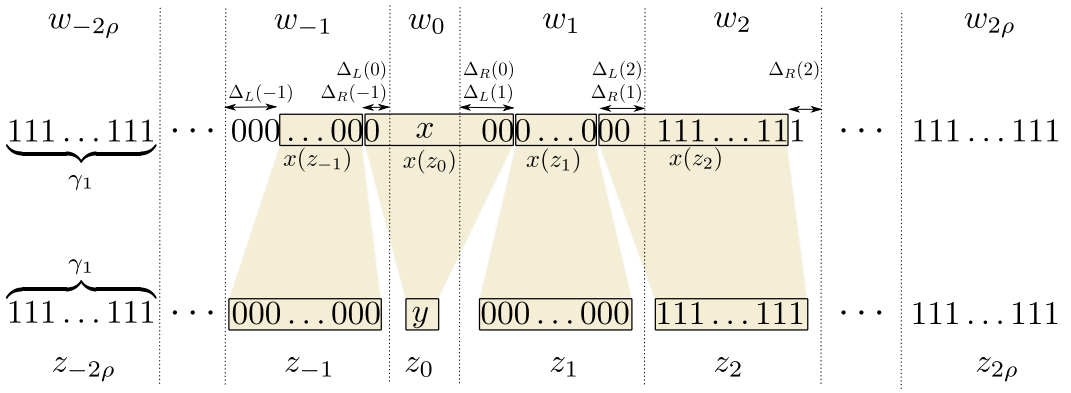


FIGURE 4.8: Illustration for the proof of Claim 4.4.8.

Similarly, we write $\mathrm{G}(x_i)$ as $w_{-2\rho}\, w_{-2\rho+1} \ldots w_{2\rho-1}\, w_{2\rho}$. We denote the distance of the start of $x(z_k)$ to the start of w_k by $\Delta_L(k)$, i.e., if $x(z_k) = x[a..b]$ and $w_k = x[c..d]$ we set $\Delta_L(k) := |a - c|$. Similarly, we set $\Delta_R(k) := |b - d|$. For an illustration, see Figure 4.8. Note that $\Delta_R(k) = \Delta_L(k+1)$ holds for any k.

First assume $(*)$: for some $k \neq 0$ the string $x(z_k)$ is longer than $\frac{5}{4}\gamma_1$ or $x(z_k)$ has less than $\frac{3}{4}\gamma_1$ common symbols with z_k. Then clearly $\boldsymbol{\delta}_{\mathrm{Edit}}(x(\mathrm{G}(y_j)), \mathrm{G}(y_j)) \geq \boldsymbol{\delta}_{\mathrm{Edit}}(x(z_k), z_k) \geq \frac{1}{4}\gamma_1$. By Fact 3.5.2(ii), we also have $\boldsymbol{\delta}_{\mathrm{Edit}}(x(\mathrm{G}(y_j)), \mathrm{G}(y_j)) \geq |\mathrm{G}(y_i)| - |x(\mathrm{G}(y_j))| = \gamma_4 - |x(\mathrm{G}(y_j))|$. As a linear combination of these two lower bounds, we obtain $\boldsymbol{\delta}_{\mathrm{Edit}}(x(\mathrm{G}(y_j)), \mathrm{G}(y_j)) \geq \beta(\gamma_4 - |x(\mathrm{G}(y_j))|) + (1-\beta)\frac{1}{4}\gamma_1$. Since $(1-\beta)\frac{1}{4}\gamma_1 = \frac{c_{\mathrm{subst}}}{20}\gamma_1 \geq \ell_{\mathrm{X}} + \ell_{\mathrm{Y}} \geq \max_{i,j'} \boldsymbol{\delta}_{\mathrm{Edit}}(x_i, y_{j'})$, we have proven the statement in this case.

If $(*)$ does not hold, then we have $\Delta_L(k), \Delta_R(k) \leq \frac{1}{2}\gamma_1$ for any $|k| > 1$: It suffices to show the claim for any even $k \neq 0$, since $\Delta_R(k) = \Delta_L(k+1)$. For any even $k \neq 0$, the string $x(z_k)$ has to contain the majority of a block w_ℓ with even $\ell \neq 0$. Since the numbers of blocks are identical in $\mathrm{G}(y_j)$ and $\mathrm{G}(x_i)$, $x(z_k)$ has to contain the majority of w_k for any even $k \neq 0$. Specifically, $x(z_k)$ contains at least $\frac{3}{4}\gamma_1$ symbols of w_k and has length at most $\frac{5}{4}\gamma_1$, implying the desired inequalities for $\Delta_L(k), \Delta_R(k)$. Note that in this case i and j are aligned.

Note that for even $k \neq 0$ we obtain $x(z_k)$ from $w_k = z_k = 1^{\gamma_1}$ by either deleting a prefix of $\Delta_L(k)$ ones or prepending $\Delta_L(k)$ zeroes, and by either deleting a suffix of $\Delta_R(k)$ ones or by appending $\Delta_R(k)$ zeroes. Hence, Fact 4.4.6 shows that

$$\boldsymbol{\delta}_{\mathrm{Edit}}(x(z_k), z_k) \geq |\Delta_L(k) - \Delta_R(k)| + c_{\mathrm{subst}} \cdot \min\{\Delta_L(k), \Delta_R(k)\}. \tag{4.6}$$

The same argument works for any k with $|k| > 1$. For $k \in \{-1, 1\}$ the argument does not work, since $z_{-1} = z_1 = 0^{\gamma_1}$ is not sorrounded by blocks of 1^{γ_1}. However, for $k \in \{-1, 1\}$ we have the weaker $\boldsymbol{\delta}_{\mathrm{Edit}}(x(z_k), z_k) \geq |\Delta_L(k) - \Delta_R(k)|$ by Fact 3.5.2(ii). Moreover, by Fact 3.5.2(iii) we have

$$\boldsymbol{\delta}_{\mathrm{Edit}}(x(z_0), z_0) \geq \boldsymbol{\delta}_{\mathrm{Edit}}(x_i, y_j) - \Delta_L(0) - \Delta_R(0). \tag{4.7}$$

Combining these inequalities yields $\boldsymbol{\delta}_{\mathrm{Edit}}(x(\mathrm{G}(y_j)), \mathrm{G}(y_j)) \geq \boldsymbol{\delta}_{\mathrm{Edit}}(x_i, y_j) + \Delta_L(-2\rho) + \Delta_R(2\rho)$ as we show in the following. This implies the desired statement, since $\Delta_L(-2\rho) + \Delta_R(2\rho) \geq \big||\mathrm{G}(y_j)| - |x(\mathrm{G}(y_j))|\big| \geq \gamma_4 - |x(\mathrm{G}(y_j))| \geq \beta(\gamma_4 - |x(\mathrm{G}(y_j))|)$. To show the claim, we set $s_L := \min\{\Delta_L(k) \mid -2\rho \leq k \leq 0\}$ and $s_R := \min\{\Delta_R(k) \mid 0 \leq k \leq 2\rho\}$. Note that $\Delta_L(k)$ has a total variation of at least $\Delta_L(-2\rho) - s_L + \Delta_L(0) - s_L$ over $k = -2\rho, \ldots, 0$, since it starts in $\Delta_L(-2\rho)$, changes to s_L, and then changes to $\Delta_L(0)$. Thus, summing $|\Delta_L(k) - \Delta_R(k)| = |\Delta_L(k) - \Delta_L(k+1)|$ over all $-2\rho \leq k \leq -1$

yields at least $\Delta_L(-2\rho) - s_L + \Delta_L(0) - s_L$. Moreover, for every $-2\rho \le k < -1$ inequality (4.6) applies and the summand $c_{\text{subst}} \cdot \min\{\Delta_L(k), \Delta_R(k)\}$ is at least $c_{\text{subst}} \cdot s_L$. As the number of such k's is $2\rho - 1 \ge 2/c_{\text{subst}}$, the total contribution of the summand $c_{\text{subst}} \cdot \min\{\Delta_L(k), \Delta_R(k)\}$ over all $k < 0$ is at least $2s_L$. Thus, we have

$$\begin{aligned}\sum_{k=-2\rho}^{-1} \boldsymbol{\delta}_{\text{Edit}}(x(z_k), z_k) &\ge \sum_{k=-2\rho}^{-1} |\Delta_L(k) - \Delta_R(k)| + \sum_{k=-2\rho}^{-2} c_{\text{subst}} \cdot \min\{\Delta_L(k), \Delta_R(k)\} \\ &\ge \big(\Delta_L(-2\rho) - s_L + \Delta_L(0) - s_L\big) + \big(2s_L\big) \ge \Delta_L(-2\rho) + \Delta_L(0).\end{aligned}$$

Using a symmetric statement for the sum over all $k > 0$ as well as equation (4.7), we obtain the desired inequality $\boldsymbol{\delta}_{\text{Edit}}(x(\mathrm{G}(y_j)), \mathrm{G}(y_j)) = \sum_{k=-2\rho}^{2\rho} \boldsymbol{\delta}_{\text{Edit}}(x(z_k), z_k) \ge \boldsymbol{\delta}_{\text{Edit}}(x_i, y_j) + \Delta_L(-2\rho) + \Delta_R(2\rho)$. □

Since $L^{\text{Y}} = 0^{n\gamma_3}$ and $x(L^{\text{Y}})$ is a prefix of x, by Claim 4.4.7 we have $\boldsymbol{\delta}_{\text{Edit}}(x(L^{\text{Y}}), L^{\text{Y}}) \ge n\gamma_3 - \beta|x(L^{\text{Y}})|$, and symmetrically we get $\boldsymbol{\delta}_{\text{Edit}}(x(R^{\text{Y}}), R^{\text{Y}}) \ge n\gamma_3 - \beta|x(L^{\text{Y}})|$. By Fact 3.5.2(ii), we have $\boldsymbol{\delta}_{\text{Edit}}(x(Z_j^{\text{Y}}), Z_j^{\text{Y}}) \ge \big||Z_j^{\text{Y}}| - |x(Z_j^{\text{Y}})|\big| \ge \beta(\gamma_2 - |x(Z_j^{\text{Y}})|)$. Putting all of this together, we obtain

$$\begin{aligned}\boldsymbol{\delta}_{\text{Edit}}(x, y) &\ge 2n\gamma_3 + \mathbf{c}(\Lambda) \\ &\quad + \beta\Big[\sum_{j=1}^{m}(\gamma_4 - |x(\mathrm{G}(y_j))|) + \sum_{j=1}^{m-1}(\gamma_2 - |x(Z_j^{\text{Y}})|) - |x(L^{\text{Y}})| - |x(R^{\text{Y}})|\Big],\end{aligned}$$

where we used $\mathbf{c}(\Lambda) = \sum_{(i,j)\in\Lambda} \boldsymbol{\delta}_{\text{Edit}}(x_i, y_j) + (m - |\Lambda|)\max_{i,j} \boldsymbol{\delta}_{\text{Edit}}(x_i, y_j)$. Note that by definition of x and since the strings $x(\mathrm{G}(y_j)), x(Z_j^{\text{Y}}), x(L^{\text{Y}}), x(R^{\text{Y}})$ partition x we have

$$n\gamma_4 + (n-1)\gamma_2 = |x| = \sum_{j=1}^{m} |x(\mathrm{G}(y_j))| + \sum_{j=1}^{m-1} |x(Z_j^{\text{Y}})| + |x(L^{\text{Y}})| + |x(R^{\text{Y}})|.$$

Together, this yields the desired bound from below

$$\boldsymbol{\delta}_{\text{Edit}}(x, y) \ge 2n\gamma_3 - \beta(n - m)(\gamma_4 + \gamma_2) + \mathbf{c}(\Lambda).$$

This finishes the proof of the alignment gadget for Edit Distance. □

Chapter 5

Multivariate Fine-grained Complexity of LCS

In Chapter 4 (Section 4.3), we presented a quadratic SETH-based lower bound for LCS. While this gives a justification why no LCS algorithm is known that succeeds in breaking the quadratic-time barrier in the strong sense, a number of these algorithms performs significantly better when the problem is restricted to inputs with certain structural properties. This is most prominently witnessed by the UNIX `diff` utility, which quickly compares large, similar files (exploiting that the longest common subsequence in such instances differs from the input strings at only few positions). In fact, since Wagner and Fischer introduced LCS in 1977, algorithmic progress has effectively been limited to obtaining algorithms whose running time is small whenever certain natural parameters are small (see, e.g., [Apo86; AG87; Epp+92; Hir77; HS77; IR09; Mye86; NKY82; Wu+90]). A notable exception is the result by Masek and Paterson [MP80], who applied the Four Russians technique to save logarithmic factors in the worst case running time. Yet, the main focus of the LCS literature (at least between the late 1970s and early 1990s) appears to be finding fast algorithms in terms of natural input parameters. Recall that we refer to such algorithms, whose running times are stated in more input parameters than just the problem size n, as *multivariate algorithms*.

More specifically, the input parameters (for multivariate algorithms) that have been studied in the literature are, besides the input size $n := \max\{|x|, |y|\}$, the length $m := \min\{|x|, |y|\}$ of the shorter string, the size of the alphabet Σ that x and y are defined on, the length L of a longest common subsequence of x and y, the number $\Delta = n - L$ of deleted symbols in x, the number $\delta = m - L$ of deleted symbols in y, the number of *matching* pairs M, and the number of *dominant* pairs d (see Section 5.1 for definitions). Among the fastest currently known multivariate algorithms are an $\tilde{\mathcal{O}}(n + d)$-algorithm due to Apostolico [Apo86], an $\tilde{\mathcal{O}}(n + \delta L)$-algorithm due to Hirschberg [Hir77] and an $\tilde{\mathcal{O}}(n + \delta\Delta)$-algorithm due to Wu, Manbers, Myers, and Miller [Wu+90]. In Table 5.1, we provide a non-exhaustive survey[1] containing the asymptotically fastest multivariate LCS algorithms.

Even when considering the progress in terms of multivariate algorithms for LCS, algorithmic improvements by more than polylogarithmic factors have stalled since the early 1990s. The natural and intriguing question arises whether SETH-based lower bounds can also explain the lack of progress of multivariate algorithms for LCS. This chapter is devoted to answering this question in the affirmative: It turns out that in

[1] Note that in our discussion, we solely regard the time complexity of computing the length of an LCS and hence omit all results concerning space usage or finding an LCS. See, e.g., [PD94] and [BHR00] for these and other aspects of LCS as well as empirical comparisons.

Reference	Running Time
Wagner and Fischer [WF74]	$\mathcal{O}(mn)$
Hunt and Szymanski [HS77]	$\mathcal{O}((n+M)\log n)$
Hirschberg [Hir77]	$\mathcal{O}(n\log n + Ln)$
Hirschberg [Hir77]	$\mathcal{O}(n\log n + L\delta\log n)$
Masek and Paterson [MP80]	$\mathcal{O}(n + nm/\log^2 n)$ for $\lvert\Sigma\rvert = \mathcal{O}(1)$
	$\mathcal{O}\big(n + nm\cdot\big(\frac{\log\log n}{\log n}\big)^2\big)$ [4]
Nakatsu et al. [NKY82]	$\mathcal{O}(n\delta)$
Apostolico [Apo86]	$\mathcal{O}(n\log n + d\log(\frac{mn}{d}))$
Myers [Mye86]	$\mathcal{O}(n\log n + \Delta^2)$
Apostolico and Guerra [AG87]	$\mathcal{O}(n\log n + Lm\min\{\log m, \log(n/m))\})$
Wu et al. [Wu+90]	$\mathcal{O}(n\log n + \delta\Delta)$ [5]
Eppstein et al. [Epp+92]	$\mathcal{O}(n\log n + d\log\log\min\{d, nm/d\})$
Iliopoulos and Rahman [IR09]	$\mathcal{O}(n + M\log\log n)$

TABLE 5.1: Short survey of LCS algorithms. See Section 5.1 for definitions of the parameters. When stating the running times, every factor possibly attaining non-positive values (such as $\delta, \log(n/m)$, etc.) is to be read as $\max\{\cdot, 1\}$. For simplicity, $\log(|\Sigma|)$-factors have been bounded from above by $\log n$ (see [PD94] for details on the case of constant alphabet size).

order to improve upon the currently best known running time[2] of $\tilde{\mathcal{O}}(n + \min\{d, \delta m, \delta\Delta\})$ by a polynomial factor, one has to either refute SETH or invent novel input parameters that can be algorithmically exploited.[3]

Further Related Work. Note that in contrast to the field of fixed-parameter tractability (studying the parameterized complexity of NP-hard problems) in our case, both the total running time and the dependence on any of the studied parameters is polynomial. However, since it separates the dependence of the running time on the problem size from the dependence on the input parameters, the results in this chapter can be seen as part of the field "FPT in P" that recently received growing attention (see the paper by Abboud, Vassilevska Williams and Wang [AVWW16] who introduced the study of the conditional complexity of this field, and, e.g., [GMN15] for a recent algorithmic case study).

On a conceptual level, our approach is reminiscent of a systematic study of the parameterized complexity of the subgraph isomorphism problem [MP14], which presents for any subset of 10 relevant input parameters either a fixed-parameter tractable algorithm or a hardness proof. To the best of our knowledge, such a systematic study has not yet been conducted for any polynomial-time problem. The lack thereof is explained by the relatively short history of conditional lower bounds for polynomial-time problems: As discussed in Chapter 2, except for the early success of 3SUM-hardness in computational geometry, only in recent years meaningful polynomial conditional lower bounds have been

[2] Note that at first sight it might seem as if Hirschberg's $\tilde{\mathcal{O}}(n + \delta L)$-time algorithm could be faster than this bound, however, for $L \geq m/2$ we have $\delta L = \Theta(\delta m)$, which appears in our lower bound, and for $L \leq m/2$ we have $\delta = m - L = \Theta(m)$ and thus $\delta L = \Theta(Lm)$ which is $\Omega(d)$ by Lemma 5.4.3, given below.

[3] This holds for ternary and larger alphabets. For the special case of binary alphabets, we provide a missing algorithm with running time $\mathcal{O}(n + \delta M/n)$ and obtain a SETH-based lower bound of $(n + \min\{d, \delta M/n, \delta\Delta\})^{1-o(1)}$ that is tight up to lower order factors.

[4] See [BFC08] for how to extend the Masek-Paterson algorithm to non-constant alphabets.

[5] Wu et al. state their running time as $\mathcal{O}(n\delta)$ in the worst case and $\mathcal{O}(n + \delta\Delta)$ in expectation for random strings. However, Myers worst-case variation trick [Mye86, Section 4c] applies and yields the claimed time bound $\mathcal{O}(n\log n + \delta\Delta)$. The additional $\mathcal{O}(n\log n)$ comes from building a suffix tree.

obtained (see, apart from the references to SETH-based lower bounds in Chapter 2, also work based on further conjectures [ABVW15a; KPP16; WW10; Pat10]). Increasingly, additional input parameter than the problem size are studied as well [Bri14; KPP16; Abb+16b], yet hitherto, to the best of our knowledge, never a combination of parameters with a complex set of relations.

We remark that using very recent work, we could even perform our multivariate study based on a weaker assumption than SETH: a remarkable result of Abboud et al. [Abb+16a] bases the quadratic hardness of the alignment gadget framework and hence LCS on a variant of SETH on branching programs[6]. Since the basis for our constructions, so-called normalized vector gadgets, are also constructed in [Abb+16a], in this chapter we could also reduce from satisfiability of branching programs. To avoid an overly technical presentation, however, we chose to start from the classical SETH/OVH.

Organization of This Chapter. This chapter is organized as follows: In Section 5.1, we define the input parameters that the LCS literature has introduced and thus are fundamental to our multivariate study. In Section 5.2, we then formally develop our paradigm of multivariate fine-grained complexity and state the results. We provide an overview of our proofs in Section 5.3, which explains the different tasks of our approach: uncovering all parameter relations (Section 5.4), embedding orthogonal vectors instances into strings with bounded input parameters (Section 5.6), padding the input parameters for large alphabets (Section 5.7) and finally giving the specialized constructions for constant alphabet sizes (Section 5.8). Technical tools and general constructions needed for our proofs are prepared in Section 5.5.

5.1 Parameter Definitions

We survey parameters that have been used in the analysis of the LCS problem (see also [PD94; BHR00]). Let x, y be any strings over alphabet Σ. We usually consider the induced alphabet $\Sigma = \{x[i] \mid 1 \leq i \leq |x|\} \cup \{y[j] \mid 1 \leq j \leq |y|\}$ and consider its size $|\Sigma|$ as a parameter. Since any symbol not contained in x or in y cannot be contained in an LCS, we can ensure the following assumption using a (near-)linear-time preprocessing.

Assumption 5.1.1. *Every symbol $\sigma \in \Sigma$ occurs at least once in x and in y, i.e., $\#_\sigma(x), \#_\sigma(y) \geq 1$.*

By possibly swapping x and y, we can assume that x is the longer of the two strings, so that $n = n(x, y) := |x|$ is the input size. Then $m = m(x, y) := |y|$ is the length of the shorter of the two strings. Another natural parameter is the solution size, i.e., the length $L(x, y) = |\mathrm{LCS}(x, y)|$ of any longest common subsequence $\mathrm{LCS}(x, y)$ of x and y.

Beyond these standard parameters (applicable for basically any optimization problem), popular structural parameters measure the *sparsity* and *similarity* of the strings. These notions are more specific to LCS and are especially relevant in practical applications such as, e.g., the `diff` file comparison utility, where symbols in x and y correspond to lines in the input files and the length of their LCS measures the similarity of x and y.

Notions of Sparsity. Based on the observation that the dynamic programming table typically stores a large amount of redundant information (suggested, e.g., by the fact that an LCS itself can be reconstructed examining only $O(n)$ entries), algorithms have

[6] For a short discussion, see Section 3.3.

	d	a	b	c	c	b	d
d	**1**	1	1	1	1	1	1
c	1	1	1	**2**	2	2	2
b	1	1	**2**	2	2	**3**	3
a	1	**2**	2	2	2	3	3
d	1	2	2	2	2	3	**4**
c	1	2	2	**3**	3	3	4

FIGURE 5.1: Illustration of the L-table, matching pairs and dominant pairs. Entries marked in orange and bold letters correspond to dominant pairs (which by definition are also matching pairs), while entries marked in blue are matching pairs only.

been studied that consider only the most relevant entries in the table. The simplest measure of such entries is the number of *matching pairs* $M(x,y) := \#\{(i,j) \mid x[i] = y[j]\}$. Especially for inputs with a large alphabet (e.g., files for which almost all lines occur only once), this parameter may significantly restrict the number of candidate pairs considered by LCS algorithms.

One can refine this notion to obtain the *dominant pairs*: A pair (i,j) *dominates* a pair (i',j') if we have $i \le i'$ and $j \le j'$. A *k-dominant pair* is a pair (i,j) such that $L(x[1..i], y[1..j]) = k$ and no other pair (i',j') with $L(x[1..i'], y[1..j']) = k$ dominates (i,j). By defining $L[i,j] := L(x[1..i], y[1..j])$ and using the well-known recursion $L[i,j] = \max\{L[i-1,j], L[i,j-1], L[i-1,j-1] + \mathbf{1}_{x[i]=y[j]}\}$, we observe that (i,j) is a k-dominant pair if and only if $L[i,j] = k$ and $L[i-1,j] = L[i,j-1] = k-1$. Denoting the set of all k-dominant pairs by D_k, the number of *dominant pairs* of x, y is $d(x,y) := |\bigcup_{k\ge 1} D_k|$. Figure 5.1 illustrates matching and dominant pairs. Note that the set of dominant pairs sparsely encodes the complete L-table: It contains, for every k, the Pareto set of table entries $L[i,j] \ge k$.

Notions of Similarity. To obtain an LCS, we have to delete $\Delta = \Delta(x,y) := n - L$ symbols from x or $\delta = \delta(x,y) := m - L$ symbols from y. Hence for very similar strings, which is the typical kind of input for file comparisons, we expect δ and Δ to be small – on practical instances, exploiting these similarity notions seems to typically outperform algorithms based on sparsity measures (see [MM85] for a classical comparison of Myer's algorithm to an algorithm based on the number of matching pairs M [HS77; HM75]).

To the best of our knowledge, these are all parameters which have been exploited to obtain faster multivariate algorithms for LCS. We summarize the parameter definitions as follows,

$$
\begin{aligned}
n(x,y) &:= |x|, \quad m(x,y) := |y|,\\
L(x,y) &:= |\mathrm{LCS}(x,y)|,\\
\delta(x,y) &:= |y| - L(x,y), \quad \Delta(x,y) := |x| - L(x,y),\\
\Sigma(x,y) &:= \{x[i] \mid 1 \le i \le |x|\} \cup \{y[j] \mid 1 \le j \le |y|\}.\\
M(x,y) &:= \#\{(i,j) \mid x[i] = y[j]\},\\
d(x,y) &:= \#\{(i,j) \mid L[i,j] > L[i-1,j] \text{ and } L[i,j] > L[i,j-1]\},\\
&\qquad \text{where } L[i,j] := L(x[1..i], y[1..j]).
\end{aligned}
$$

We remark that for some strings x, y considered in this chapter the assumption $|x| \geq |y|$ may be violated; in this case we use the definitions given above (and thus we may have $n(x,y) < m(x,y)$ and $\Delta(x,y) < \delta(x,y)$). Since L, M, d, and Σ are symmetric in the sense that $L(x,y) = L(y,x)$, these parameters are independent of the assumption $|x| \geq |y|$.

5.2 Formal Statement of Results

Recall that n is the input size and $\mathcal{P} := \{m, L, \delta, \Delta, |\Sigma|, M, d\}$ is the set of parameters that were previously studied in the literature. We let $\mathcal{P}^* := \mathcal{P} \cup \{n\}$. A parameter setting fixes a polynomial relation between any parameter and n. To formalize this, we call a vector $\boldsymbol{\alpha} = (a_p)_{p \in \mathcal{P}}$ with $a_p \in \mathbb{R}_{\geq 0}$ a *parameter setting*, and an LCS instance x, y *satisfies* the parameter setting $\boldsymbol{\alpha}$ if each parameter p attains a value $p(x,y) = \Theta(n^{\alpha_p})$. This yields a subproblem of LCS consisting of all instances that satisfy the parameter setting. We sometimes use the notation $\alpha_n = 1$.

For our running time bounds, for each parameter $p \in \mathcal{P}$ except for $|\Sigma|$ we can assume $\alpha_p > 0$, since otherwise one of the known algorithms runs in time $\tilde{\mathcal{O}}(n)$ and there is nothing to show. Similarly, for $\alpha_d \leq 1$ there is an $\tilde{\mathcal{O}}(n)$ algorithm and there is nothing to show. For $|\Sigma|$, however, the case $\alpha_\Sigma = 0$, i.e., $|\Sigma| = \Theta(1)$, is an important special case. We study this case more closely by also allowing to fix $|\Sigma|$ to any specific constant greater than 1.

Definition 5.2.1 (Parameter Setting)**.** *Fix $\gamma \geq 1$. Let $\boldsymbol{\alpha} = (a_p)_{p \in \mathcal{P}}$ be any vector of non-negative reals. We define* $\mathrm{LCS}^\gamma(\boldsymbol{\alpha})$ *as the problem of computing the length of an LCS of two given strings x, y satisfying $n^{\alpha_p}/\gamma \leq p(x,y) \leq n^{\alpha_p} \cdot \gamma$ for every parameter $p \in \mathcal{P}$, where $n = |x|$, and $|x| \geq |y|$. We call $\boldsymbol{\alpha}$ and* $\mathrm{LCS}^\gamma(\boldsymbol{\alpha})$ parameter settings*. In some statements we simply write* $\mathrm{LCS}(\boldsymbol{\alpha})$ *to abbreviate that there exists a $\gamma \geq 1$ such that the statement holds for* $\mathrm{LCS}^\gamma(\boldsymbol{\alpha})$*.*

For any fixed alphabet Σ, constant $\gamma \geq |\Sigma|$, and parameter setting $\boldsymbol{\alpha}$ with $\alpha_\Sigma = 0$, we also define the problem $\mathrm{LCS}^\gamma(\boldsymbol{\alpha}, \Sigma)$*, where additionally the alphabet of x, y is fixed to be Σ. We again call $(\boldsymbol{\alpha}, \Sigma)$ and* $\mathrm{LCS}^\gamma(\boldsymbol{\alpha}, \Sigma)$ *parameter settings.*

We call a parameter setting $\boldsymbol{\alpha}$ or $(\boldsymbol{\alpha}, \Sigma)$ trivial *if for all $\gamma \geq 1$ the problem* $\mathrm{LCS}^\gamma(\boldsymbol{\alpha})$ *or* $\mathrm{LCS}^\gamma(\boldsymbol{\alpha}, \Sigma)$*, respectively, has only finitely many instances.*

We are ready to formally state our lower bound results, whose proof will be outlined in Section 5.3.

Theorem 5.2.2. *For any non-trivial parameter setting $\boldsymbol{\alpha}$ there is a constant $\gamma \geq 1$ such that any algorithm for* $\mathrm{LCS}^\gamma(\boldsymbol{\alpha})$ *takes time* $\min\{d, \delta\Delta, \delta m\}^{1-o(1)}$*, unless* OVH *fails.*

In the case of constant alphabet size, the (conditional) complexity differs between $|\Sigma| = 2$ and $|\Sigma| \geq 3$. Note that $|\Sigma| = 1$ makes LCS trivial.

Theorem 5.2.3. *For any non-trivial parameter setting $(\boldsymbol{\alpha}, \Sigma)$, there is a constant $\gamma \geq |\Sigma|$ such that unless* OVH *fails any algorithm for* $\mathrm{LCS}^\gamma(\boldsymbol{\alpha}, \Sigma)$ *takes time*

- $\min\{d, \delta\Delta, \delta m\}^{1-o(1)}$ *if* $|\Sigma| \geq 3$*,*
- $\min\{d, \delta\Delta, \delta M/n\}^{1-o(1)}$ *if* $|\Sigma| = 2$*.*

Note that in Section 6.2, we prove the following algorithmic result, handling the case $|\Sigma| = 2$ faster whenever M and δ are sufficiently small. This yields matching upper and lower bounds also for $|\Sigma| = 2$.

Theorem 5.2.4. *For $|\Sigma| = 2$, LCS can be solved in time $\mathcal{O}(n + \delta M/n)$.*

5.3 Hardness Proof Overview

In order to get a handle on non-trivial parameter settings, we first need necessary conditions for being non-trivial. We prove several inequalities on the values α_p, $p \in \mathcal{P}^*$, see Table 5.2 on page 54. Note that for small alphabet sizes $|\Sigma| \in \{2, 3\}$, additional parameter relations hold. We will later see in Corollary 5.3.7 that for parameter settings $\boldsymbol{\alpha}$ these necessary conditions are also sufficient, and thus our list of relations is complete.

Lemma 5.3.1 (Parameter Relations, Section 5.4)**.** *For any strings x and y, the parameter values $\mathcal{P}^*$ satisfy the relations in Table 5.3. Thus, any non-trivial parameter setting $\boldsymbol{\alpha}$ or $(\boldsymbol{\alpha}, \Sigma)$ satisfies Table 5.2.*

Proof Sketch. The full proof is deferred to Section 5.4. Some parameter relations are trivial, like $L \leq m \leq n$. By a simple preprocessing, we can ensure that every symbol in Σ appears in x and y, implying parameter relations like $m \leq |\Sigma|$. Other parameter relations need a non-trivial proof, like $M \geq md/(80L)$ if $|\Sigma| = 3$. From a relation like $L \leq m$ we infer that if $\alpha_L > \alpha_m$ then for sufficiently large n no strings x, y have $L(x, y) = \Theta(n^{\alpha_L})$ and $m(x, y) = \Theta(n^{\alpha_m})$, and thus $\mathrm{LCS}(\boldsymbol{\alpha})$ is finite. Hence, any non-trivial parameter setting satisfies Table 5.2. □

It might be tempting to assume that the optimal running time for parameter settings is monotone in the input size n and the parameters $\mathcal{P}$ (say up to constant factors, as long as all considered parameters settings are non-trivial). However, since our parameters have complex dependencies (see Table 5.3), it is far from obvious whether this intuition is correct. In fact, the intuition *fails* for $|\Sigma| = 2$, where the running time $\mathcal{O}(n + \delta M/n)$ of our new algorithm is not monotone, and thus also the tight time bound $\tilde{\mathcal{O}}(n + \min\{d, \delta\Delta, \delta M/n\})$ is not monotone.

Thus, let us focus on large alphabets, i.e., parameter settings $\boldsymbol{\alpha}$, where the alphabet size is at least a large constant. A non-negligible fraction of this chapter is devoted to proving monotonicity in this case. To formalize this task, we consider the problem $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$ where the slice of input size n consists of all instances x, y of $\mathrm{LCS}(\boldsymbol{\alpha})$ with input size and all parameters *at most* as in $\mathrm{LCS}(\boldsymbol{\alpha})$.

Definition 5.3.2 (Downward Closure of Parameter Setting)**.** *Fix $\gamma \geq 1$ and let $\boldsymbol{\alpha}$ be a parameter setting. We define the* downward closure $\mathrm{LCS}^{\gamma}_{\leq}(\boldsymbol{\alpha})$ *as follows. An instance of this problem is a triple (n, x, y), where $p(x, y) \leq \gamma \cdot n^{\alpha_p}$ for any $p \in \mathcal{P}^* = \{n, m, L, \delta, \Delta, |\Sigma|, M, d\}$, and the task is to compute the length of an LCS of x and y. In some statements, we simply write $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$ to abbreviate that there exists a $\gamma \geq 1$ such that the statement holds for $\mathrm{LCS}^{\gamma}_{\leq}(\boldsymbol{\alpha})$.*

Similarly, for any fixed alphabet Σ, we define the downward closure $\mathrm{LCS}^{\gamma}_{\leq}(\boldsymbol{\alpha}, \Sigma)$ as the family of instances (n, x, y) where x, y are strings over alphabet Σ and $p(x, y) \leq \gamma \cdot n^{\alpha_p}$ for any $p \in \mathcal{P}^*$. We use the shorthand $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha}, \Sigma)$ analogously to $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$.

We prove a conditional lower bound directly for $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$. Then in Lemma 5.3.4 below, we show that monotonicity implies the same lower bound for $\mathrm{LCS}(\boldsymbol{\alpha})$.

Theorem 5.3.3 (Hardness for Large Alphabet, Section 5.6)**.** *For any non-trivial parameter setting $\boldsymbol{\alpha}$, there is a constant $\gamma \geq 1$ such that any algorithm for $\mathrm{LCS}^{\gamma}_{\leq}(\boldsymbol{\alpha})$ takes time $\min\{d, \delta m, \delta\Delta\}^{1-o(1)}$, unless* OVH *fails.*

Proof Sketch. The full proof is deferred to Section 5.6. We provide different reductions for the cases $\alpha_\delta = \alpha_m$ and $\alpha_L = \alpha_m$. In the former case, our reduction from OV to

LCS, given in Chapter 4, yields the claim for $|\Sigma| = \mathcal{O}(1)$. For larger alphabet, we use the construction for constant alphabet multiple times and concatenate the results (with reversed order in one string).

In the case $\alpha_L = \alpha_m$, we want to construct strings where the longest common subsequence may be very large, but the numbers of deleted symbols δ, Δ could be small. Our construction of Chapter 4 fails, and we present a new way of combining "normalized vector gadgets". We think that this construction is our main contribution to specific lower bound proof techniques and might find more applications. In this construction, picking any two vector gadgets in x and one in y yields an LCS without any deletions in the vector gadget of y, thus not increasing δ, which is important for keeping δ small. Picking one vector gadget in both x and y yields an LCS whose length depends on the orthogonality of the vectors. We then need to ensure that such "2vs1" and "1vs1" matchings always generate an LCS and at least one "1vs1" matchings appears in any LCS. □

Monotonicity of the optimal running time for parameter settings is equivalent to the following statement (since for any $\boldsymbol{\alpha}' \leq \boldsymbol{\alpha}$ the parameter setting $\mathrm{LCS}(\boldsymbol{\alpha}')$ "is contained in" $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$). Together with Theorem 5.3.3 this proves Theorem 5.2.2.

Lemma 5.3.4 (Monotonicity)**.** *If $\boldsymbol{\alpha}$ is non-trivial and $\mathrm{LCS}(\boldsymbol{\alpha})$ has an $\mathcal{O}(n^\beta)$ algorithm for some $\beta \geq 1$, then $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$ also has an $\mathcal{O}(n^\beta)$ algorithm.*

For proving the above lemma, we need the useful property that all studied parameters sum up if we concatenate strings over disjoint alphabets.

Lemma 5.3.5 (Disjoint Alphabets)**.** *Let $\Sigma_1, \ldots, \Sigma_k$ be disjoint alphabets and let x_i, y_i be strings over alphabet Σ_i. Consider $x := x_1 \ldots x_k$ and $y := y_1 \ldots y_k$. For any parameter $p \in \mathcal{P}^*$, we have $p(x, y) = \sum_{i=1}^{k} p(x_i, y_i)$.*

Proof. The statement is trivial for the string lengths n, m, alphabet size $|\Sigma|$, and number of matching pairs M. For the LCS length L, we observe that any common subsequence z can be decomposed into $z_1 \ldots z_k$ with z_i using only symbols from Σ_i, so that $|z_i| \leq L(x_i, y_i)$ and thus $L(x, y) \leq \sum_{i=1}^{k} L(x_i, y_i)$. Concatenating longest common subsequences of x_i, y_i, we obtain equality. Using $\delta = m - L$ and $\Delta = n - L$, the claim follows also for δ and Δ.

Since every dominant pair is also a matching pair, every dominant pair of x, y stems from prefixes $x_1 \ldots x_j x'$ and $y_1 \ldots y_j y'$, with x' being a prefix of x_{j+1} and y' being a prefix of y_{j+1} for some j. Since $L(x_1 \ldots x_j x', y_1 \ldots y_j y') = \sum_{i=1}^{j} L(x_i, y_i) + L(x', y')$, where the first summand does not depend on x', y', the dominant pairs of x, y of the form $x_1 \ldots x_j x', y_1 \ldots y_j y'$ are in one-to-one correspondence with the dominant pairs of x_{j+1}, y_{j+1}. This yields the claim for parameter d. □

The general idea to prove Lemma 5.3.4 is as follows. Given an instance (n, x, y) of $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$, take any fixed instance x', y' of size $\Theta(n)$ of $\mathrm{LCS}(\boldsymbol{\alpha})$, which exists since $\boldsymbol{\alpha}$ is non-trivial, and make sure that (x, y) and (x', y') use disjoint alphabets. Consider $x'' := x \circ x'$ and $y'' := y \circ y'$. Then the Disjoint Alphabets Lemma implies that (x'', y'') is an instance of $\mathrm{LCS}(\boldsymbol{\alpha})$. Hence, we can compute $L(x'', y'')$ in time $\mathcal{O}(n^\beta)$, which yields $L(x, y)$ via $L(x, y) = L(x'', y'') - L(x', y')$.

For this argument, we need to be able to compute, given any n, an instance (x', y') of $\mathrm{LCS}(\boldsymbol{\alpha})$ of size $\Theta(n)$, as well as the LCS length $L(x', y')$, in time $\mathcal{O}(n^\beta)$ (or, say, $\mathcal{O}(n)$). This is why we need the following constructive characterization of non-trivial parameter settings.

Parameter	Restriction
m	$0 \leq \alpha_m \leq 1$
L	$0 \leq \alpha_L \leq \alpha_m$
δ	$\begin{cases} 0 \leq \alpha_\delta \leq \alpha_m & \text{if } \alpha_L = \alpha_m \\ \alpha_\delta = \alpha_m & \text{otherwise} \end{cases}$
Δ	$\begin{cases} \alpha_\delta \leq \alpha_\Delta \leq 1 & \text{if } \alpha_L = \alpha_m = 1 \\ \alpha_\Delta = 1 & \text{otherwise} \end{cases}$
$\lvert\Sigma\rvert$	$0 \leq \alpha_\Sigma \leq \alpha_m$
d	$\max\{\alpha_L, \alpha_\Sigma\} \leq \alpha_d \leq \min\{2\alpha_L + \alpha_\Sigma, \alpha_L + \alpha_m, \alpha_L + \alpha_\Delta\}$
M	$\max\{1, \alpha_d, 2\alpha_L - \alpha_\Sigma\} \leq \alpha_M \leq \alpha_L + 1$ if $\lvert\Sigma\rvert = 2$: $\alpha_M \geq \max\{\alpha_L + \alpha_m, 1 + \alpha_d - \alpha_L\}$ if $\lvert\Sigma\rvert = 3$: $\alpha_M \geq \alpha_m + \alpha_d - \alpha_L$

Table 5.2: Non-trivial parameter settings.

Lemma 5.3.6 (Section 5.7)**.** *Let $\boldsymbol{\alpha}$ be a parameter setting satisfying Table 5.2. For every parameter $p \in \mathcal{P}^*$ and any $n \geq 1$, there is an instance (n, x_p, y_p) of $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$ such that $p(x_p, y_p) = \Theta(n^{\alpha_p})$, and given n we can compute $x_p = x_p(n)$, $y_p = y_p(n)$, and $L(x_p, y_p)$ in time $\mathcal{O}(n)$.*

Proof Sketch. We defer the full proof to Section 5.7 and here only sketch the idea for the parameter L. Let $n \geq 1$ and $L := n^{\alpha_L}$. We argue that if $\alpha_M \geq 2\alpha_L$ then $x := y := 1^L$ satisfy the claim for parameter L. The construction is more complicated if $\alpha_M < 2\alpha_L$, and for other parameters.

Note that we have $n(x, y) = m(x, y) = L(x, y) = L$. Since $\alpha_L \leq \alpha_m \leq 1$ by Table 5.2, we have $L = n^{\alpha_L} \leq n^{\alpha_m} \leq n$. Hence, the parameter $p(x, y)$ realized by x, y is bounded by the target value n^{α_p} for $p \in \{n, m, L\}$. For the other parameters we argue similarly: By the trivial lower bounds $\alpha_p \geq 0$, we have $\Delta(x, y) = \delta(x, y) = 0 \leq n^{\alpha_\delta} \leq n^{\alpha_\Delta}$ and $|\Sigma(x, y)| = 1 \leq n^{\alpha_\Sigma}$. We will later see that $d(x, y) = L \leq n \leq n^{\alpha_d}$ (see Lemma 5.5.1). From the assumption $\alpha_M \geq 2\alpha_L$ we obtain $M(x, y) = L^2 \leq n^{2\alpha_L} \leq n^{\alpha_M}$. Thus, we have shown that (n, x, y) is an instance of $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$. Moreover, as desired $L(x, y) = \Theta(n^{\alpha_L})$, and x, y, and $L(x, y)$ can be computed in time $\mathcal{O}(n)$. □

In particular, the above lemma and Lemma 5.3.1 imply a characterization of non-trivial parameter settings, which is interesting in its own right.

Corollary 5.3.7 (Classification of non-trivial parameter settings)**.** *A parameter setting $\boldsymbol{\alpha}$ is non-trivial if and only if it satisfies Table 5.2.*

Proof. One direction follows from Lemma 5.3.1. For the other direction, for any $n \geq 1$ consider the instances (n, x_p, y_p) constructed in Lemma 5.3.6, and let them use disjoint alphabets for different $p \in \mathcal{P}^*$. Then the concatenations $x := \bigcirc_{p \in \mathcal{P}} x_p$ and $y := \bigcirc_{p \in \mathcal{P}} x_p$ form an instance of $\mathrm{LCS}(\boldsymbol{\alpha})$, since for any parameter $p \in \mathcal{P}^*$ we have $p(x_p, y_p) = \Theta(n^{\alpha_p})$, and for all other instances $x_{p'}, y_{p'}$ the parameter p is $\mathcal{O}(n^{\alpha_p})$, and thus $p(x, y) = \Theta(n^{\alpha_p})$ by the Disjoint Alphabets Lemma. As we constructed instances of $\mathrm{LCS}(\boldsymbol{\alpha})$ for any $n \geq 1$, $\boldsymbol{\alpha}$ is non-trivial. □

We are ready to prove Lemma 5.3.4, finishing the proof overview for large alphabets.

Proof of Lemma 5.3.4. Let (n, x, y) be an instance of $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$. Since $\boldsymbol{\alpha}$ is non-trivial, we can use Corollary 5.3.7 and Lemma 5.3.6 to obtain strings x_p, y_p for $p \in \mathcal{P}^*$ such that (n, x_p, y_p) is an instance of $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$ and $p(x_p, y_p) = \Theta(n^{\alpha_p})$. Consider the concatenations $x' := x \circ \bigcirc_{p\in\mathcal{P}} x_p$ and $y' := y \circ \bigcirc_{p\in\mathcal{P}} y_p$. If we construct these strings over disjoint alphabets, we may use the Disjoint Alphabets Lemma to analyze x', y'. Since $p(x_p, y_p) = \Theta(n^{\alpha_p})$ and for all other pairs $(x_{p'}, y_{p'})$ the parameter p is bounded by $\mathcal{O}(n^{\alpha_p})$ as they are instances of $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$, we obtain $p(x', y') = \Theta(n^{\alpha_p})$ for any $p \in \mathcal{P}^*$. Hence, (x', y') is an instance of $\mathrm{LCS}(\boldsymbol{\alpha})$, and we can use the algorithm that we assume to exist for $\mathrm{LCS}(\boldsymbol{\alpha})$ to solve (x', y') in time $\mathcal{O}(n^{\beta})$. This also solves the given instance (n, x, y) of $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$ via $L(x, y) = L(x', y') - \sum_{p\in\mathcal{P}} L(x_p, y_p)$ (by the Disjoint Alphabets Lemma) where the right hand side can now be computed in time $\mathcal{O}(n)$ by Lemma 5.3.6. We finish the proof by observing that the time to construct x', y' is $\mathcal{O}(n)$. □

Small Alphabets. Proving our results for small constant alphabets poses additional challenges. While for $\alpha_\Sigma = 0$, the hard instances constructed in the proof of Theorem 5.3.3 could have any large constant alphabet size, for proving hardness in case $|\Sigma| = 2$ we need to construct binary strings. Luckily, our construction of Chapter 4 already produces binary strings. However, for our new 1vs1/2vs1 gadgets it is significantly harder to implement them on binary alphabet than for alphabet size at least 3, see Section 5.6.

Note that our proof of Lemma 5.3.4 fails for $\mathrm{LCS}(\boldsymbol{\alpha}, \Sigma)$ if $|\Sigma|$ is too small, since it produces strings over alphabet size at least $|\mathcal{P}| = 7$. In particular, for $|\Sigma| = 2$ we may not use the Disjoint Alphabets Lemma at all, rendering our proofs of Lemmas 5.3.4 and 5.3.6 and Corollary 5.3.7 completely useless. In fact, our results in this chapter show that *the Monotonicity Lemma (Lemma 5.3.4) is wrong for binary strings.* Indeed, for $|\Sigma| = 2$ our new algorithm has a running time $\tilde{\mathcal{O}}(n + \delta M/n)$ which is not monotone, and thus also the tight time bound $\tilde{\mathcal{O}}(n + \min\{d, \delta\Delta, \delta M/n\})$ is not monotone.

Hence, it is impossible to use general strings from $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha}, \Sigma)$ as a hard core for $\mathrm{LCS}(\boldsymbol{\alpha}, \Sigma)$, and instead we take strings with additional properties (see Lemmas 5.6.2 and 5.6.8). Moreover, instead of padding with new strings x_p, y_p for each parameter, we need an integrated construction where we control all parameters at once. This is a technically demanding task to which we devote a large part of this chapter (Section 5.8). Since the cases $|\Sigma| = 2$, $|\Sigma| = 3$, and $|\Sigma| \geq 4$ adhere to different relations of Table 5.2, these three cases have to be treated separately. Moreover, as in the case of large alphabet we consider cases $\alpha_\delta = \alpha_m$ and $\alpha_L = \alpha_m$.

We remark that in Section 5.8, we implicitly also characterize all non-trivial parameter settings $(\boldsymbol{\alpha}, \Sigma)$ as the settings satisfying Table 5.2. Indeed, we construct hard instances for each parameter setting satisfying Table 5.2, and thus establish that any such setting contains infinitely many instances and thus is non-trivial. The other direction follows from Lemma 5.3.1.

5.4 Parameter Relations

In this section, we prove relations among the studied parameters, summarized in Table 5.3. Some of these parameter relations can be found at various places in the literature, however, our set of relations is complete in the sense that any parameter setting $\boldsymbol{\alpha}$ is non-trivial if and only if it satisfies all our relations, see Corollary 5.3.7.

Consider a relation like $d(x, y) \leq L(x, y) \cdot m(x, y)$, given in Lemma 5.4.3(i) below. Fix exponents α_d, α_L, and α_m, and consider all instances x, y with $d(x, y) = \Theta(n^{\alpha_d})$, $L(x, y) = \Theta(n^{\alpha_L})$, and $m(x, y) = \Theta(n^{\alpha_m})$. Note that the relation may be satisfied for infinitely

Relation	Restriction	Reference
$L \leq m \leq n$		trivial
$L \leq d \leq M$		trivial
$\Delta \leq n$		trivial
$\delta \leq m$		trivial
$\delta \leq \Delta$		trivial
$\delta = m - L$		by definition
$\Delta = n - L$		by definition
$\lvert\Sigma\rvert \leq m$		Assumption 5.1.1
$n \leq M$		Assumption 5.1.1
$d \leq Lm$		Lemma 5.4.3(i)
$d \leq L^2\lvert\Sigma\rvert$		Lemma 5.4.3(ii)
$d \leq 2L(\Delta + 1)$		Lemma 5.4.4
$\lvert\Sigma\rvert \leq d$		Lemma 5.4.5
$\frac{L^2}{\lvert\Sigma\rvert} \leq M \leq 2Ln$		Lemma 5.4.6
$M \geq Lm/4$	if $\lvert\Sigma\rvert = 2$	Lemma 5.4.7
$M \geq nd/(5L)$	if $\lvert\Sigma\rvert = 2$	Lemma 5.4.9
$M \geq md/(80L)$	if $\lvert\Sigma\rvert = 3$	Lemma 5.4.10

Table 5.3: Relations between the parameters.

many instances if $\alpha_d \leq \alpha_L + \alpha_m$. On the other hand, if $\alpha_d > \alpha_L + \alpha_m$ then the relation is satisfied for only finitely many instances. This argument translates Table 5.3 into Table 5.2 (using $\alpha_n = 1$), thus generating a complete list of restrictions for non-trivial parameter settings.

Let x, y be any strings. In the remainder of this section, for convenience, we write $p = p(x, y)$ for any parameter $p \in \mathcal{P}^*$. Recall that by possibly swapping x and y, we may assume $m = |y| \leq |x| = n$. This assumption is explicit in our definition of parameter settings. For some other strings x, y considered in this chapter, this assumption may be violated. In this case, the parameter relations of Table 5.3 still hold after replacing n by $\max\{n, m\}$ and m by $\min\{n, m\}$, as well as Δ by $\max\{\Delta, \delta\}$ and δ by $\min\{\Delta, \delta\}$ (as the other parameters are symmetric).

Note that Assumption 5.1.1 (i.e., every symbol in Σ appears at least once in x and y) implies $|\Sigma| \leq m$ and ensures that any symbol of x has at least one matching symbol in y, and thus $M \geq n$.

We next list trivial relations. The length of the LCS L satisfies $L \leq m$. The numbers of deleted positions satisfy $\Delta = n - L \leq n$, $\delta = m - L \leq m$, and $\delta \leq \Delta$. Since any dominant pair is also a matching pair, we have $d \leq M$. Moreover, $d \geq L$ since for any $1 \leq k \leq L$ there is at least one k-dominant pair.

To prepare the proofs of the remaining relations, recall that we defined $L[i, j] = L(x[1..i], y[1..j])$. Moreover, observe that $L(x, y) \leq \sum_{\sigma \in \Sigma} \min\{\#_\sigma(x), \#_\sigma(y)\}$, which we typically exploit without explicit notice. Furthermore, we shall need the following two simple facts.

Observation 5.4.1. *For any $\sigma \in \Sigma$, we have $\#_\sigma(x) \leq L$ or $\#_\sigma(y) \leq L$.*

Proof. If some $\sigma \in \Sigma$ occurs at least $L + 1$ times in both x and y, then σ^{L+1} is an LCS of x and y of length $L + 1 > L$, which is a contradiction. □

Observation 5.4.2. *Fix $1 \leq k \leq L$ and $1 \leq \bar{i} \leq n$. Then there is at most one k-dominant pair $(\bar{i}, j)$ with $1 \leq j \leq m$, namely the pair $(\bar{i}, j^*)$ with $j^* = \min\{j \mid L[\bar{i}, j] = k\}$*

if it exists. Symmetrically for every $1 \leq k \leq n$ *and* $1 \leq \bar{j} \leq m$*, there is at most one* k*-dominant pair* $(i, \bar{j})$ *with* $1 \leq i \leq n$*.*

Proof. All pairs $(\bar{i}, j)$ with $j \neq j^*$ and $L[\bar{i}, j] = k$ satisfy $j \geq j^*$, so they are dominated by $(\bar{i}, j^*)$. □

We are set up to prove the more involved relations of Table 5.3. We remark that while the inequality $d \leq Lm$ is well known since the first formal treatment of dominant pairs, the bound $d \leq L^2|\Sigma|$ seems to go unnoticed in the literature.

Lemma 5.4.3. *It holds that (i)* $d \leq Lm$ *and (ii)* $d \leq L^2|\Sigma|$*.*

Proof. (i) Let $1 \leq k \leq L$. For any $1 \leq \bar{j} \leq m$, there is at most one k-dominant pair $(i, \bar{j})$ by Observation 5.4.2. This proves $|D_k| \leq m$ and thus $d = \sum_{i=1}^{L} |D_k| \leq Lm$.

(ii) Let $\sigma \in \Sigma$. By Observation 5.4.1, we may assume that $\#_\sigma(x) \leq L$ (the case $\#_\sigma(y) \leq L$ is symmetric). For any occurrence i_σ of σ in x and any $1 \leq k \leq L$, there can be at most one k-dominant pair (i_σ, j) by Observation 5.4.2. Hence, σ contributes at most L k-dominant pairs. Summing over all $\sigma \in \Sigma$ and $k = 1, \ldots, L$ yields the claim. □

Lemma 5.4.4. *We have* $d \leq 2L(\Delta + 1)$*.*

Proof. Fix an LCS z of x and y. Since z can be obtained by deleting at most $\Delta = n - L$ positions from x or by deleting at most $\delta = m - L$ positions from y, $x[1..i]$ and $y[1..j]$ contain $z[1..i - \Delta]$ and $z[1..j - \delta]$, respectively, as a subsequence. Hence, we have $\min\{i - \Delta, j - \delta\} \leq L[i, j] \leq \min\{i, j\}$.

Let $1 \leq k \leq L$. By the previous property, if $L[i, j] = k$ then (i) $k \leq i \leq k + \Delta$ or (ii) $k \leq j \leq k + \delta$. Note that for each $\bar{i} \in \{k, \ldots, k + \Delta\}$, we have (by Observation 5.4.2) at most one k-dominant pair $(\bar{i}, j)$, and similarly, for each $\bar{j} \in \{k, \ldots, k + \delta\}$, we have at most one k-dominant pair $(i, \bar{j})$. This proves $|D_k| \leq \Delta + \delta + 2 \leq 2(\Delta + 1)$, from which the claim follows. □

Lemma 5.4.5. *We have* $d \geq |\Sigma|$*.*

Proof. By Assumption 5.1.1, every symbol $\sigma \in \Sigma$ appears in x and y. Let i be minimal with $x[i] = \sigma$ and j be minimal with $y[j] = \sigma$. We show that (i, j) is a dominant pair of x, y, and thus $d \geq |\Sigma|$. Let $k = L[i, j] = L(x[1..i], y[1..j])$. Since $x[i] = y[j]$, we have $L[i - 1, j - 1] = k - 1$. Moreover, since the last symbol in $x[1..i]$ does not appear in $y[1..j - 1]$, it cannot be matched, and we obtain $L[i, j - 1] = L[i - 1, j - 1] = k - 1$. Similarly, $L[i - 1, j] = k - 1$. This proves that (i, j) is a k-dominant pair of x, y, as desired. □

Lemma 5.4.6. *We have (i)* $M \geq L^2/|\Sigma|$ *and (ii)* $M \leq 2Ln$*.*

Proof. (i) Let z be an LCS of x and y. We have $M = \sum_{\sigma \in \Sigma} \#_\sigma(x) \cdot \#_\sigma(y) \geq \sum_{\sigma \in \Sigma} \#_\sigma(z)^2$. By $\sum_{\sigma \in \Sigma} \#_\sigma(z) = L$ and the arithmetic-quadratic mean inequality, the result follows.

(ii) Let $\Sigma_w := \{\sigma \in \Sigma \mid \#_\sigma(w) \leq L\}$ for $w \in \{x, y\}$. By Observation 5.4.1, we have $\Sigma_x \cup \Sigma_y = \Sigma$. We can thus bound

$$M = \sum_{\sigma \in \Sigma} \#_\sigma(x) \cdot \#_\sigma(y) \leq \sum_{\sigma \in \Sigma_y} L \cdot \#_\sigma(x) + \sum_{\sigma \in \Sigma_x} L \cdot \#_\sigma(y) \leq L(n + m) \leq 2Ln. \quad \square$$

Small Alphabets. Not surprisingly, in the case of very small alphabets there are more relations among the parameters. The following relations are specific to $|\Sigma| = 2$ and $|\Sigma| = 3$.

Lemma 5.4.7. *Let $\Sigma = \{0, 1\}$. Then $M \geq Lm/4$.*

Proof. Let z be an LCS of x and y. Without loss of generality we may assume $\#_0(x) \geq n/2$ (by possibly exchanging 0 and 1). If $\#_0(y) \geq L/2$, then $M \geq \#_0(x) \cdot \#_0(y) \geq Ln/4 \geq Lm/4$. Otherwise we have $\#_0(y) < L/2 \leq m/2$, which implies $\#_1(y) \geq m/2$. By $\#_0(y) < L/2$, we must have that $\#_1(x) \geq L/2$, since otherwise $L \leq \min\{\#_0(x), \#_0(y)\} + \min\{\#_1(x), \#_1(y)\} \leq \#_0(y) + \#_1(x) < L$, which is a contradiction. Hence $M \geq \#_1(x) \cdot \#_1(y) \geq Lm/4$, proving the claim. □

The following lemma (which is also applicable for large alphabets) asserts that if most positions in x and y are the same symbol, say 0, then the number of dominant pairs is small.

Lemma 5.4.8. *Let 0 be a symbol in Σ and set $\lambda := \sum_{\sigma \in \Sigma \setminus \{0\}} \min\{\#_\sigma(x), \#_\sigma(y)\}$. Then $d \leq 5\lambda L$. In particular, for $\Sigma = \{0, 1\}$, we have $d \leq 5L \cdot \#_1(y)$.*

Proof. Let $1 \leq k \leq L$ and $\sigma \in \Sigma \setminus \{0\}$. By Observation 5.4.2, there are at most $\min\{\#_\sigma(x), \#_\sigma(y)\}$ k-dominant pairs (i, j) with $x[i] = y[j] = \sigma$. Hence in total, there are at most $\lambda \cdot L$ dominant pairs (i, j) with $x[i] = y[j] \neq 0$.

To count the remaining dominant pairs, which are contributed by 0, we use a similar argument to Lemma 5.4.4. Let $1 \leq k \leq L$ and consider any pair (i, j) with $x[i] = y[j] = 0$, say $x[i]$ is the ℓ_x-th occurrence of 0 in x and $y[j]$ is the ℓ_y-th occurrence of 0 in y. If (i, j) is a k-dominant pair then $k - \lambda \leq \min\{\ell_x, \ell_y\} \leq k$. Indeed, if $\min\{\ell_x, \ell_y\} < k - \lambda$, then

$$L[i, j] \leq \sum_{\sigma \in \Sigma} \min\{\#_\sigma(x[1..i]), \#_\sigma(y[1..j])\} \leq \min\{\ell_x, \ell_y\} + \lambda < k,$$

contradicting the definition of a k-dominant pair. Moreover, if $\min\{\ell_x, \ell_y\} > k$, then $0^{\min\{\ell_x, \ell_y\}}$ is a common subsequence of $x[1..i], y[1..j]$ of length strictly larger than k, which is again a contradiction to (i, j) being a k-dominant pair.

Hence, we have $k - \lambda \leq \ell_x \leq k$ or $k - \lambda \leq \ell_y \leq k$. Since any choice of ℓ_x uniquely determines i by Observation 5.4.2 (and symmetrically ℓ_y determines j), there are at most $2\lambda + 2$ k-dominant pairs with $x[i] = y[j] = 0$. In total, we have at most $(3\lambda + 2)L \leq 5\lambda L$ dominant pairs (note that $\lambda \geq 1$ by Assumption 5.1.1 and $|\Sigma| \geq 2$). □

Lemma 5.4.9. *If $\Sigma = \{0, 1\}$ then $M \geq nd/(5L)$.*

Proof. Without loss of generality assume that $\min\{\#_0(x), \#_0(y)\} \geq \min\{\#_1(x), \#_1(y)\}$ (by possibly exchanging 0 and 1). Then $\lambda = \min\{\#_1(x), \#_1(y)\}$ satisfies $\#_\sigma(y) \geq \lambda$ for all $\sigma \in \Sigma$. Thus, $M = \sum_{\sigma \in \Sigma} \#_\sigma(x) \cdot \#_\sigma(y) \geq (\#_0(x) + \#_1(x)) \cdot \lambda = \lambda n$. By Lemma 5.4.8, we have $\lambda \geq d/(5L)$ and the claim follows. □

For ternary alphabets, the following weaker relation holds.

Lemma 5.4.10. *If $\Sigma = \{0, 1, 2\}$ then $M \geq md/(80L)$.*

Proof. Let z be an LCS of x and y. We permute the symbols such that $\sigma = 0$ maximizes $\#_\sigma(z)$. Thus, we have $\#_0(x) \geq \#_0(z) \geq L/|\Sigma| = L/3$ and symmetrically $\#_0(y) \geq L/3$.

If $\#_0(y) \geq m/2$ then we have $M \geq \#_0(x) \cdot \#_0(y) \geq Lm/6 \geq dm/(18L)$ by Lemma 5.4.3(ii). Similarly, if $\#_0(x) \geq n/2$, we obtain $M \geq Ln/6 \geq dn/(18L) \geq$

$dm/(18L)$. Hence, it remains to consider the case $\#_0(x) \leq n/2$ and $\#_0(y) \leq m/2$. Let x', y' be the subsequences obtained by deleting all 0s from x and y, respectively, and note that $|x'|, |y'| \geq m/2$. Since x', y' have alphabet size 2, Lemma 5.4.7 is applicable and yields $M(x', y') \geq L(x', y') \cdot m(x', y')/4$. Observe that there is a common subsequence of x', y' of length at least $\lambda/2$, where $\lambda = \sum_{\sigma \in \Sigma \setminus \{0\}} \min\{\#_\sigma(x), \#_\sigma(y)\}$ (consider the longer subsequence of $1^{\min\{\#_1(x), \#_1(y)\}}$ and $2^{\min\{\#_2(x), \#_2(y)\}}$). Hence,

$$M(x, y) \geq M(x', y') \geq \frac{1}{4} \cdot L(x', y') \cdot m(x', y') \geq \frac{1}{4} \cdot \frac{\lambda}{2} \cdot \frac{m}{2} = \frac{\lambda m}{16}.$$

The claim now follows from $\lambda \geq d/(5L)$ as proven in Lemma 5.4.8. □

5.5 Technical Tools and Constructions

The aim of this section is to collect several technical results to prepare the constructions in the following sections. We start off with the simple fact that equal prefixes can be greedily matched.

Lemma 5.5.1 (Greedy Prefix Matching)**.** *For any strings w, x, y, we have $L(wx, wy) = |w| + L(x, y)$ and $d(wx, wy) = |w| + d(x, y)$.*

Proof. For the first statement, it suffices to prove the claim when $w = 1$ is a single symbol (by induction and renaming of symbols). In Fact 3.5.3(i), it was already proved that $\boldsymbol{\delta}_{\mathrm{LCS}}(1x, 1y) = \boldsymbol{\delta}_{\mathrm{LCS}}(x, y)$ (where $\boldsymbol{\delta}_{\mathrm{LCS}}(x, y) = |x| + |y| - 2 \cdot L(x, y)$), which immediately yields the claim.

For the second statement, let $x' = wx$ and $y' = wy$. For $i \in [|w|]$, we have $L(x'[1..i], y'[1..i]) = i$. Hence (i, i) is the unique i-dominant pair of x', y' and no other dominant pairs (i, j) with $i \leq |w|$ or $j \leq |w|$ exist. This yields $|w|$ dominant pairs. Consider now any (i, j) with $i = |w| + \bar{i}$ and $j = |w| + \bar{j}$ where $\bar{i} \in [|x|], \bar{j} \in [|y|]$. By the first statement, $L(x'[1..i], y'[1..j]) = |w| + L(x[1..\bar{i}], y[1..\bar{j}])$. Thus (i, j) is a $(|w| + k)$-dominant pair of x' and y' if and only if $(\bar{i}, \bar{j})$ is a k-dominant pair of x and y. This yields $d(x, y)$ additional dominant pairs, proving the claim. □

For bounding the number of dominant pairs from below we often use the following observation.

Observation 5.5.2. *For any strings a, x, b, y, we have $d(ax, by) \geq d(a, b)$.*

Proof. This trivially follows from the fact that any prefixes a', b' of a, b are also prefixes of ax,by. □

5.5.1 Generating Dominant Pairs

The dependency of d on the other parameters is quite complicated. Indeed, eight of the parameter relations of Section 5.4 involve the dominant pairs. Apostolico [Apo86] introduced the parameter under the initial impression that "it seems that whenever $[M]$ gets too close to mn, then this forces d to be linear in m". While we show that this intuition is somewhat misleading by constructing instances with high values of both M and d, it is a rather complex task to generate a desired number of dominant pairs while respecting given bounds on all other parameters (which is why we need to give different constructions for different parameter regimes).

	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	**1**	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	**2**	2	2	2	2	2	2	2	2	2	2	2	2	2
0	1	1	2	2	**3**	3	3	3	3	3	3	3	3	3	3	3
0	1	1	2	2	3	3	**4**	4	4	4	4	4	4	4	4	4
1	1	**2**	2	**3**	3	**4**	4	**5**	5	5	5	5	5	5	5	5
0	1	2	**3**	3	**4**	4	**5**	5	**6**	6	6	6	6	6	6	6
1	1	2	3	**4**	4	**5**	5	**6**	6	**7**	7	7	7	7	7	7
0	1	2	3	4	**5**	5	**6**	6	**7**	7	**8**	8	8	8	8	8
1	1	2	3	4	5	**6**	6	**7**	7	**8**	8	**9**	9	9	9	9
0	1	2	3	4	5	6	**7**	7	**8**	8	**9**	9	**10**	10	10	10
1	1	2	3	4	5	6	7	**8**	8	**9**	9	**10**	10	**11**	11	11
0	1	2	3	4	5	6	7	8	**9**	9	**10**	10	**11**	11	**12**	12
1	1	2	3	4	5	6	7	8	9	**10**	10	**11**	11	**12**	12	**13**

FIGURE 5.2: The L-table for the strings $a = (01)^{R+S}$ and $b = 0^R(01)^S$ with $R = 3, S = 5$ (where the entry in row j and column i denotes $L(a[1..i], b[1..j])$). The indicated dominant pairs visualize the results of Lemma 5.5.3.

The following lemma establishes the first such construction, illustrated in Figure 5.2. We remark that statements (iii) and (iv) are technical tools that we will only use for $|\Sigma| = \mathcal{O}(1)$ in Section 5.8.

Lemma 5.5.3 (Generating dominant pairs)**.** *Let $R, S \geq 0$ and define $a := (01)^{R+S}$ and $b := 0^R(01)^S$. The following statements hold.*

(i) We have $L(a, b) = |b| = R + 2S$.

(ii) We have $R \cdot S \leq d(a, b) \leq \min\{2(R+1), 5S\} \cdot (R + 2S) = \mathcal{O}(R \cdot S)$.

(iii) For any $\alpha, \beta, \beta' \geq 0$, we have $L(a1^\alpha, 0^\beta b 0^{\beta'}) = |b| = R + 2S$.

(iv) For any $\alpha, \beta, \beta' \geq 0$, we have $R \cdot S \leq d(a1^\alpha, 0^\beta b 0^{\beta'}) \leq 2(\max\{R + \alpha, \beta + \beta'\} + 1)(R + 2S)$.

Proof. All statements follow from the following fact.

(∗) For any $0 \leq s \leq S, s \leq r \leq R + s$ and $\beta \geq 0$, we have $L((01)^r, 0^\beta 0^R (01)^s) = r + s$.

To prove (∗), note that by Lemma 5.5.1 (reversing the strings) we have

$$\begin{aligned} L((01)^r, 0^{\beta+R}(01)^s) &= 2s + L((01)^{r-s}, 0^{\beta+R}) \\ &= 2s + \min\{\#_0((01)^{r-s}), \beta + R\} = 2s + (r - s) = r + s. \end{aligned}$$

Statement (i) now follows from setting $s = S$, $r = R + S$, and $\beta = 0$.

To see (iii), note that $L(a1^\alpha, 0^\beta b 0^{\beta'}) \geq L(a, b) = |b|$ by (i). For the upper bound, we compute

$$\begin{aligned} L(a1^\alpha, 0^\beta b 0^{\beta'}) &\leq \min\{\#_0(a1^\alpha), \#_0(0^\beta b 0^{\beta'})\} + \min\{\#_1(a1^\alpha), \#_1(0^\beta b 0^{\beta'})\} \\ &= \min\{R + S, R + S + \beta + \beta'\} + \min\{R + S + \alpha, S\} \\ &= R + 2S = L(a, b). \end{aligned}$$

To prove (ii), note that $d(a,b) \leq 5 \cdot \#_1(b) \cdot L(a,b) = 5S(R+2S)$ by Lemma 5.4.8. The bound $d(a,b) \leq 2(R+1) \cdot (R+2S)$ follows from (iv) by setting $\alpha = \beta = \beta' = 0$, hence it remains to prove (iv).

For the lower bound, we use $d(a1^\alpha, 0^\beta b 0^{\beta'}) \geq d(a, 0^\beta b)$ (by Observation 5.5.2) and consider $L'[r,s] := L((01)^r, 0^{\beta+R}(01)^s)$. We prove that for any $1 \leq s \leq S$, $s < r \leq R+s$, we have at least one $(r+s)$-dominant pair (i,j) with $2(r-1) < i \leq 2r$ and $\beta+R+2(s-1) < j \leq \beta+R+2s$. Indeed, $L'[r,s] = r+s$ (by (i)) implies that an $(r+s)$-dominant pair (i,j) with $i \leq 2r$ and $j \leq \beta+R+2s$ exists. If we had $i \leq 2(r-1)$, then by monotonicity of $L(x[1..i], y[1..j])$ in i and j we would have $L'[r-1,s] \geq r+s$, contradicting $L'[r-1,s] = r+s-1$ (by (i)). Thus, we obtain $i > 2(r-1)$, and symmetrically we have $j > \beta+R+2(s-1)$. Hence, for every $1 \leq s \leq S$, $s < r \leq R+s$, we have at least one dominant pair which is not counted for any other choice of r and s. Since for any $1 \leq s \leq S$ there are R choices for $s < r \leq R+s$, we conclude that $d(a,b) \geq S \cdot R$. For the upper bound, note that $\Delta(a1^\alpha, 0^\beta b 0^{\beta'}) = \max\{R+\alpha, \beta+\beta'\}$ by (iii), and hence Lemma 5.4.4 yields $d(a1^\alpha, 0^\beta b 0^{\beta'}) \leq 2 \cdot L(a1^\alpha, 0^\beta b 0^{\beta'}) \cdot (\Delta(a1^\alpha, 0^\beta b 0^{\beta'}) + 1) = 2(R+2S)(\max\{R+\alpha, \beta+\beta'\}+1)$. □

The previous construction uses alphabet size $|\Sigma| = 2$, enforcing $M(a,b) \geq L(a,b)^2/2$. If the desired number of matching pairs is much smaller than L^2, which can only be the case if the desired alphabet size is large, we have to use a more complicated construction exploiting the larger alphabet. To make the analysis more convenient, we first observe a simple way to bound the number of dominant pairs from below: if we can find k pairwise non-dominated index pairs (i,j) that have the same LCS value (of the corresponding prefixes) and whose predecessors $(i-1, j-1)$ have a strictly smaller LCS value, then at least $k/2$ dominant pairs exist.

Recall that for any strings x, y, we defined $L[i,j] = L(x[1..i], y[1..j])$.

Lemma 5.5.4. *Suppose that there are index pairs $(i_1,j_1), \ldots, (i_k,j_k)$ with $i_1 < i_2 < \cdots < i_k$ and $j_1 > j_2 > \cdots > j_k$ such that for some γ and all $1 \leq \ell \leq k$ we have $L[i_\ell, j_\ell] = \gamma$ and $L[i_\ell - 1, j_\ell - 1] = \gamma - 1$. Then the number of γ-dominant pairs of x and y is at least $k/2$.*

Proof. For each k, fix any γ-dominant pair $p_\ell = (i^*_\ell, j^*_\ell)$ that dominates (i_ℓ, j_ℓ). Note we may have $p_\ell = (i_\ell, j_\ell)$ if (i_ℓ, j_ℓ) is itself a dominant pair, and thus p_ℓ always exists. Set $i_0 := 1$. We argue that for every ℓ, we have $i_{\ell-1} \leq i^*_\ell \leq i_\ell$.

Note that for all ℓ, we have $L[i_\ell - 1, j_\ell - 1] < \gamma$ and hence $L[i,j] < \gamma$ for all $i < i_\ell$, $j < j_\ell$. Thus, we have either (1) $i^*_\ell = i_\ell$ (and $j^*_\ell \leq j_\ell$) or (2) $j^*_\ell = j_\ell$ (and $i^*_\ell \leq i_\ell$). Case (1) trivially satisfies $i_{\ell-1} \leq i^*_\ell \leq i_\ell$. Thus, it remains to argue that in case (2) we have $i_{\ell-1} \leq i^*_\ell$. Indeed, otherwise we have $i^*_\ell < i_{\ell-1}$ and $j^*_\ell = j_\ell < j_{\ell-1}$, which implies $L[i^*_\ell, j^*_\ell] < \gamma$ and (i^*_ℓ, j^*_ℓ) is no γ-dominant pair.

Note that the above property implies $p_\ell \neq p_{\ell+2}$ for all $1 \leq \ell \leq k-2$, since $i^*_\ell \leq i_\ell < i_{\ell+1} \leq i^*_{\ell+2}$. Thus, the number of γ-dominant pairs is bounded from below by $|\{p_\ell \mid 1 \leq \ell \leq k\}| \geq k/2$. □

Note that the previous lemma would not hold without the condition $L[i_\ell - 1, j_\ell - 1] = \gamma - 1$. We are set to analyze our next construction, which is illustrated in Figure 5.3.

Lemma 5.5.5 (Generating dominant pairs, large alphabet)**.** *Let $t \geq 2$, $1 \leq t' \leq t$, and $S \geq R \geq 1$. Over alphabet $\Sigma = \{1, \ldots, t\}$ we define the strings*

$$\begin{aligned} a &:= ((1 \ldots t) \circ (t' \ldots 1))^R \circ (1 \ldots t)^{S-R}, \\ b &:= (1 \ldots t)^S. \end{aligned}$$

	1	2	3	3	2	1	1	2	3	3	2	1	1	2	3	1	2	3	1	2	3	1	2	3
1	**1**	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	**2**	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	1	2	**3**	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
1	1	2	3	3	3	**4**	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
2	1	2	3	3	**4**	4	4	**5**	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
3	1	2	3	**4**	4	4	4	5	**6**	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
1	1	2	3	4	4	**5**	5	5	6	6	6	**7**	7	7	7	7	7	7	7	7	7	7	7	7
2	1	2	3	4	**5**	5	5	**6**	6	6	**7**	7	7	**8**	8	8	8	8	8	8	8	8	8	8
3	1	2	3	4	5	5	5	6	**7**	7	7	7	7	8	**9**	9	9	9	9	9	9	9	9	9
1	1	2	3	4	5	**6**	6	6	7	7	7	**8**	8	8	9	**10**	10	10	10	10	10	10	10	10
2	1	2	3	4	5	6	6	**7**	7	7	**8**	8	8	**9**	9	10	**11**	11	11	11	11	11	11	11
3	1	2	3	4	5	6	6	7	**8**	8	8	8	8	9	**10**	10	11	**12**	12	12	12	12	12	12
1	1	2	3	4	5	6	**7**	7	8	8	8	**9**	9	9	10	**11**	11	12	**13**	13	13	13	13	13
2	1	2	3	4	5	6	7	**8**	8	8	**9**	9	9	**10**	10	11	**12**	12	13	**14**	14	14	14	14
3	1	2	3	4	5	6	7	8	**9**	9	9	9	9	10	**11**	11	12	**13**	13	14	**15**	15	15	15
1	1	2	3	4	5	6	7	8	9	9	9	**10**	10	10	11	**12**	12	13	**14**	14	15	**16**	16	16
2	1	2	3	4	5	6	7	8	9	9	**10**	10	10	**11**	11	12	**13**	13	14	**15**	15	16	**17**	17
3	1	2	3	4	5	6	7	8	9	**10**	10	10	10	11	**12**	12	13	**14**	14	15	**16**	16	17	**18**
1	1	2	3	4	5	6	7	8	9	10	10	**11**	11	11	12	**13**	13	14	**15**	15	16	**17**	17	18
2	1	2	3	4	5	6	7	8	9	10	**11**	11	11	**12**	12	13	**14**	14	15	**16**	16	17	**18**	18
3	1	2	3	4	5	6	7	8	9	10	11	11	11	12	**13**	13	14	**15**	15	16	**17**	17	18	**19**

FIGURE 5.3: The L-table for strings a, b of Lemma 5.5.5 with $t = t' = 3$, $R = 2$, $S = 7$.

It holds that

(i) $L(a, b) = |b| = St$,

(ii) *Assume that* $S \geq R(t' + 1)$. *Then* $(St)(Rt')/8 \leq d(a, b) \leq 4(St)(Rt')$,

(iii) $tS^2 \leq M(a, b) \leq t(S + R)S$.

Proof. Note that (i) trivially follows from the fact that b is a subsequence of a. For (iii), observe that for all $\sigma \in \Sigma$, we have $S \leq \#_\sigma(a) \leq R + S$ and $\#_\sigma(b) = S$, from which the claim follows immediately. The upper bound of (ii) follows from $d(a, b) \leq 2 \cdot L(a, b) \cdot (\Delta(a, b) + 1) = 2(St)(Rt' + 1) \leq 4(St)(Rt')$ (by Lemma 5.4.4). To prove the remaining lower bound, we establish the following fact.

$(*)$ Let $w := 1 \dots t$, $w' := t' \dots 1$. Then $L((ww')^R, w^{R+k}) = Rt + k$ for all $0 \leq k \leq t'R$.

Let us postpone the proof and show that (ii) follows from $(*)$. Define $v := w^{S-R}$. For $0 \leq k \leq K := \min\{S - R, Rt'\}$ and $0 \leq \ell \leq (S - R - k)t$, we let $a(k, \ell) := (ww')^R v[1..\ell]$ and $b(k, \ell) := w^{R+k} v[1..\ell]$. Note that $a(k, \ell)$ and $b(k, \ell)$ are prefixes of a and b, respectively. By greedy suffix matching (i.e., Lemma 5.5.1 applied to the reversed strings) and $(*)$, we obtain

$$L(a(k, \ell), b(k, \ell)) = \ell + L((ww')^R, w^{R+k}) = Rt + k + \ell.$$

Hence, any $0 \leq k \leq K$, $1 \leq \ell \leq (S - R - k)t$ give rise to an index pair (i, j) with $L(a[1..i], b[1..j]) = L(a(k, \ell), b(k, \ell)) > L(a(k, \ell - 1), b(k, \ell - 1)) = L(a[1..i - 1], b[1..j - 1])$. Let I_γ denote the set of all such index pairs (i, j) that additionally satisfy $L(a[1..i], b[1..j]) = \gamma$. Then for any γ, no $(i, j) \in I_\gamma$ dominates another

$(i', j') \in I_\gamma$. Thus by Lemma 5.5.4, $d(a,b) \geq \sum_\gamma |I_\gamma|/2$. By counting all possible choices for k and ℓ, we obtain $\sum_\gamma |I_\gamma| = \sum_{0 \leq k \leq K} t(S-R-k) \geq tK(S-R)/2$. This yields $d(a,b) \geq t \cdot \min\{S-R, Rt'\} \cdot (S-R)/4$. For $S \geq R(t'+1)$, we have $S - R \geq S/2$ as well as $S - R \geq Rt'$ and the lower bound of (ii) follows.

To prove $(*)$, let $a' := (ww')^R$ and $b' := w^{R+k}$. For the lower bound, it is easy to see that we can completely match R of the copies of w in b' to copies of w in a', and at the same time match a single symbol in each of the remaining k copies of w in b to a single symbol in some copy of w' in a' (since $k \leq R|w'| = Rt'$). This yields $L(a', b') \geq R|w| + k = Rt + k$.

For the upper bound, we write $b' = \bigcirc_{j=1}^{R+k} b_j$ with $b_j := w$ and consider a partitioning $a' = \bigcirc_{j=1}^{R+k} a_j$ such that $L(a', b') = \sum_{j=1}^{R+k} L(a_j, b_j)$ (this is analogous to the optimal partition of Fact 3.5.1). For any a_j, let $w(a_j)$ denote the number of symbols that a_j shares with any occurrence of w (if, e.g., $a_j = xw'y$ for some prefix x of w and some suffix y of w, then $w(a_j) = |x| + |y|$). We first show that

$$L(a_j, b_j) \leq \begin{cases} 1 & \text{if } w(a_j) = 0, \\ \min\{w(a_j), |w|\} & \text{otherwise.} \end{cases} \tag{5.1}$$

Note that trivially $L(a_j, b_j) \leq |b_j| = |w|$. Hence for an upper bound, we may assume that $w(a_j) < |w|$, and in particular that a_j is a subsequence of $a'_j = xw'y$ for some prefix $x = \sigma_x \dots t$ of w and some suffix $y = 1 \dots \sigma_y$ of w with $\sigma_y \leq \sigma_x$, where $|x| + |y| = w(a_j)$. Note that any longest common subsequence z of a'_j and $b_j = w = 1 \dots t$ is an increasing subsequence of a'_j. Hence, if z starts with a symbol $\sigma' \geq \sigma_y$, then z is an increasing subsequence in $x\, t' \dots \sigma'$; easy inspection shows that in this case $|z| \leq \max\{|x|, 1\}$. If z starts with a symbol $\sigma' \leq \sigma_x$, then z is an increasing subsequence in $\sigma' \dots 1\, y$; again, one can see that $|z| \leq \max\{|y|, 1\}$ holds in this case. Thus, $L(a_j, b_j) \leq L(a'_j, b_j) = |z| \leq \max\{|x|, |y|, 1\} \leq \max\{|x| + |y|, 1\} = \max\{w(a_j), 1\}$, concluding the proof of (5.1).

Let $J = \{j \mid w(a_j) \geq 1\}$. We compute

$$\begin{aligned} L(a', b') = \sum_{j=1}^{R+k} L(a_j, b_j) &\leq \left(\sum_{j \in J} \min\{w(a_j), |w|\} \right) + (R + k - |J|) \\ &\leq \min\left\{ \sum_{j=1}^{R+k} w(a_j), |J| \cdot |w| \right\} + (R + k - |J|) \\ &\leq \min\{R \cdot |w|, |J| \cdot |w|\} + R + k - |J| \leq R|w| + k = Rt + k, \end{aligned}$$

where the last inequality follows from the observation that $|J| = R$ maximizes the expression $\min\{R \cdot |w|, |J| \cdot |w|\} - |J|$. This finishes the proof of $(*)$ and thus the lemma. □

5.5.2 Block Elimination and Dominant Pair Reduction

We collect some convenient tools for the analysis of later constructions. The first allows us to "eliminate" 0^ℓ-blocks when computing the LCS of strings of the form $x0^\ell y, 0^\ell z$, provided that ℓ is sufficiently large.

Lemma 5.5.6. *For any strings x, y, z and $\ell \geq \#_0(x) + |z|$ we have $L(x0^\ell y, 0^\ell z) = \ell + L(0^{\#_0(x)} y, z)$.*

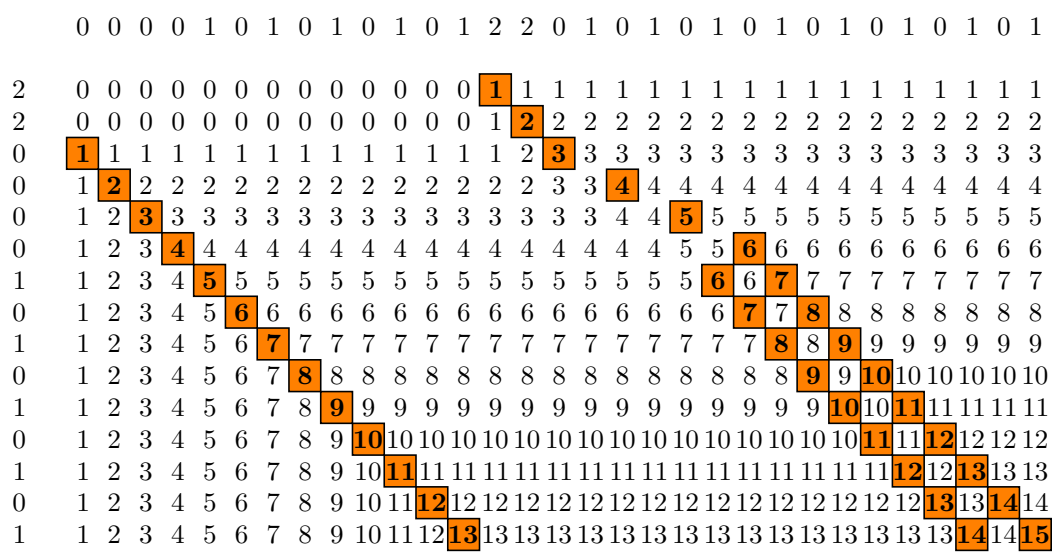


FIGURE 5.4: Illustration of Lemma 5.5.8. The strings $x' = y2^\ell x, y' = 2^\ell y$ are defined using $x = (01)^{R+S}$, $y = 0^R(01)^S$, $R = 3$, $S = 5$, and $\ell = 2$. The number of dominant pairs corresponding to x and y are significantly reduced compared to Figure 5.2.

Proof. Let $u := x0^\ell y$ and $v := 0^\ell z$. In case we match no symbol in the 0^ℓ-block of v with a symbol in $0^\ell y$ in u, then at most $\min\{\#_0(x), \ell\}$ symbols of the 0^ℓ-block of v are matched. The remainder z yields at most $|z|$ matched symbols. Otherwise, in case we match any symbol in the 0^ℓ-block of v with a symbol in $0^\ell y$ in u, then no symbol $\sigma \neq 0$ of x can be matched. Thus, in this case we may replace x by $0^{\#_0(x)}$. Together this case distinction yields $L(u, v) = \max\{\min\{\#_0(x), \ell\} + |z|, L(0^{\#_0(x)+\ell}y, 0^\ell z)\}$. Using Lemma 5.5.1, we obtain $L(u, v) = \max\{\#_0(x) + |z|, \ell + L(0^{\#_0(x)}y, z)\}$. The assumption $\ell \geq \#_0(x) + |z|$ now yields the claim. □

The following lemma bounds the number of dominant pairs of strings of the form $x' = yx$, $y' = zy$ by $d(x', y') = \mathcal{O}(|z| \cdot |y'|)$. If $|x'| \geq |y'|$, this provides a bound of $\mathcal{O}(\delta(x', y') \cdot m(x', y'))$ instead of the general, weaker bound $\mathcal{O}(\Delta(x', y') \cdot m(x', y'))$ of Lemma 5.4.4.

Lemma 5.5.7. *For any strings x, y, z, let $x' = yx$, $y' = zy$. Then*

$$d(x', y') \leq |y| \cdot (|z| + 1) + d(x', z) \leq |y| \cdot (|z| + 1) + |z|^2.$$

Proof. For every prefix $\tilde{y} = y'[1..j]$, we bound the number of dominant pairs (i, j) of x', y'. Clearly, all prefixes $\tilde{y}$ of z (i.e., $j \leq |z|$) contribute $d(x', z) \leq L(x', z) \cdot m(x', z) \leq |z|^2$ dominant pairs.

It remains to consider $\tilde{y} = z\,y[1..\ell]$ (i.e., $j = |z| + \ell$) for $\ell \in [|y|]$. For $i < \ell$, the string $\tilde{x} = x'[1..i] = y[1..i]$ is a subsequence of $\tilde{y}$, i.e., $L(\tilde{x}, \tilde{y}) = i$, but the prefix $z\,y[1..i]$ of $\tilde{y}$ already satisfies $L(\tilde{x}, z\,y[1..i]) = i$. Hence, there are no dominant pairs with $i < \ell$. Thus, consider $i \geq \ell$ and let $\tilde{x} = x'[1..i]$. Clearly, $y[1..j]$ is a common subsequence of $\tilde{x}, \tilde{y}$. This yields $L(\tilde{x}, \tilde{y}) \geq j = |\tilde{y}| - |z|$ and hence any such dominant pair (i, j) satisfies $j - |z| \leq L(x'[1..i], y'[1..j]) \leq j$. By Observation 5.4.2, there are at most $|z| + 1$ such dominant pairs for fixed j. This yields at most $|y| \cdot (|z| + 1)$ dominant pairs (i, j) with $|z| < j \leq |y'|$, concluding the proof. □

The above lemma gives rise to a surprising technique: Given strings x, y, we can build strings x', y' such that $L(x', y')$ lets us recover $L(x, y)$, but the number of dominant pairs may be reduced significantly, namely to a value $d(x', y') = \mathcal{O}(\delta(x, y) \cdot n(x, y))$, independently of $d(x, y)$.

Lemma 5.5.8 (Dominant Pair Reduction)**.** *Consider strings x, y and a number $\ell > |y| - L(x, y)$.*

(i) If 2 is a symbol not appearing in x, y, then $x' := y2^\ell x$ and $y' := 2^\ell y$ satisfy $L(x', y') = L(x, y) + \ell$ and $d(x, y) \leq 3\ell \cdot |y|$.

(ii) For any symbols $0, 1$ (that may appear in x, y) set $x'' := 0^k1^k y1^\ell 0^k 1^k x$ and $y'' := 1^\ell 0^k 1^k y$ with $k := 2|y| + |x| + 1$. Then $L(x'', y'') = L(x, y) + \ell + 2k$ and $d(x, y) \leq \mathcal{O}(\ell(|x| + |y| + \ell))$.

Proof. (i) Clearly, we have $L(x', y') \geq L(2^\ell, 2^\ell) + L(x, y) \geq \ell + L(x, y)$. For the other direction, let z be a common subsequence of x', y'. If z contains no 2, then by inspecting y' we obtain $|z| \leq |y| = L(x, y) + (|y| - L(x, y)) < L(x, y) + \ell$, so z is no LCS. Otherwise, if z contains a 2, then no symbol from the copy of y in x' can be matched by z, implying $|z| \leq L(2^\ell x, y') = \ell + L(x, y)$.

For the dominant pairs, we apply Lemma 5.5.7 to obtain $d(x', y') \leq |y|(\ell+1)+d(x', 2^\ell)$. Note that $d(x', 2^\ell) = d(2^\ell, 2^\ell) = \ell$, since we can delete all characters different from 2 in x' without affecting the dominant pairs of $x', 2^\ell$ (thus making x, y satisfy Assumption 5.1.1) and then apply Lemma 5.5.1. Hence, $d(x', y') \leq |y|(\ell + 1) + \ell \leq 3\ell \cdot |y|$.

(ii) The argument is slightly more complicated when the padding symbols may appear in x, y. Clearly, we have $L(x'', y'') \geq L(1^\ell 0^k 1^k x, 1^\ell 0^k 1^k y) \geq \ell + 2k + L(x, y)$. For the other direction, let z be a common subsequence of x'', y''. If z does not match any 0 in the 0^k-block of y'' with a symbol in a 0^k-block in x'', then from the 0^k-block of y'' we match at most $|y|+|x|$ symbols, and we obtain $|z| \leq (|y|+|x|)+|1^\ell 1^k y| = |x|+2|y|+\ell+k < \ell+2k$, since $k > |x| + 2|y|$, so z is no longest common subsequence. If z matches a 0 in the 0^k-block of y'' with a symbol in the left 0^k-block of x'', then no symbol in the 1^ℓ-block in y'' is matched by z, so we obtain $|z| \leq |0^k 1^k y| = 2k+L(x, y)+(|y|-L(x, y)) < 2k+L(x, y)+\ell$, so z is no longest common subsequence. The case remains that z matches some 0 in the 0^k-block of y'' with a symbol in the right 0^k-block of x''. Then the part $1^k y$ of y'' has to be matched to a subsequence of $0^k 1^k x$ in x''. This yields $|z| \leq \ell + k + L(0^k 1^k x, 1^k y)$. Since $k > |y|$ we can apply Lemma 5.5.6 (swapping the roles of 0 and 1) to obtain $L(0^k 1^k x, 1^k y) = k + L(x, y)$, so as desired we have $|z| \leq \ell + 2k + L(x, y)$.

For the dominant pairs, we apply Lemma 5.5.7 to obtain $d(x'', y'') \leq |0^k 1^k y| \cdot (\ell + 1) + \ell^2 = \mathcal{O}(\ell(|x| + |y| + \ell))$. □

5.6 Hardness for Large Alphabet

In this section, we consider a parameter setting $\boldsymbol{\alpha}$ satisfying the relations of Table 5.2, and we prove a lower bound for $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$ assuming OVH, thus proving Theorem 5.3.3. We split our proof into the two cases $\alpha_\delta = \alpha_m$ (where L may be small) and $\alpha_L = \alpha_m$ (where L is large). For readability, but abusing notation, for the target value $\lceil n^{\alpha_p} \rceil$ of parameter p we typically simply write p.

Throughout the section we may assume that

$$\alpha_L, \alpha_m, \alpha_\delta, \alpha_\Delta > 0 \quad \text{and} \quad \alpha_d > 1, \tag{LB}$$

since otherwise the known $\tilde{\mathcal{O}}(n + \min\{d, \delta m, \delta\Delta\})$ algorithm runs in (near-)optimal time $\tilde{\mathcal{O}}(n)$ and there is nothing to show (here we used the parameter relations $d \leq Lm$ and $L, m, \delta, \Delta \leq n$).

5.6.1 Small LCS

Assume $\alpha_\delta = \alpha_m$, i.e., $\delta = \Theta(m)$. In this case, the longest common subsequence might be arbitrarily small, i.e., any value $0 < \alpha_L \leq \alpha_m$ is admissible.

Hard Core

At the heart of our constructions lies our reduction from OV to LCS that was given in Chapter 4. Recall that it was obtained by combining the alignment gadget for LCS (Lemma 4.3.3, or alternatively the more general construction of Lemma 4.4.5, specialized to LCS) with the general framework result of Theorem 4.1.3. We summarize the reduction here together with the properties needed for the purposes of this section.

Lemma 5.6.1. *Let two sets $\mathcal{A} = \{a_1, \ldots, a_A\}$ and $\mathcal{B} = \{b_1, \ldots, b_B\}$ of vectors in $\{0,1\}^D$ with $A \geq B$ be given. In time $\mathcal{O}(AD)$, we can construct strings $x_1, \ldots, x_{2A}$ and $y_1, \ldots, y_B$ over $\{0,1\}$ and $\gamma, \gamma' = \Theta(D)$ such that the strings x and y defined by*

$$\begin{aligned} x &:= \quad x_1\ 0^\gamma\ x_2\ 0^\gamma\ \ldots\ x_{2A-1}\ 0^\gamma\ x_{2A}, \\ y &:= 0^{A\gamma'}\ y_1\ 0^\gamma\ y_2\ 0^\gamma\ \ldots\ y_{B-1}\ 0^\gamma\ y_B\ 0^{A\gamma'}, \end{aligned}$$

satisfy the following properties:

(i) We can compute some ρ in time $\mathcal{O}(AD)$ such that $L(x,y) \geq \rho$ if and only if there is a pair i, j with $\langle a_i, b_j \rangle = 0$.

(ii) $|x|, |y| = \mathcal{O}(AD)$.

(iii) $\#_1(y) = \mathcal{O}(BD)$.

(iv) For all $\beta \geq 0$, we have $L(x, 0^\beta y) = L(x,y)$.

Proof. Note that when applying the proof of Theorem 4.1.3 to LCS, we apply the alignment gadget thrice: Twice to obtain normalized vector gadgets $x_1, \ldots, x_{2A}$ and $y_1, \ldots, y_B$ of length $\mathcal{O}(D)$, and another time to combine these strings to x and y as stated above. Claim (ii) hence follows directly from the construction, while Claim (i) follows from the proof of correctness of Theorem 4.1.3 and Lemma 4.3.3. Claim (iii) is immediate by construction, since in y, only the y_i's can contain ones, and these B strings are of length $\mathcal{O}(D)$.

For (iv), observe that in Lemma 4.3.3, the value of γ' is chosen sufficiently large to satisfy $A\gamma' \geq \#_0(x)$. This already yields (iv), since any common subsequence z of x and $0^\beta y$ starts with at most $\#_0(x)$ symbols 0, which can all be matched to the initial $0^{A\gamma'}$-block of y, so z is also a common subsequence of x and y, and we obtain $L(x,y) = L(x, 0^\beta y)$. □

Constant Alphabet

First assume $\alpha_\Sigma = 0$ and thus $|\Sigma| = \mathcal{O}(1)$. Consider any $n \geq 1$ and target values $p = \lceil n^{\alpha_p} \rceil$ for $p \in \mathcal{P}$. Let $\mathcal{A} = \{a_1, \ldots, a_A\}$, $\mathcal{B} = \{b_1, \ldots, b_B\} \subseteq \{0,1\}^D$ be a given OV instance with $D = n^{o(1)}$ and where we set $A := \lfloor L/D \rfloor$ and $B := \lfloor d/(LD) \rfloor$. Note that

$A = n^{\alpha_L - o(1)} = n^{\Omega(1)}$ and $B = n^{\alpha_d - \alpha_L - o(1)} = n^{\Omega(1)}$ by (LB) and $\alpha_L \leq 1$. Also note that UOVH implies that solving such OV instances takes time $(AB)^{1-o(1)} = n^{\alpha_d - o(1)} = d^{1-o(1)}$.

Construct strings x, y as in Lemma 5.6.1. Then from the LCS length $L(x, y)$ we can infer whether $\mathcal{A}, \mathcal{B}$ has an orthogonal pair of vectors by Lemma 5.6.1(i). Moreover, this reduction runs in time $\mathcal{O}(AD) = \mathcal{O}(L) = \mathcal{O}(d^{1-\varepsilon})$ for sufficiently small $\varepsilon > 0$ (since $\alpha_L \leq \alpha_m \leq 1 < \alpha_d$ by Table 5.2 and (LB)). We claim that (n, x, y) is an instance of $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$. This shows that any algorithm solving $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$ in time $\mathcal{O}(d^{1-\varepsilon})$ implies an algorithm for our OV instances with running time $\mathcal{O}(d^{1-\varepsilon})$, contradicting UOVH. Hence, in the current case $\alpha_\delta = \alpha_m$ and $\alpha_\Sigma = 0$, any algorithm for $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$ takes time $d^{1-o(1)}$, proving part of Theorem 5.3.3.

It remains to show the claim that (n, x, y) is an instance of $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$. Using the parameter relations $L \leq m \leq n$, Lemma 5.6.1(ii), and the definition of A, we have $L(x, y) \leq m(x, y) \leq n(x, y) = \max\{|x|, |y|\} = \mathcal{O}(AD) = \mathcal{O}(L) = \mathcal{O}(m) = \mathcal{O}(n)$, so indeed $p(x, y) = \mathcal{O}(p) = \mathcal{O}(n^{\alpha_p})$ for $p \in \{L, m, n\}$. Similarly, we obtain $\delta(x, y) \leq \Delta(x, y) \leq n(x, y) = \mathcal{O}(m) = \mathcal{O}(\delta) = \mathcal{O}(\Delta)$, where the fourth bound holds in the current case $\alpha_\delta = \alpha_m$. Since x, y use the binary alphabet $\{0, 1\}$, we have $|\Sigma(x, y)| = 2 = \mathcal{O}(n^{\alpha_\Sigma})$. For the number of matching pairs we have $M(x, y) \leq n(x, y)^2 = \mathcal{O}((AD)^2) = \mathcal{O}(L^2)$. Since we are in the case $\alpha_\Sigma = 0$, from the parameter relation $M \geq L^2/|\Sigma|$ (Lemma 5.4.6(i)) we obtain $L^2 = \mathcal{O}(M)$ and thus also $M(x, y)$ is sufficiently small. Finally, we use Lemmas 5.4.8 and 5.6.1(iii) to bound $d(x, y) = \mathcal{O}(L(x, y) \cdot \#_1(y)) = \mathcal{O}(AD \cdot BD)$, which by definition of A, B is $\mathcal{O}(d)$. This proves that (n, x, y) belongs to $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$.

We remark that our proof also yields the following lemma, which we will use for small alphabets in Section 5.8.

Lemma 5.6.2. *Let $\boldsymbol{\alpha}$ be a parameter setting satisfying Table 5.2 with $\alpha_\Sigma = 0$ and $\alpha_\delta = \alpha_m$. There is a constant $\gamma \geq 1$ such that any algorithm for $\mathrm{LCS}^{\gamma}_{\leq}(\boldsymbol{\alpha}, \{0, 1\})$ takes time $d^{1-o(1)}$, unless* OVH *fails. This holds even restricted to instances (n, x, y) of $\mathrm{LCS}^{\gamma}_{\leq}(\boldsymbol{\alpha}, \{0, 1\})$ with $|x|, |y| \leq \gamma \cdot n^{\alpha_L}$ and $\#_1(y) \leq \gamma \cdot n^{\alpha_d - \alpha_L}$ satisfying $L(x, 0^\beta y) = L(x, y)$ for all $\beta \geq 0$.*

Superconstant Alphabet

To tackle the general case $\alpha_\Sigma \geq 0$ (while still assuming $\alpha_\delta = \alpha_m$), we use the following fact which is similar to the Disjoint Alphabets Lemma.

Lemma 5.6.3 (Crossing Alphabets)**.** *Let $\Sigma_1, \ldots, \Sigma_k$ be disjoint alphabets and let x_i, y_i be strings over alphabet Σ_i. Consider $x := x_1 \ldots x_k$ and $y := y_k \ldots y_1$, i.e., the order in y is reversed. For any parameter $p \in \{n, m, |\Sigma|, M, d\}$, we have $p(x, y) = \sum_{i=1}^{k} p(x_i, y_i)$. Moreover, $L(x, y) = \max_i L(x_i, y_i)$.*

Proof. The statement is trivial for the string lengths n, m, alphabet size $|\Sigma|$, and number of matching pairs M. For the LCS length L we observe that any common subsequence z that matches a symbol in Σ_i cannot match any symbols in other alphabets, which yields $L(x, y) \leq \max_i L(x_i, y_i)$. Since any common subsequence of x_i, y_i is also a common subsequence of x, y, we obtain equality.

Since every dominant pair is also a matching pair, every dominant pair of x, y stems from prefixes $x_1 \ldots x_{j-1} x'$ and $y_k \ldots y_{j+1} y'$, with x' a prefix of x_j and y' a prefix of y_j for some j. Since $L(x_1 \ldots x_{j-1} x', y_k \ldots y_{j+1} y') = L(x', y')$, we obtain that the dominant pairs of x, y of the form $x_1 \ldots x_{j-1} x'$, $y_k \ldots y_{j+1} y'$ are in one-to-one correspondence with the dominant pairs of x_j, y_j. Since these dominant pairs of x, y are incomparable, this yields the claim for parameter d. □

We make use of the above lemma by invoking the following construction.

Definition 5.6.4. *Let $\Sigma_1, \dots, \Sigma_t$ be a collection of disjoint two-element alphabets. For any string z over $\{0,1\}$ and Σ_i, let $z \uparrow \Sigma_i$ denote the string z lifted to Σ_i, i.e., we replace the symbols $\{0,1\}$ in z bijectively by Σ_i. Then for given $x_1, \dots, x_t, y_1, \dots, y_t \in \{0,1\}^*$ we construct*

$$\begin{aligned} \mathrm{cr}^{\mathrm{X}}(x_1, \dots, x_t) := \quad & x_1 \uparrow \Sigma_1 \quad & x_2 \uparrow \Sigma_2 \quad & \dots \quad & x_t \uparrow \Sigma_t, \\ \mathrm{cr}^{\mathrm{Y}}(y_1, \dots, y_t) := \quad & y_t \uparrow \Sigma_t \quad & y_{t-1} \uparrow \Sigma_{t-2} \quad & \dots \quad & y_1 \uparrow \Sigma_1. \end{aligned}$$

We adapt the construction from Lemma 5.6.1 using the following trick that realizes an "OR" of $t = \mathcal{O}(\Sigma)$ instances, without significantly increasing the parameters d and M.

Consider any $n \geq 1$ and target values $p = \lceil n^{\alpha_p} \rceil$ for $p \in \mathcal{P}$. Let $\mathcal{A} = \{a_1, \dots, a_A\}$, $\mathcal{B} = \{b_1, \dots, b_B\} \subseteq \{0,1\}^D$ be a given OV instance with $D = n^{o(1)}$ and where we set $A = \lfloor \frac{d}{\min\{L, \sqrt{d}\} \cdot D} \rfloor$ and $B = \lfloor \frac{\min\{L, \sqrt{d}\}}{D} \rfloor$. Note that UOVH implies that solving such OV instances takes time $(AB)^{1-o(1)} = n^{\alpha_d - o(1)} = d^{1-o(1)}$. Since clearly $A \geq B$, we can partition $\mathcal{A}$ into $t := \lceil A/B \rceil$ groups $\mathcal{A}_1, \dots, \mathcal{A}_t$ of size B (filling up the last group with all-ones vectors). Using the relation $d \leq L^2|\Sigma|$ (Lemma 5.4.3(ii)), we obtain $t = \mathcal{O}(d/L^2 + 1) = \mathcal{O}(\Sigma)$.

For each $i = 1, \dots, t$ we construct strings x_i and y_i for the sets $\mathcal{A}_i$ and $\mathcal{B}$ using Lemma 5.6.1. Finally, we set $x := \mathrm{cr}^{\mathrm{X}}(x_1, \dots, x_t)$ and $y := \mathrm{cr}^{\mathrm{Y}}(y_1, \dots, y_t)$. By the Crossing Alphabets Lemma and Lemma 5.6.1(i), from $L(x,y)$ we can infer whether $\mathcal{A}, \mathcal{B}$ has an orthogonal pair of vectors. We claim that (n, x, y) is an instance of $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$. This shows that any algorithm solving $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$ in time $\mathcal{O}(d^{1-\varepsilon})$ implies an algorithm for our OV instances with running time $\mathcal{O}(d^{1-\varepsilon})$, contradicting UOVH. Hence, in the current case $\alpha_\delta = \alpha_m$, any algorithm for $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$ takes time $d^{1-o(1)}$, proving part of Theorem 5.3.3.

It remains to show the claim that (n, x, y) is an instance of $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$. This is similar to the proof for the case $\alpha_\Sigma = 0$, additionally using the Crossing Aphabets Lemma. Specifically, we obtain $m(x,y) \leq n(x,y) = |x| = \sum_{i=1}^{t} |x_i| = \mathcal{O}(t \cdot BD) = \mathcal{O}(AD) = \mathcal{O}(\max\{d/L, \sqrt{d}\})$, which is at most $\mathcal{O}(m) = \mathcal{O}(n)$ using the parameter relations $d \leq Lm \leq m^2$ (Lemma 5.4.3(i)). Similarly, we obtain $\delta(x,y) \leq \Delta(x,y) \leq n(x,y) = \mathcal{O}(m) = \mathcal{O}(\delta) = \mathcal{O}(\Delta)$, where the fourth bound holds in the current case $\alpha_m = \alpha_\delta$. For L, we obtain $L(x,y) = \max_i L(x_i, y_i) \leq |y_i| = \mathcal{O}(BD) = \mathcal{O}(L)$. Since $t = \mathcal{O}(\Sigma)$, we have $|\Sigma(x,y)| = \mathcal{O}(\Sigma)$. Using the parameter relation $d \leq M$, we have $d(x,y) \leq M(x,y) = \sum_{i=1}^{t} M(x_i, y_i) \leq t \cdot |x_i| \cdot |y_i| = t \cdot \mathcal{O}((BD)^2) = \mathcal{O}(AD \cdot BD) = \mathcal{O}(d) = \mathcal{O}(M)$. This proves that (n, x, y) belongs to $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$.

5.6.2 Large LCS

We turn to the case that $\alpha_L = \alpha_m$, i.e., $L = \Theta(m)$. Here, the number of deletions in the shorter string might be arbitrary small, i.e., any value $0 < \alpha_\delta \leq \alpha_m$ is admissible. In this case, our construction of Lemma 5.6.1 is not applicable as is. Instead, we design new 1vs1/2vs1 gadgets for constructing hard strings for small δ, which can be seen as one of our main technical contributions in this chapter.

Hard Core

The following lemma (which effectively constitutes an intermediate step when reducing OV to LCS using Theorem 4.1.3 with the alignment gadget for LCS) yields the basic method to embed sets of vectors into strings x and y.

Lemma 5.6.5. *Let two sets $\mathcal{A} = \{a_1, \dots, a_A\}$ and $\mathcal{B} = \{b_1, \dots, b_B\}$ of vectors in $\{0,1\}^D$ be given. In time $\mathcal{O}((A+B)D)$, we can construct strings $x_1, \dots, x_A$ of length ℓ_{X} and $y_1, \dots, y_B$ of length ℓ_{Y} over alphabet $\{0,1\}$, as well as integers $\rho_1 < \rho_0$, such that for all $i \in [A], j \in [B]$ we have*

(i) $\ell_{\mathrm{Y}} \leq \ell_{\mathrm{X}} = \mathcal{O}(D)$,

(ii) $L(x_i, y_j) = \rho_0$ *if* $\langle a_i, b_j \rangle = 0$,

(iii) $L(x_i, y_j) = \rho_1$ *if* $\langle a_i, b_j \rangle \neq 0$, *and*

(iv) $L(x_i, y_j) > \ell_{\mathrm{Y}}/2$.

Proof. Using Theorem 4.1.3 and Lemma 4.3.3, we can construct normalized vector gadgets as in Claim 4.1.5, i.e., strings $x'_1, \dots, x'_A$ of length $\ell'_{\mathrm{X}} = \mathcal{O}(D)$ and $y'_1, \dots, y'_B$ of length $\ell'_{\mathrm{Y}} = \mathcal{O}(D)$, as well as integers $\rho'_1 < \rho'_0$ that satisfy $L(x'_i, y'_j) = \rho'_0$ if $\langle a_i, b_j \rangle = 0$ and $L(x'_i, y'_j) = \rho'_1$ otherwise. To additionally enforce conditions (i) and (iv), we define $x_i := 1^{\ell'_{\mathrm{Y}}} 0^{\ell'_{\mathrm{Y}}+1} x'_i$ and $y_j := 0^{\ell'_{\mathrm{Y}}+1} y'_j$. Since $L(x_i, y_j) = L(x'_i, y'_j) + \ell'_{\mathrm{Y}} + 1$ by Lemmas 5.5.6 and 5.5.1, we thus obtain conditions (ii) and (iii) for $\rho_0 := \rho'_0 + \ell'_{\mathrm{Y}} + 1$ and $\rho_1 := \rho'_1 + \ell'_{\mathrm{Y}} + 1$. Since by definition $\ell_{\mathrm{Y}} = 2\ell'_{\mathrm{Y}} + 1$ holds, the first condition follows directly and the trivial bound $L(x_i, y_j) \geq \ell'_{\mathrm{Y}} + 1 > \ell_{\mathrm{Y}}/2$ shows that the last condition is fulfilled. □

1vs1/2vs1 Gadget. The aim of the following construction is to embed given strings $y_1, \dots, y_Q$ into a string y and strings $x_1, \dots, x_P$ into x, where $P = \Theta(Q)$, such that in an LCS each y_j is either aligned to a single string x_i or to several strings $x_i, x_{i+1}, \dots, x_{i'}$. In the first case, $|y_j| - L(x_i, y_j)$ characters of y_j are not contained in an LCS of x and y, while in the second case y_j can be completely aligned. By choosing $P = 2Q - N$ for an arbitrary $1 \leq N \leq Q$, it will turn out that the LCS aligns N strings y_j to a single partner x_i, and the remaining $Q - N$ strings y_j to two strings x_i, x_{i+1} each. Thus, only N strings y_j are not completely aligned.

To formalize this intuition, let $P \geq Q$. We call a set $\Lambda = \{(i_1, j_1), \dots, (i_k, j_k)\}$ with $0 \leq k \leq Q$ and $1 \leq i_1 < i_2 < \cdots < i_k \leq P$ and $1 \leq j_1 \leq j_2 \leq \cdots \leq j_k \leq Q$ a *(partial) multi-alignment*[7]. Let $\Lambda(j) = \{i \mid (i,j) \in \Lambda\}$. We say that every $j \in [Q]$ with $|\Lambda(j)| = k$ is *k-aligned*. We will also refer to a 1-aligned $j \in [Q]$ as being *uniquely aligned to i*, where $\Lambda(j) = \{i\}$. Every $j \in [Q]$ with $\Lambda(j) = \emptyset$ is called *unaligned*. Note that each $i \in [P]$ occurs in at most one $(i,j) \in \Lambda$. We denote the set of multi-alignments as $\mathbf{\Lambda}^{\mathrm{multi}}_{P,Q}$.

We will also need the following specialization of multi-alignments. We call a multi-alignment $\Lambda \in \mathbf{\Lambda}^{\mathrm{multi}}_{P,Q}$ a *(1,2)-alignment*, if each j is either 1-aligned or 2-aligned. Let $\mathbf{\Lambda}^{1,2}_{P,Q}$ denote the set of all (1,2)-alignments.

Given strings $x_1, \dots, x_P$ of length ℓ_{X} and $y_1, \dots, y_Q$ of length ℓ_{Y}, we define the *value* $\mathbf{v}(\Lambda)$ of a multi-alignment $\Lambda \in \mathbf{\Lambda}^{\mathrm{multi}}_{P,Q}$ as $\mathbf{v}(\Lambda) = \sum_{j=1}^{Q} v_j$ where

$$v_j := \begin{cases} 0 & \text{if } j \text{ is unaligned,} \\ L(x_i, y_j) & \text{if } j \text{ is uniquely aligned to } i, \\ \ell_{\mathrm{Y}} & \text{if } j \text{ is } k\text{-aligned for } k \geq 2. \end{cases}$$

[7] The analysis of the 1vs1/2vs1 gadget mimics the structure of the analysis of the alignment gadget. In contrast to the (partial) alignments defined in Chapter 4, intuitively a multi-alignment allows objects y_j to be aligned to multiple partners $x_{i_1}, \dots, x_{i_k}$ instead of at most one partner x_i.

Lemma 5.6.6. *Given strings $x_1, \ldots, x_P$ of length ℓ_{X} and $y_1, \ldots, y_Q$ of length ℓ_{Y}, construct*

$$\begin{aligned} x &:= \mathrm{G}(x_1)\,\mathrm{G}(x_2)\,\ldots\,\mathrm{G}(x_P), \\ y &:= \mathrm{G}(y_1)\,\mathrm{G}(y_2)\,\ldots\,\mathrm{G}(y_Q), \end{aligned}$$

where $\mathrm{G}(w) := 0^{\gamma_1}\,1^{\gamma_2}\,(01)^{\gamma_3}\,w\,1^{\gamma_3}$ with $\gamma_3 := \ell_{\mathrm{X}} + \ell_{\mathrm{Y}}$, $\gamma_2 := 8\gamma_3$ and $\gamma_1 := 6\gamma_2$. Then we have

$$\max_{\Lambda \in \mathbf{\Lambda}^{1,2}_{P,Q}} \mathbf{v}(\Lambda) \leq L(x,y) - Q(\gamma_1+\gamma_2+3\gamma_3) \leq \max_{\Lambda \in \mathbf{\Lambda}^{\mathrm{multi}}_{P,Q}} \mathbf{v}(\Lambda). \tag{5.2}$$

Proof. For the first inequality of (5.2), let $\Lambda \in \mathbf{\Lambda}^{1,2}_{P,Q}$. For every y_j, we define $z_j = \bigcirc_{i \in \Lambda(j)} \mathrm{G}(x_i)$. Consider a 1-aligned j and let $i \in [P]$ be the index j is uniquely aligned to. We have that $z_j = \mathrm{G}(x_i) = 0^{\gamma_1}1^{\gamma_2}(01)^{\gamma_3}x_i 1^{\gamma_3}$ and hence by Lemma 5.5.1, we obtain $L(z_j, \mathrm{G}(y_j)) = \gamma_1+\gamma_2+3\gamma_3 + L(x_i, y_j) = \gamma_1+\gamma_2+3\gamma_3+v_j$. Likewise, consider a 2-aligned j and let $i, i' \in [P]$ be such that $\Lambda(j) = \{i, i'\}$. Then $z_j = \mathrm{G}(x_i)\mathrm{G}(x_{i'})$. We compute

$$\begin{aligned} L(z_j, \mathrm{G}(y_j)) &= \gamma_1+\gamma_2+3\gamma_3 + L(x_i 0^{\gamma_1}1^{\gamma_2}(01)^{\gamma_3}x_{i'}, y_j) \\ &\geq \gamma_1+\gamma_2+3\gamma_3 + L((01)^{\gamma_3}, y_j) \\ &= \gamma_1+\gamma_2+3\gamma_3+\ell_{\mathrm{Y}} = \gamma_1+\gamma_2+3\gamma_3+v_j, \end{aligned}$$

where the first line follows from Lemma 5.5.1, the second line from monotonicity and the third line from $\gamma_3 \geq \ell_{\mathrm{Y}} = |y_j|$. Observe that $z_1 z_2 \ldots z_Q$ is a subsequence of x. We conclude that

$$L(x,y) \geq \sum_{j=1}^{Q} L(z_j, \mathrm{G}(y_j)) = Q(\gamma_1+\gamma_2+3\gamma_3) + \sum_{j=1}^{Q} v_j.$$

It remains to prove the second inequality of (5.2). Write $x = z_1 z_2 \ldots z_Q$ such that $L(x,y) = \sum_{j=1}^{Q} L(z_j, \mathrm{G}(y_j))$. We define a multi-alignment Λ by letting $(i,j) \in \Lambda$ if and only if z_j contains strictly more than half of the 0^{γ_1}-block of $\mathrm{G}(x_i)$. Note that the thus defined set satisfies the definition of a multi-alignment, since no two z_j's can contain more than half of $\mathrm{G}(x_i)$'s 0^{γ_1}-block and if $(i,j),(i',j') \in \Lambda$, then $j < j'$ implies $i < i'$. It remains to show that $L(z_j, \mathrm{G}(y_j)) \leq \gamma_1+\gamma_2+3\gamma_3+v_j$ for all j to prove the claim.

In what follows, we use the shorthand $H(w) := 1^{\gamma_2}(01)^{\gamma_3}w1^{\gamma_3}$. Note that $G(w) = 0^{\gamma_1}H(w)$. Consider an unaligned $j \in [Q]$. By definition, z_j is a subsequence of $0^{\gamma_1/2}H(x_i)0^{\gamma_1/2}$ for some $i \in [P]$. We can thus bound (using Lemma 5.5.1)

$$L(z_j, \mathrm{G}(y_j)) \leq L(0^{\gamma_1/2}H(x_i)0^{\gamma_1/2}, 0^{\gamma_1}H(y_j)) = \frac{\gamma_1}{2} + L(H(x_i)0^{\gamma_1/2}, 0^{\gamma_1/2}H(y_j)).$$

By Lemma 5.5.6 with $\ell := \gamma_1/2 \geq 2\gamma_2+6\gamma_3+\ell_{\mathrm{X}}+\ell_{\mathrm{Y}} = |H(x_i)|+|H(y_j)| \geq \#_0(H(x_i))+|H(y_j)|$, we obtain

$$\begin{aligned} L(H(x_i)0^{\gamma_1/2}, 0^{\gamma_1/2}H(y_j)) &= \gamma_1/2 + L(0^{\#_0(H(x_i))}, H(y_j)) \\ &\leq \gamma_1/2 + \#_0(H(y_j)) \leq \gamma_1/2+\gamma_3+\ell_{\mathrm{Y}}. \end{aligned}$$

Hence, in total we have $L(z_j, \mathrm{G}(y_j)) \leq \gamma_1+\gamma_3+\ell_{\mathrm{Y}} \leq \gamma_1+\gamma_2+3\gamma_3 = \gamma_1+\gamma_2+3\gamma_3+v_j$, as desired.

Consider a $j \in [Q]$ that is uniquely aligned (under Λ) to some i. Then z_j is a subsequence of $0^{\gamma_1/2}H(x_{i-1})0^{\gamma_1}H(x_i)0^{\gamma_1/2}$. Analogously to above we compute

$$\begin{aligned} L(z_j, \mathrm{G}(y_j)) &\leq \frac{\gamma_1}{2} + L(H(x_{i-1})0^{\gamma_1}H(x_i)0^{\gamma_1/2}, 0^{\gamma_1/2}H(y_j)) \\ &= \gamma_1 + L(0^{\#_0(H(x_{i-1}))+\gamma_1}H(x_i)0^{\gamma_1/2}, H(y_j)) \\ &= \gamma_1 + L(0^{\#_0(H(x_{i-1}))+\gamma_1}1^{\gamma_2}(01)^{\gamma_3}x_i1^{\gamma_3}0^{\gamma_1/2}, 1^{\gamma_2}(01)^{\gamma_3}y_j1^{\gamma_3}). \end{aligned}$$

Using Lemma 5.5.6 with symbol 0 replaced by 1 yields, since $\ell := \gamma_2 \geq 3\gamma_3 + \ell_{\mathrm{Y}} = |(01)^{\gamma_3}y_j1^{\gamma_3}|$ and $\#_1(0^{\#_0(H(x_{i-1}))+\gamma_1}) = 0$,

$$\begin{aligned} L(z_j, \mathrm{G}(y_j)) &\leq \gamma_1 + \gamma_2 + L((01)^{\gamma_3}x_i1^{\gamma_3}0^{\gamma_1/2}, (01)^{\gamma_3}y_j1^{\gamma_3}) \\ &= \gamma_1 + \gamma_2 + 2\gamma_3 + L(x_i1^{\gamma_3}0^{\gamma_1/2}, y_j1^{\gamma_3}). \end{aligned}$$

Similarly, using Lemma 5.5.6 with symbol 0 replaced by 1 on the reversed strings yields, since $\ell := \gamma_3 \geq \ell_{\mathrm{Y}} = |y_j|$ and $\#_1(0^{\gamma_1/2}) = 0$,

$$L(x_i1^{\gamma_3}0^{\gamma_1/2}, y_j1^{\gamma_3}) = \gamma_3 + L(x_i, y_j).$$

Hence, we obtain the desired $L(z_j, \mathrm{G}(y_j)) \leq \gamma_1 + \gamma_2 + 3\gamma_3 + L(x_i, y_j) = \gamma_1 + \gamma_2 + 3\gamma_3 + v_j$.

It remains to consider $j \in [Q]$ that is k-aligned for $k \geq 2$. In this case, the claim follows from the trivial bound $L(z_j, \mathrm{G}(y_j)) \leq |\mathrm{G}(y_j)| = \gamma_1 + \gamma_2 + 3\gamma_3 + v_j$.

Thus $z_1, \ldots, z_Q$ defines a multi-alignment $\Lambda \in \mathbf{\Lambda}^{\mathrm{multi}}_{P,Q}$ with

$$L(x, y) = \sum_{j=1}^{Q} L(z_j, \mathrm{G}(y_j)) \leq Q(\gamma_1 + \gamma_2 + 3\gamma_3) + \mathbf{v}(\Lambda),$$

proving the second inequality of (5.2). □

We can now show how to embed an OV instance $\mathcal{A} = \{a_1, \ldots, a_A\}, \mathcal{B} = \{b_1, \ldots, b_B\} \subseteq \{0,1\}^D$ with $A \leq B$ into strings x and y of length $\mathcal{O}(B \cdot D)$ whose LCS can be obtained by deleting at most $\mathcal{O}(A \cdot D)$ symbols from y. For this we will without loss of generality assume that A divides B by possibly duplicating some arbitrary element of $\mathcal{B}$ up to $A-1$ times without affecting the solution of the instance.

The key idea is that for any P and $Q = 2P - N$ with $N \in \{0, \ldots, P\}$, $\mathbf{\Lambda}^{1,2}_{P,Q}$ is non-empty and each $\Lambda \in \mathbf{\Lambda}^{1,2}_{P,Q}$ has exactly N uniquely aligned $j \in [Q]$ and exactly $P - N$ 2-aligned $j \in [Q]$. At the same time each $\Lambda \in \mathbf{\Lambda}^{\mathrm{multi}}_{P,Q}$ leaves at least N indices $j \in [Q]$ either unaligned or uniquely aligned.

Lemma 5.6.7. *Let $a_1, \ldots, a_A, b_1, \ldots b_B \subseteq \{0,1\}^D$ be given with $A \mid B$. Construct the corresponding strings $x_1, \ldots, x_A$ of length ℓ_{X}, $y_1, \ldots, y_B$ of length $\ell_{\mathrm{Y}} \leq \ell_{\mathrm{X}} = \mathcal{O}(D)$, and integers ρ_0, ρ_1 as in Lemma 5.6.5 and define*

$$\begin{aligned} \tilde{x} &:= (\tilde{x}_1, \ldots, \tilde{x}_P) = (\overbrace{x_1, \ldots, x_A, x_1, \ldots, x_A, \ldots, x_1, \ldots, x_A}^{2\cdot(B/A)+3 \text{ groups of size } A}), \\ \tilde{y} &:= (\tilde{y}_1, \ldots, \tilde{y}_Q) = (\underbrace{y_1, \ldots, y_1}_{A \text{ copies of } y_1}, y_1, \ldots, y_B, \underbrace{y_1, \ldots, y_1}_{A \text{ copies of } y_1}), \end{aligned}$$

where $P := 2B + 3A$ and $Q := B + 2A$. Then the instance $x := \bigcirc_i \mathrm{G}(\tilde{x}_i)$, $y := \bigcirc_j \mathrm{G}(\tilde{y}_j)$ of Lemma 5.6.6 (with the corresponding choice of γ_1, γ_2 and γ_3) satisfies the following properties:

(i) For every $i \in [A], j \in [B]$, there is a (1,2)-alignment $\Lambda \in \mathbf{\Lambda}^{1,2}_{P,Q}$ such that some $\ell \in [Q]$ is uniquely aligned to some $k \in [P]$ with $\tilde{x}_k = x_i$ and $\tilde{y}_\ell = y_j$.

(ii) We have $L(x,y) \geq Q(\gamma_1+\gamma_2+3\gamma_3) + (A-1)\rho_1 + \rho_0 + (Q-A)\ell_{\mathrm{Y}}$ if and only if there are $i \in [A], j \in [B]$ with $\langle a_i, b_j\rangle = 0$.

(iii) We have $|y| \leq |x| = \mathcal{O}(B \cdot D)$ and $\delta(x,y) = \mathcal{O}(A \cdot D)$.

Proof. For (i), we let $j \in [B]$ and note that $y_j = \tilde{y}_\ell$ for $\ell := A + j$. We will show that for every $\lambda \in \{0, \ldots, A-1\}$, there is a (1,2)-alignment Λ with $(k,\ell) \in \mathbf{\Lambda}^{1,2}_{P,Q}$ where $k := 2(A+j) - 1 - \lambda$. By the cyclic structure of $\tilde{x}$, $(\tilde{x}_k)_{0\leq\lambda<A}$ cycles through all values $x_1, \ldots, x_A$. Hence, for some choice of λ the desired $\tilde{x}_k = x_i$ follows, yielding the claim.

To see that for any $\lambda \in \{0, \ldots, A-1\}$, some $\Lambda \in \mathbf{\Lambda}^{1,2}_{P,Q}$ with $(k,\ell) \in \Lambda$ exists, observe that there are $\ell - 1 = A + j - 1$ predecessors of $\tilde{y}_\ell$ and $k - 1 = 2(A+j-1) - \lambda = 2(\ell-1) - \lambda$ predecessors of $\tilde{x}_k$. Hence there is a (1,2)-alignment $\Lambda_1 \in \mathbf{\Lambda}^{1,2}_{k-1,\ell-1}$ (leaving λ indices $j \in [Q]$ uniquely aligned). Similarly, observe that there are $Q - \ell = B + A - j$ successors of $\tilde{y}_\ell$ and $P - k = 2B + A - 2j + \lambda + 1 = 2(Q-\ell) - (A - \lambda - 1)$ successors of $\tilde{x}_k$, hence there is a (1,2)-alignment $\Lambda_2 \in \mathbf{\Lambda}^{1,2}_{P-k,Q-\ell}$ (which leaves $A - (\lambda+1)$ indices j uniquely aligned). By canonically composing Λ_1, (k,ℓ) and Λ_2 we can thus obtain $\Lambda \in \mathbf{\Lambda}^{1,2}_{P,Q}$ with $(k,\ell) \in \Lambda$.

For (ii), assume that there are $i \in [A], j \in [B]$ satisfying $\langle a_i, b_j\rangle = 0$. By (i), there is some $\Lambda \in \mathbf{\Lambda}^{1,2}_{P,Q}$ where some $\ell \in [Q]$ is uniquely aligned to some $k \in [P]$ such that $\tilde{x}_k = x_i$ and $\tilde{y}_\ell = y_j$. To apply Lemma 5.6.6, observe that Λ has $Q - A$ 2-aligned $j \in [Q]$, which contribute value ℓ_{Y} to $\mathbf{v}(\Lambda)$, and A uniquely aligned $j \in [Q]$, in particular, ℓ is uniquely aligned to k. Since any $\tilde{x}_i$ corresponds to some $x_{i'}$, every $\tilde{y}_j$ corresponds to some $y_{j'}$ and $L(x_{i'}, y_{j'}) \in \{\rho_0, \rho_1\}$, we conclude that ℓ contributes ρ_0 to $\mathbf{v}(\Lambda)$ and the other $A-1$ uniquely aligned j contribute at least ρ_1. Hence by the lower bound in Lemma 5.6.6, we obtain $L(x,y) \geq Q(\gamma_1+\gamma_2+3\gamma_3) + \mathbf{v}(\Lambda)$, where $\mathbf{v}(\Lambda) \geq (A-1)\rho_1 + \rho_0 + (Q-A)\ell_{\mathrm{Y}}$.

Assume now that no $i \in [A], j \in [B]$ satisfy $\langle a_i, b_j\rangle = 0$, and let $\Lambda \in \mathbf{\Lambda}^{\mathrm{multi}}_{P,Q}$. Then any $j \in [Q]$ uniquely aligned to some $i \in [P]$ contributes $L(\tilde{x}_i, \tilde{y}_j) = \rho_1$ to $\mathbf{v}(\Lambda)$. Let λ be the number of $j \in [Q]$ that are k-aligned for any $k \geq 2$, each contributing ℓ_{Y} to $\mathbf{v}(\Lambda)$. Then there are at most $\min\{P - 2\lambda, Q - \lambda\}$ uniquely aligned $j \in [Q]$ (since every k-aligned j blocks at least two $i \in [P]$ for other alignments), and the remaining $j \in [Q]$ are unaligned, with no contribution to $\mathbf{v}(\Lambda)$. Hence $\mathbf{v}(\Lambda) \leq \lambda\ell_{\mathrm{Y}} + \min\{P-2\lambda, Q-\lambda\} \cdot \rho_1 = \min\{P\rho_1 + (\ell_{\mathrm{Y}} - 2\rho_1)\lambda, Q\rho_1 + (\ell_{\mathrm{Y}} - \rho_1)\lambda\}$. Note that $\ell_{\mathrm{Y}}/2 < \rho_1 \leq \ell_{\mathrm{Y}}$ (by Lemma 5.6.5(iv)), hence this minimum of linear functions with leading coefficients $\ell_{\mathrm{Y}} - 2\rho_1 < 0$ and $\ell_{\mathrm{Y}} - \rho_1 \geq 0$ is maximized when both have the same value, i.e., when $\lambda = P - Q = Q - A$. Thus, $\mathbf{v}(\Lambda) \leq (Q-A)\ell_{\mathrm{Y}} + A\rho_1 < (Q-A)\ell_{\mathrm{Y}} + (A-1)\rho_1 + \rho_0$. Thus by the upper bound of Lemma 5.6.6 we conclude that $L(x,y) < Q(\gamma_1+\gamma_2+3\gamma_3) + (Q-A)\ell_{\mathrm{Y}} + (A-1)\rho_1 + \rho_0$.

For (iii), since $P \geq Q$ and $\ell_{\mathrm{X}} \geq \ell_{\mathrm{Y}}$ we have $|x| \geq |y|$, and by $P = \mathcal{O}(A)$ and $|G(\tilde{x}_i)| = \mathcal{O}(\ell_{\mathrm{X}}) = \mathcal{O}(D)$ we obtain $|x| = \mathcal{O}(AD)$. Note that for any (1,2)-alignment $\Lambda \in \mathbf{\Lambda}^{1,2}_{P,Q}$, we have

$$\mathbf{v}(\Lambda) = Q \cdot \ell_{\mathrm{Y}} - \sum_{j \text{ uniquely aligned to } i} (\ell_{\mathrm{Y}} - L(x_i, y_j)) = Q \cdot \ell_{\mathrm{Y}} - \mathcal{O}(A \cdot D),$$

since by $P = 2Q - A$ the number of uniquely aligned indices j in Λ equals A, and $\ell_{\mathrm{Y}} = \mathcal{O}(D)$. Hence by Lemma 5.6.6, $L(x,y) \geq Q(\gamma_1+\gamma_2+3\gamma_3) + Q\ell_{\mathrm{Y}} - \mathcal{O}(A \cdot D) = |y| - \mathcal{O}(A \cdot D)$, implying $\delta(x,y) = |y| - L(x,y) = \mathcal{O}(A \cdot D)$. □

Constant Alphabet

First assume $\alpha_\Sigma = 0$ and thus $|\Sigma| = \mathcal{O}(1)$. Consider any $n \geq 1$ and target values $p = \lceil n^{\alpha_p} \rceil$ for $p \in \mathcal{P}$. We write $\lfloor x \rfloor_2$ for the largest power of 2 less than or equal to x. Let $\mathcal{A} = \{a_1, \ldots, a_A\}$, $\mathcal{B} = \{b_1, \ldots, b_B\} \subseteq \{0,1\}^D$ be a given OV instance with $D = n^{o(1)}$ and where we set

$$A := \left\lfloor \frac{1}{D} \min\left\{\delta, \frac{d}{\min\{m,\Delta\}}\right\} \right\rfloor_2 \quad \text{and} \quad B := \left\lfloor \frac{1}{D} \min\{m,\Delta\} \right\rfloor_2.$$

By $\alpha_m, \alpha_\Delta \leq 1$ and (LB) we obtain that $A \geq n^{\min\{\alpha_L, \alpha_d - 1\} - o(1)} = n^{\Omega(1)}$ as well as $B = n^{\min\{\alpha_m, \alpha_\Delta\} - o(1)} = n^{\Omega(1)}$. Also note that UOVH implies that solving such OV instances takes time $(AB)^{1-o(1)} = \min\{d, \delta m, \delta\Delta\}^{1-o(1)}$, which is the desired bound. We claim that $A \leq B$, implying $A \mid B$. Indeed, if $\delta \leq d/\min\{m,\Delta\}$, this follows from the simple parameter relations $\delta \leq m$ and $\delta \leq \Delta$. Otherwise, if $\delta > d/\min\{m,\Delta\}$, then in particular $\delta\Delta > d$, implying $d < \Delta^2$. Together with the parameter relations $d \leq Lm \leq m^2$ we indeed obtain $d/\min\{m,\Delta\} \leq \min\{m,\Delta\}$.

Thus, we may construct strings x, y as in Lemma 5.6.7. We finish the construction by invoking the Dominant Pair Reduction (Lemma 5.5.8(ii)) to obtain strings $x' := 0^k 1^k y 1^\ell 0^k 1^k x$ and $y' := 1^\ell 0^k 1^k y$ with $k := 2|y| + |x| + 1$ and $\ell := \beta \cdot \lceil AD \rceil$ with sufficiently large constant β, so that $\ell > \delta(x,y)$. Then from the LCS length $L(x',y')$ we can infer whether $\mathcal{A}, \mathcal{B}$ has an orthogonal pair of vectors by $L(x',y') = L(x,y) + \ell + 2k$ and Lemma 5.6.7(ii). Moreover, this reduction runs in time $\mathcal{O}(|x'| + |y'|) = \mathcal{O}(|x| + |y|) = \mathcal{O}(BD) = \mathcal{O}(\min\{d, \delta m, \delta\Delta\}^{1-\varepsilon})$ for sufficiently small $\varepsilon > 0$ (since $\alpha_\delta > 0$ and $\alpha_d > 1 \geq \alpha_m, \alpha_\Delta$ by (LB) and Table 5.2). We claim that (n, x', y') is an instance of $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$. This shows that any algorithm solving $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$ in time $\mathcal{O}(\min\{d, \delta m, \delta\Delta\}^{1-\varepsilon})$ implies an algorithm for our OV instances with running time $\mathcal{O}(\min\{d, \delta m, \delta\Delta\}^{1-\varepsilon})$, contradicting UOVH. Hence, in the current case $\alpha_L = \alpha_m$ and $\alpha_\Sigma = 0$, any algorithm for $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$ takes time $\min\{d, \delta m, \delta\Delta\}^{1-o(1)}$, proving part of Theorem 5.3.3.

It remains to show the claim that (n, x', y') is an instance of $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$. From Lemmas 5.5.8 and 5.6.7(iii) we obtain $L(x',y') = \ell + 2k + L(x,y) = \ell + 2k + |y| - \delta(x,y) = |y'| - \mathcal{O}(AD)$, and thus $\delta(x',y') = \mathcal{O}(AD) = \mathcal{O}(\delta)$. Using the parameter relations $L \leq m \leq n$, Lemma 5.6.7(iii), and the definition of B, we have $L(x',y') \leq m(x',y') \leq n(x',y') = |x'| = \mathcal{O}(BD) = \mathcal{O}(\min\{m,\Delta\})$, which together with the relation $m \leq n$ and the assumption $\alpha_L = \alpha_m$ shows that $p(x',y') = \mathcal{O}(p) = \mathcal{O}(n^{\alpha_p})$ for $p \in \{L, m, n\}$. Similarly, we obtain $\Delta(x',y') \leq n(x',y') \leq \mathcal{O}(\min\{m,\Delta\}) \leq \mathcal{O}(\Delta)$. Since x', y' use the binary alphabet $\{0,1\}$, we have $|\Sigma(x',y')| = 2 = \mathcal{O}(n^{\alpha_\Sigma})$. For the number of matching pairs we have $M(x',y') \leq n(x',y')^2 = \mathcal{O}((BD)^2) = \mathcal{O}(L^2)$. Since we are in the case $\alpha_\Sigma = 0$, from the parameter relation $M \geq L^2/|\Sigma|$ (Lemma 5.4.6(i)) we obtain $L^2 = \mathcal{O}(M)$ and thus also $M(x',y')$ is sufficiently small. Finally, we use Lemma 5.5.8 to bound $d(x',y') = \mathcal{O}(\ell \cdot |y|) = \mathcal{O}(AD \cdot BD)$, which by definition of A, B is $\mathcal{O}(d)$. This proves that (n, x', y') belongs to $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$.

We remark that our proof also yields the following, which we will use for small alphabets in Section 5.8.

Lemma 5.6.8. *Let $\boldsymbol{\alpha}$ be a parameter setting satisfying Table 5.2 with $\alpha_\Sigma = 0$ and $\alpha_L = \alpha_m$. There is a constant $\gamma \geq 1$ such that any algorithm for $\mathrm{LCS}^{\gamma}_{\leq}(\boldsymbol{\alpha}, \{0,1\})$ takes time $\min\{d, \delta m, \delta\Delta\}^{1-o(1)}$, unless* OVH *fails. This holds even restricted to instances (n, x, y) of $\mathrm{LCS}^{\gamma}_{\leq}(\boldsymbol{\alpha}, \{0,1\})$ with $|y| \leq |x| \leq \gamma \cdot \min\{n^{\alpha_m}, n^{\alpha_\Delta}\}$.*

Superconstant Alphabet

The crucial step in extending our construction to larger alphabets is to adapt the 1vs1/2vs1 gadget such that the strings use each symbol in the alphabet Σ roughly evenly, thus reducing the number of matching pairs by a factor $|\Sigma|$.

Recall that given a 2-element alphabet Σ' and a string z over $\{0,1\}$, we let $z{\uparrow}\Sigma'$ denote the string z lifted to alphabet Σ' by bijectively replacing $\{0,1\}$ with Σ'.

Lemma 5.6.9. *Let $P = 2B + 3A$ and $Q = B + 2A$ for some $A \mid B$. Given strings $x_1, \ldots, x_P$ of length ℓ_{X} and $y_1, \ldots, y_Q$ of length ℓ_{Y}, we define, as in Lemma 5.6.6, $\mathrm{G}(w) := 0^{\gamma_1}\, 1^{\gamma_2}\, (01)^{\gamma_3}\, w\, 1^{\gamma_3}$ with $\gamma_3 := \ell_{\mathrm{X}} + \ell_{\mathrm{Y}}$, $\gamma_2 := 8\gamma_3$ and $\gamma_1 := 6\gamma_2$. Let $\Sigma_1, \ldots, \Sigma_t$ be disjoint alphabets of size 2 with $Q/t \geq A/2 + 1$. We define*

$$\begin{aligned} x &:= H(x_1) \quad H(x_2) \quad \ldots \quad H(x_P), \\ y &:= \mathrm{G}(y_1){\uparrow}\Sigma_{f(1)} \quad \mathrm{G}(y_2){\uparrow}\Sigma_{f(2)} \quad \ldots \quad \mathrm{G}(y_Q){\uparrow}\Sigma_{f(Q)}, \end{aligned}$$

where $f(j) = \lceil \frac{j}{Q} \cdot t \rceil$ and

$$H(x_i) := \begin{cases} \mathrm{G}(x_i){\uparrow}\Sigma_{k+1}\, \mathrm{G}(x_i){\uparrow}\Sigma_k & \text{if } \bigcup_{j=\lceil i/2 \rceil}^{\lfloor (i+A)/2 \rfloor} \{f(j)\} = \{k, k+1\} \\ \mathrm{G}(x_i){\uparrow}\Sigma_k & \text{if } \bigcup_{j=\lceil i/2 \rceil}^{\lfloor (i+A)/2 \rfloor} \{f(j)\} = \{k\} \end{cases}$$

Then we have

$$\max_{\Lambda \in \mathbf{\Lambda}^{1,2}_{P,Q}} \mathbf{v}(\Lambda) \leq L(x,y) - Q(\gamma_1 + \gamma_2 + 3\gamma_3) \leq \max_{\Lambda \in \mathbf{\Lambda}^{\mathrm{multi}}_{P,Q}} \mathbf{v}(\Lambda). \tag{5.3}$$

Proof. Note that $H(\cdot)$ is well defined, since $f(\cdot)$ maps $\{1, \ldots, Q\}$ to constant-valued intervals of length at least $Q/t - 1 \geq A/2$, as $f(j) = k$ if and only if $j \in \big(\frac{Qk}{t} - \frac{Q}{t}, \frac{Qk}{t}\big]$, containing at least $Q/t - 1$ integers. Hence for every i, the at most $A/2$ values $f(\lceil i/2 \rceil), \ldots, f(\lfloor (i+A)/2 \rfloor)$ can touch at most 2 different constant-valued intervals.

The proof of (5.3) is based on the proof of Lemma 5.6.6 (the analogous lemma for alphabet $\Sigma = \{0,1\}$). For the first inequality of (5.3), let $\Lambda \in \mathbf{\Lambda}^{1,2}_{P,Q}$ and define for every j the substring $z'_j = \bigcirc_{i \in \Lambda(j)} H(x_i)$. Note that under Λ, each $\Lambda(j)$ consists of one or two elements from $\{2j - A, \ldots, 2j\}$, since there are at most $2Q - P = A$ uniquely aligned j. In other words, for any $i \in \Lambda(j)$ we have $j \in \{\lceil i/2 \rceil, \ldots, \lfloor (i+A)/2 \rfloor\}$. Thus, by definition each $H(x_i)$ for $i \in \Lambda(j)$ contains $\mathrm{G}(x_i){\uparrow}\Sigma_{f(j)}$ as a substring and hence z'_j contains $\bigcirc_{i \in \Lambda(j)} \mathrm{G}(x_i){\uparrow}\Sigma_{f(j)}$ as a subsequence. This proves

$$L\big(z'_j, \mathrm{G}(y_j){\uparrow}\Sigma_{f(j)}\big) \geq L\Big(\bigcirc_{i \in \Lambda(j)} \mathrm{G}(x_i){\uparrow}\Sigma_{f(j)}, \mathrm{G}(y_j){\uparrow}\Sigma_{f(j)}\Big) = L\Big(\bigcirc_{i \in \Lambda(j)} \mathrm{G}(x_i), \mathrm{G}(y_j)\Big),$$

which reduces the proof to the case of $\Sigma = \{0,1\}$ – note that the last term is equal to $L(z_j, \mathrm{G}(y_j))$ in the proof of the same inequality of Lemma 5.6.6 and thus the remainder follows verbatim.

It remains to show the second inequality of (5.3). Essentially as in the proof of Lemma 5.6.6, we write $x = z'_1 z'_2 \ldots z'_Q$ with $L(x,y) = \sum_{j=1}^{Q} L(z'_j, \mathrm{G}(y_j){\uparrow}\Sigma_{f(j)})$. For every z'_j, we obtain a string z_j by deleting all symbols not contained in $\Sigma_{f(j)}$ and then lifting it to the alphabet $\{0,1\}$. We conclude that $L(z'_j, \mathrm{G}(y_j){\uparrow}\Sigma_{f(j)}) = L(z_j, \mathrm{G}(y_j))$. We claim that $z := z_1 z_2 \ldots z_Q$ is a subsequence of $x_{\{0,1\}} := \mathrm{G}(x_1) \ldots \mathrm{G}(x_P)$ (which is equal to the string x that we constructed in the case of $\Sigma = \{0,1\}$). Indeed, if $H(x_i)$ is of the form $w_{k+1} w_k$ for some k with $w_\ell = \mathrm{G}(x_i){\uparrow}\Sigma_\ell$, then symbols of at most one of w_k

and w_{k+1} are contained in z. To see this, note that if w_k is not deleted then at least one of its symbols is contained in some z'_j with $f(j) = k$, but then no symbol in w_{k+1} can be contained in $z'_{j'}$ with $f(j') = k+1$, since this would mean $j' > j$, so w_{k+1} is deleted. Thus,

$$L(x,y) = \sum_{j=1}^{Q} L\big(z'_j, \mathrm{G}(y_j) \upharpoonright \Sigma_{f(j)}\big) = \sum_{j=1}^{Q} L\big(z_j, \mathrm{G}(y_j)\big) \leq L(x_{\{0,1\}}, y_{\{0,1\}}),$$

where $y_{\{0,1\}} := \mathrm{G}(y_1)\dots\mathrm{G}(y_Q)$ is the string y that we constructed in the case of $\Sigma = \{0,1\}$. Hence, the second inequality of (5.3) follows from the proof of Lemma 5.6.6. □

By the same choice of vectors as in Lemma 5.6.7, we can embed orthogonal vectors instances.

Lemma 5.6.10. *Let $a_1, \dots, a_A, b_1, \dots b_B \subseteq \{0,1\}^D$ be given with $A \mid B$. Construct the corresponding strings $x_1, \dots, x_A$ of length ℓ_{X}, $y_1, \dots, y_B$ of length $\ell_{\mathrm{Y}} \leq \ell_{\mathrm{X}} = \mathcal{O}(D)$ and integers ρ_0, ρ_1 as in Lemma 5.6.5 and define*

$$\begin{aligned} \tilde{x} &:= (\tilde{x}_1, \dots, \tilde{x}_P) = (\overbrace{x_1, \dots, x_A, x_1, \dots, x_A, \dots, x_1, \dots, x_A}^{2\cdot(B/A)+3 \text{ groups of size } A}), \\ \tilde{y} &:= (\tilde{y}_1, \dots, \tilde{y}_Q) = (\underbrace{y_1, \dots, y_1}_{A \text{ copies of } y_1}, y_1, \dots, y_B, \underbrace{y_1, \dots, y_1}_{A \text{ copies of } y_1}), \end{aligned}$$

where $P := 2B + 3A$ and $Q := B + 2A$. For disjoint alphabets $\Sigma_1, \dots, \Sigma_t$ of size 2 with $Q/t \geq A/2 + 1$, we construct the instance $x := \bigcirc_i H(\tilde{x}_i)$, $y := \bigcirc_j \mathrm{G}(\tilde{y}_j)$ of Lemma 5.6.6 (with the corresponding choice of γ_1, γ_2 and γ_3). This satisfies the following properties:

(i) We have that $L(x,y) \geq Q(\gamma_1 + \gamma_2 + 3\gamma_3) + (A-1)\rho_1 + \rho_0 + (Q-A)\ell_{\mathrm{Y}}$ if and only if there are $i \in [A], j \in [B]$ with $\langle a_i, b_j \rangle = 0$.

(ii) We have $|y| \leq |x| = \mathcal{O}(B \cdot D)$ and $\delta(x,y) = \mathcal{O}(A \cdot D)$.

Proof. The lemma and its proof are a slight adaptation of Lemma 5.6.7: For (i), since Lemma 5.6.9 proves (5.3) which is identical to (5.2), we can follow the proof of Lemma 5.6.7(i) and (ii) verbatim (since we have chosen $\tilde{x}$ and $\tilde{y}$ as in this lemma). For (ii), the bounds $|y| \leq |x| \leq \mathcal{O}(B \cdot D)$ and $\delta(x,y) = \mathcal{O}(A \cdot D)$ follow exactly as in Lemma 5.6.6 (note that only $|x|$ has increased by at most a factor of 2, so that $|x| = \mathcal{O}(B \cdot D)$ still holds by the trivial bound). □

We can now finalize the proof of Theorem 5.3.3. Consider any $n \geq 1$ and target values $p = \lceil n^{\alpha_p} \rceil$ for $p \in \mathcal{P}$. Let $\mathcal{A} = \{a_1, \dots, a_A\}$, $\mathcal{B} = \{b_1, \dots, b_B\} \subseteq \{0,1\}^D$ be a given OV instance with $D = n^{o(1)}$ and where we set, as in the case $\alpha_\Sigma = 0$,

$$A := \Big\lfloor \frac{1}{D} \min\Big\{\delta, \frac{d}{\min\{m, \Delta\}}\Big\} \Big\rfloor_2 \quad \text{and} \quad B := \Big\lfloor \frac{1}{D} \min\{m, \Delta\} \Big\rfloor_2.$$

As before, we have $A \mid B$, so we may construct strings x, y as in Lemma 5.6.10, where we set $t := \min\{\lfloor Q/(A/2+1) \rfloor, |\Sigma|\} = \Theta(\min\{B/A, |\Sigma|\})$. We finish the construction by invoking the Dominant Pair Reduction (Lemma 5.5.8(i)) to obtain strings $x' := y2^\ell x$ and $y' := 2^\ell y$, where 2 is a symbol not appearing in x, y and we set $\ell := \beta \cdot \lceil AD \rceil$ with sufficiently large constant β, so that $\ell > \delta(x,y)$.

For the remainder of the proof, we can follow the case $\alpha_\Sigma = 0$ almost verbatim. The only exception is the bound on the number of matching pairs. Note that symbol 2

appears $\mathcal{O}(AD)$ times in x' and y'. As in x and y every symbol appears roughly equally often and the total alphabet size is $\Theta(t)$, for any symbol $\sigma \neq 2$ we have $\#_\sigma(x) = \mathcal{O}(|x|/t)$ and $\#_\sigma(y) = \mathcal{O}(|y|/t)$, implying $\#_\sigma(x'), \#_\sigma(y') = \mathcal{O}(BD/t)$. Hence, $M(x', y') = \mathcal{O}((AD)^2 + t \cdot (BD/t)^2)$. Using $t = \Theta(\min\{B/A, |\Sigma|\})$ and $A \leq B$, we obtain $M(x', y') = \mathcal{O}(\max\{AD \cdot BD, (BD)^2/|\Sigma|\}) = \mathcal{O}(\max\{d, m^2/|\Sigma|\}$. The assumption $\alpha_L = \alpha_m$ and the parameter relations $M \geq L^2/|\Sigma|$ and $M \geq d$ now imply $M(x', y') = \mathcal{O}(M)$. This concludes the proof of Theorem 5.3.3.

5.7 Paddings

In this section, we construct paddings that allow us to augment any strings from $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$ to become strings in $\mathrm{LCS}(\boldsymbol{\alpha})$. Specifically, we prove Lemma 5.3.6. To this end, let $\boldsymbol{\alpha}$ be a parameter setting satisfying Table 5.2, let $p \in \mathcal{P}^* = \{n, m, L, \delta, \Delta, |\Sigma|, M, d\}$ be a parameter, and let $n \geq 1$. We say that strings x, y *prove Lemma 5.3.6 for parameter* p if (n, x, y) is an instance of $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$ with $p(x, y) = \Theta(n^{\alpha_p})$, and given n we can compute $x = x(n)$, $y = y(n)$, and $L(x, y)$ in time $\mathcal{O}(n)$. Note that for the first requirement of being an instance of $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha})$, we have to show that $p'(x, y) = \mathcal{O}(n^{\alpha_{p'}})$ for any parameter $p' \in \mathcal{P}^*$. Recall that we write $p = \lceil n^{\alpha_p} \rceil$ for the target value of parameter p.

Lemma 5.7.1. *Let Σ' be an alphabet of size $\min\{|\Sigma|, L\}$. Then the strings $x := y := \bigcirc_{\sigma \in \Sigma'} \sigma^{\lfloor L/|\Sigma'| \rfloor}$ prove Lemma 5.3.6 for parameter L.*

Proof. Note that $\lfloor L/|\Sigma'| \rfloor = \Theta(L/|\Sigma'|)$, since $|\Sigma'| \leq L$. Thus, indeed $L(x, y) = |x| = \Theta(L/|\Sigma'|) \cdot |\Sigma'| = \Theta(L)$. Moreover, $L(x, y)$ can be computed in time $\mathcal{O}(n)$, as well as x and y. For the number of matching pairs we note that $M(x, y) \leq |\Sigma'| \cdot (L/|\Sigma'|)^2$, which is $\max\{L, L^2/|\Sigma|\}$ by choice of $|\Sigma'|$. This is $\mathcal{O}(M)$, using the parameter relations $M \geq n \geq m \geq L$ and $M \geq L^2/|\Sigma|$ (see Table 5.3).

The remaining parameters are straight-forward. Using $m(x, y) = n(x, y) = L(x, y) = \Theta(L)$ and the parameter relations $L \leq m \leq n$ we obtain that $m(x, y) = \mathcal{O}(m)$ and $n(x, y) = \mathcal{O}(n)$. Moreover, $\delta(x, y) = \Delta(x, y) = 0 \leq \delta \leq \Delta$. The alphabet size $|\Sigma(x, y)| = |\Sigma'|$ is at most $|\Sigma|$ by choice of $|\Sigma'|$. By Lemma 5.5.1 we obtain $d(x, y) = |x| = \Theta(L) = \mathcal{O}(d)$ using the parameter relation $L \leq d$. □

Lemma 5.7.2. *The strings $x := 1^{\Delta+1}$ and $y := 1$ prove Lemma 5.3.6 for parameter Δ. The strings $x := 1$ and $y := 1^{\delta+1}$ prove Lemma 5.3.6 for parameter δ.*

Proof. The analysis is straight-forward. Note that indeed $\Delta(x, y) = \Delta$, and that $L(x, y) = 1 = \mathcal{O}(L)$. We have $n(x, y) = \Delta + 1 = \mathcal{O}(n)$ and $m(x, y) = 1 = \mathcal{O}(m)$. Clearly, $L(x, y)$, x, and y can be computed in time $\mathcal{O}(n)$. Moreover, $\delta(x, y) = 0 \leq \delta$ and the alphabet size is $1 = \mathcal{O}(\Sigma)$. Finally, we have $M(x, y) = \Theta(\Delta) = \mathcal{O}(n) = \mathcal{O}(M)$ using the parameter relations $L \leq n \leq M$, and using the relation $d \leq Lm$ we obtain $d(x, y) \leq 1 = \mathcal{O}(d)$.

The analysis for δ is symmetric; the same proof holds almost verbatim. □

Lemma 5.7.3. *The strings constructed in Lemma 5.7.1 or the strings constructed in Lemma 5.7.2 prove Lemma 5.3.6 for parameters n and m.*

Proof. Since $n = L + \Delta$, we have $L = \Theta(n)$ or $\Delta = \Theta(n)$, i.e., $\alpha_L = 1$ or $\alpha_\Delta = 1$. In the first case, in Lemma 5.7.1 we construct strings of length $\Theta(L) = \Theta(n)$, and thus these strings prove Lemma 5.3.6 not only for parameter L but also for parameter n. In the second case, the same argument holds for the first pair of strings constructed in Lemma 5.7.2. The parameter m is symmetric. □

Lemma 5.7.4. *Let $w := 12\dots|\Sigma|$ be the concatenation of $|\Sigma|$ unique symbols. The strings w, w or the strings $w, \mathrm{rev}(w)$ prove Lemma 5.3.6 for parameter $|\Sigma|$.*

Proof. Clearly, both pairs of strings realize an alphabet of size exactly $|\Sigma|$. Since $m = L+\delta$, we have $L = \Theta(m)$ or $\delta = \Theta(m)$. In the first case, we use $L(w, w) = |\Sigma| \leq m = \Theta(L)$ and $\delta(w, w) = \Delta(w, w) = 0 \leq \delta \leq \Delta$. In the second case, we have $L(w, \mathrm{rev}(w)) = 1 = \mathcal{O}(L)$ and $\delta(w, \mathrm{rev}(w)) = \Delta(w, \mathrm{rev}(w)) = |\Sigma| - 1 \leq m = \Theta(\delta) = \mathcal{O}(\Delta)$.

The remaining parameters are straight-forward. Let $(x, y) \in \{(w, w), (w, \mathrm{rev}(w))\}$. We have $n(x, y) = m(x, y) = |\Sigma| \leq m \leq n$. Moreover, $d(x, y) \leq M(x, y) = |\Sigma| \leq d \leq M$ using the relations $|\Sigma| \leq d \leq M$. Clearly, the strings and their LCS length can be computed in time $\mathcal{O}(n)$. □

5.7.1 Matching Pairs

Lemma 5.7.5. *If $\alpha_\Delta = 1$ then $x := 1^{\lfloor M/n \rfloor + \Delta}$ and $y := 1^{\lfloor M/n \rfloor}$ prove Lemma 5.3.6 for parameter M.*

Proof. Note that $\lfloor M/n \rfloor = \Theta(M/n)$ by the parameter relation $M \geq n$. We have $M(x, y) = \Theta((M/n)^2 + \Delta M/n)$. By the parameter relations $M \leq 2Ln \leq 2n^2$ the first summand is $\mathcal{O}(n \cdot M/n) = \mathcal{O}(M)$. Since $\alpha_\Delta = 1$, the second summand is $\Theta(M)$. Thus, we indeed have $M(x, y) = \Theta(M)$.

The remainder is straight-forward. Clearly, x, y, and $L(x, y) = \lfloor M/n \rfloor$ can be computed in time $\mathcal{O}(n)$. Since $M \leq 2Ln$, we also obtain $L(x, y) = m(x, y) = \lfloor M/n \rfloor = \mathcal{O}(L) = \mathcal{O}(m)$. Moreover, $n(x, y) = \lfloor M/n \rfloor + \Delta = \mathcal{O}(n)$ by the relations $M/n \leq 2L \leq 2n$ and $\Delta \leq n$. Note that $\Delta(x, y) = \Delta$ and $\delta(x, y) = 0 \leq \delta$. The alphabet size is 1. By Lemma 5.5.1, we have $d(x, y) = \lfloor M/n \rfloor \leq 2L \leq 2d$. □

Lemma 5.7.6. *Assume $\alpha_\Delta < 1$ and let Σ' be an alphabet of size $\min\{\lceil m^2/M \rceil, |\Sigma|\}$. Then $x := y := \bigcirc_{\sigma \in \Sigma'} \sigma^{\lfloor m/|\Sigma'| \rfloor}$ prove Lemma 5.3.6 for parameter M.*

Proof. Observe that $\alpha_\Delta < 1$ implies $\alpha_L = \alpha_m = 1$, so that $n = \Theta(L) = \Theta(m)$ (see Table 5.2). The number of matching pairs is $M(x, y) = |\Sigma'| \cdot \lfloor m/|\Sigma'| \rfloor^2$. By the parameter relation $m \geq |\Sigma|$ and $|\Sigma| \geq |\Sigma'|$, we have $\lfloor m/|\Sigma'| \rfloor = \Theta(m/|\Sigma'|)$, and by $M \leq 2Ln = \Theta(m^2)$ we obtain $\lceil m^2/M \rceil = \Theta(m^2/M)$. Thus, $M(x, y) = \Theta(m^2/|\Sigma'|) = \Theta(\max\{M, m^2/|\Sigma|\})$ by choice of $|\Sigma'|$. Using $m = \Theta(L)$ and the parameter relation $M \geq L^2/|\Sigma|$, we indeed obtain $M(x, y) = \Theta(M)$.

The remainder is straight-forward. Since $x = y$ we have $L(x, y) = m(x, y) = n(x, y) = |\Sigma'| \cdot \lfloor m/|\Sigma'| \rfloor \leq m = \Theta(L) = \Theta(n)$. Moreover, $\delta(x, y) = \Delta(x, y) = 0 \leq \delta \leq \Delta$. The alphabet size is $|\Sigma(x, y)| = |\Sigma'| \leq |\Sigma|$ by choice of $|\Sigma'|$. By Lemma 5.5.1, we have $d(x, y) = L(x, y) = \mathcal{O}(L) = \mathcal{O}(d)$. Clearly, x, y and $L(x, y)$ can be computed in time $\mathcal{O}(n)$. □

5.7.2 Dominant Pairs

For the dominant pairs, we start with a simple construction that always works on constant-sized alphabets ($\alpha_\Sigma = 0$).

Lemma 5.7.7. *Assume $\alpha_d \leq 2\alpha_L \leq \alpha_M$ and set $x := (01)^{R+S}$ and $y := 0^R(01)^{R+S}$ (as analyzed in Lemma 5.5.3), instantiated with $R = \lfloor \min\{\Delta, \sqrt{d}\} \rfloor$, $S = \lceil d/R \rceil$. Then x, y prove Lemma 5.3.6 for parameter d.*

Proof. Note that by definition $R \leq \sqrt{d}$, and hence $S \geq d/R \geq R$. By Lemma 5.5.3(ii), we obtain $d(x,y) = \Theta(R \cdot S) = \Theta(R \cdot d/R) = \Theta(d)$. For the other parameters, note that $n(x,y) = 2(R+S) = \mathcal{O}(d/R) = \mathcal{O}(d/\Delta + \sqrt{d})$. By the relation $d \leq 2L(\Delta+1)$, we have $d/\Delta = \mathcal{O}(L)$, and by the assumption $\alpha_d \leq 2\alpha_L$, we have $d = \mathcal{O}(L^2)$ and hence $\sqrt{d} = \mathcal{O}(L)$. Thus, $n(x,y) = \mathcal{O}(L)$.

The remainder is straight-forward. By $L(x,y) \leq m(x,y) \leq n(x,y) = \mathcal{O}(L) = \mathcal{O}(m) = \mathcal{O}(n)$ we have verified n, m, L. Consequently, also $M(x,y) \leq n(x,y)^2 = \mathcal{O}(L^2) = \mathcal{O}(M)$ by the assumption $2\alpha_L \leq \alpha_M$. Trivially, $|\Sigma(x,y)| = 2 = \mathcal{O}(|\Sigma|)$. Observing that $\delta(x,y) = 0 = \mathcal{O}(\delta)$ and $\Delta(x,y) = R = \mathcal{O}(\Delta)$ by Lemma 5.5.3(i) concludes the parameter verification. Since x, y and $L(x,y) = R + 2S$ (by Lemma 5.5.3(i)) can be computed in time $\mathcal{O}(n)$, the claim follows. □

The construction above creates a long LCS of length $L(x,y) = \Theta(m(x,y))$ which forces $d(x,y) = \mathcal{O}(L(x,y)^2)$. With super-constant alphabet sizes, one can construct larger numbers of dominant pairs (compared to $L(x,y)$) by exploiting the crossing gadgets defined in Definition 5.6.4.

Lemma 5.7.8. *Assume $\alpha_d > 2\alpha_L$ and set $v := (01)^{R+S}$ and $w := 0^R(01)^S$ with $R = S = L$. Construct $x := \mathrm{cr}^{\mathrm{X}}(v, \ldots, v)$ and $y := \mathrm{cr}^{\mathrm{Y}}(w, \ldots, w)$ on $\lfloor d/L^2 \rfloor$ copies of v and w. Then x, y prove Lemma 5.3.6 for parameter d.*

Proof. Note that $\lfloor d/L^2 \rfloor = \Theta(d/L^2)$ since we assume $\alpha_d > 2\alpha_L$. By the Crossing Alphabets Lemma (Lemma 5.6.3), we obtain $L(x,y) = L(v,w) = 3L$, and in particular $L(x,y)$, x, and y can be computed in time $\mathcal{O}(n)$.

Furthermore, the Crossing Alphabets Lemma also yields $d(x,y) = \Theta(d/L^2) \cdot d(v,w) = \Theta(d)$, where the bound $d(v,w) = \Theta(L^2)$ follows from Lemma 5.5.3(i). Similarly, we observe that $\Delta(x,y) \leq n(x,y) = \Theta(d/L^2) \cdot n(v,w) = \Theta(d/L)$, which is at most $\mathcal{O}(\Delta) = \mathcal{O}(n)$ by the parameter relation $d \leq 2L(\Delta+1)$. Likewise, $m(x,y) \leq n(x,y) = \mathcal{O}(d/L) = \mathcal{O}(m)$ by the parameter relation $d \leq Lm$. Moreover, $M(x,y) = \mathcal{O}(d/L^2) \cdot M(v,w) = \mathcal{O}(d) = \mathcal{O}(M)$ by $d \leq M$. Finally, the assumption $2\alpha_L < \alpha_d$ together with the parameter relation $d \leq Lm$, i.e., $\alpha_d \leq \alpha_L + \alpha_m$, forces $\alpha_L < \alpha_m$. Hence, $\alpha_\delta = \alpha_m$, i.e., $\delta = \Theta(m)$ (see Table 5.2), and thus $\delta(x,y) \leq m(x,y) = \mathcal{O}(m) = \mathcal{O}(\delta)$. Since v and w have alphabet size 2 and we use $\lfloor d/L^2 \rfloor$ copies over disjoint alphabets, we have $|\Sigma(x,y)| = 2\lfloor d/L^2 \rfloor = \mathcal{O}(|\Sigma|)$ by the parameter relation $d \leq L^2|\Sigma|$, which concludes the proof. □

For super-constant alphabet sizes, the number of matching pairs $M(x,y)$ can attain values much smaller than $L(x,y)^2$, which is an orthogonal situation to the lemma above. In this case, we use a different generalization of the first construction (that we already prepared in Section 5.5).

Lemma 5.7.9. *Assume $\alpha_M < 2\alpha_L$ and set $x := (1 \ldots t\ t' \ldots 1)^R(1 \ldots t)^{S-R}$ and $y := (1 \ldots t)^S$ (as analyzed in Lemma 5.5.5) instantiated with*

$$t := \left\lfloor \frac{L^2}{M} \right\rfloor, \qquad t' := \min\{r, t\} \qquad R := \left\lceil \frac{r}{t} \right\rceil \qquad S := 4\left\lceil \frac{d}{rt} \right\rceil,$$

where $r := \min\{\Delta, \lfloor \sqrt{d/t} \rfloor\}$. Then x, y prove Lemma 5.3.6 for parameter d.

Proof. We first verify the conditions of Lemma 5.5.5. Observe that by the assumption $\alpha_M < 2\alpha_L$ we indeed have $t = \Theta(L^2/M)$ and $t \geq 2$ (for sufficiently large n). From the parameter relation $M \geq L^2/|\Sigma|$ we obtain $t \leq |\Sigma|$, and from the parameter relation $d \geq |\Sigma|$ (and $\alpha_\Delta > 0$) this yields $r \geq 1$. Thus, $1 \leq t' \leq t$. Moreover, $r = \Theta(\min\{\Delta, \sqrt{d/t}\})$.

Observe that $r \leq Rt' \leq 2r$. Indeed, if $r \leq t$ then $R = 1$ and $t' = r$, and if $r > t$ then $r/t \leq R \leq 2r/t$ and $t' = t$. In particular, we have

$$Rt' = \Theta\big(\min\{\Delta, \sqrt{d/t}\}\big) \quad \text{and} \quad S = \Theta\Big(\frac{d}{Rt' \cdot t}\Big).$$

Note that $R(t'+1) \leq 2Rt' \leq 4r \leq S$, since $r \leq \sqrt{d/t}$. In particular, this yields $1 \leq R \leq S$, so that all conditions of Lemma 5.5.5 are satisfied.

In the remainder we show that x, y satisfy the parameter constraints. We have $n(x,y) \leq (R+S)t = \mathcal{O}(St) = \mathcal{O}(d/(Rt')) = \mathcal{O}(d/\Delta + \sqrt{dt})$. Note that $d/\Delta = \mathcal{O}(L)$ by the parameter relation $d \leq 2L(\Delta+1)$, and that $\sqrt{dt} = \mathcal{O}(\sqrt{dL^2/M}) = \mathcal{O}(L)$ by the parameter relation $d \leq M$. Thus, $L(x,y) \leq m(x,y) \leq n(x,y) = \mathcal{O}(L) = \mathcal{O}(m) = \mathcal{O}(n)$, which satisfies the parameters L, m, n.

For d, note that by Lemma 5.5.5(ii), we have $d = \Theta((Rt')\cdot(St)) = \Theta((Rt')\cdot d/(Rt')) = \Theta(d)$. For M, Lemma 5.5.5(iii) shows that $M(x,y) = \mathcal{O}(S^2 t) = \mathcal{O}((d/(Rt'))^2 \cdot (1/t)) = \mathcal{O}(L^2 \cdot (M/L^2)) = \mathcal{O}(M)$, where we used $d/(Rt') = \mathcal{O}(L)$ as shown above. Since $L(x,y) = |b| = St$, we obtain $\delta(x,y) = 0 = \mathcal{O}(\delta)$ and $\Delta(x,y) = Rt' = \mathcal{O}(\Delta)$. Finally, $|\Sigma(x,y)| = t = \Theta(L^2/M) = \mathcal{O}(|\Sigma|)$ follows from the parameter relation $M \geq L^2/|\Sigma|$. Observing that x, y, and $L(x,y) = St$ can be computed in time $\mathcal{O}(n)$ concludes the proof. □

5.8 Small Constant Alphabets

In this section, we show hardness of the parameter settings $\mathrm{LCS}(\alpha, \Sigma)$ for alphabets of constant size $|\Sigma| \geq 2$, i.e., we prove Theorem 5.2.3. The general approach, as outlined in Section 5.3, is to take the hard instances x, y of $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha}, \{0,1\})$ constructed in Section 5.6 and pad them to instances x', y' of $\mathrm{LCS}(\boldsymbol{\alpha}, \Sigma)$. Notably, unlike the black-box method of Lemma 5.3.4 that effectively considered each parameter separately, we now cannot make extensive use of the Disjoint Alphabets Lemma, as this would introduce more symbols than admissible. Instead, for small alphabet sizes (such as $|\Sigma| = 2$), we need to pad all parameters simultaneously in a combined construction, taking care of the interplay of the parameters manually. Additionally, for $|\Sigma| \in \{2, 3\}$, more complex parameter relations hold.

Unfortunately, this general approach fails for $\Sigma = \{0,1\}$, i.e., we cannot always pad hard strings x, y of $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha}, \{0,1\})$ to $\mathrm{LCS}(\boldsymbol{\alpha}, \{0,1\})$. Surprisingly, the reason is that by an $\mathcal{O}(n + \delta M/n)$-time algorithm (given in Section 6.2), some parameter settings $\mathrm{LCS}(\boldsymbol{\alpha}, \{0,1\})$ are indeed simpler to solve than $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha}, \{0,1\})$ (conditional on SETH). In these cases, we take hard instances (n, x, y) from $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha}', \{0,1\})$ for a suitably defined "simpler" parameter setting $\boldsymbol{\alpha}'$ and pad x, y to instances of $\mathrm{LCS}(\boldsymbol{\alpha}, \{0,1\})$.

Recall that given a parameter setting $\boldsymbol{\alpha}$, for each parameter $p \in \mathcal{P}$, we write $p = \lceil n^{\alpha_p} \rceil$ for its target value. Furthermore, we may use the assumption (LB) from Section 5.6, i.e., $\alpha_L, \alpha_m, \alpha_\delta, \alpha_\Delta > 0$ and $\alpha_d > 1$. As in Section 5.6, we distinguish between the two cases $\alpha_\delta = \alpha_m$ (i.e., $\delta = \Theta(m)$ and any $0 < \alpha_L \leq \alpha_m$ is admissible) and $\alpha_L = \alpha_m$ (i.e., $L = \Theta(m)$ and any $0 < \alpha_\delta < \alpha_m$ is admissible).

5.8.1 Small LCS

In this section, we assume $\alpha_\delta = \alpha_m$. It can be checked that this assumption implies $\alpha_\Delta = 1$, i.e., $\Delta = \Theta(n)$. Moreover, if $|\Sigma| = 2$, the assumption and the parameter relation $M \geq nd/(80L) \geq \Omega(nd/m)$ imply $\delta M/n = \Omega(d)$. Thus, the desired running time bound

simplifies to $d^{1-o(1)}$. Theorem 5.2.3 in this regime follows from the following statement (and Lemma 5.3.1).

Lemma 5.8.1. *Let $(\boldsymbol{\alpha}, \Sigma)$ be a parameter setting satisfying Table 5.2 with $\alpha_\delta = \alpha_m$. There is a constant $\gamma \geq 1$ such that any algorithm for $\mathrm{LCS}^\gamma(\boldsymbol{\alpha}, \Sigma)$ takes time $d^{1-o(1)}$ unless* OVH *fails.*

We prove the above lemma in the remainder of this section. Note that any parameter setting $(\boldsymbol{\alpha}, \Sigma)$ satisfying Table 5.2 gives rise to a parameter setting $\boldsymbol{\alpha}$ satisfying Table 5.2 with $\alpha_\Sigma = 0$ (where the converse does not hold in general). Recall that for any such $\boldsymbol{\alpha}$, in Lemma 5.6.2 we constructed hard instances (n, x, y) of $\mathrm{LCS}^\gamma_\leq(\boldsymbol{\alpha}, \{0,1\})$ with an additional threshold ρ such that deciding $L(x,y) \geq \rho$ decides the corresponding OV instance, yielding hardness of $\mathrm{LCS}^\gamma_\leq(\boldsymbol{\alpha}, \{0,1\})$. Furthermore, the constructed instances have the additional guarantees that $|x|, |y| \leq \gamma \cdot n^{\alpha_L}$ and $\#_1(y) \leq \gamma \cdot n^{\alpha_d - \alpha_L}$ and for any $\beta \geq 0$ we have $L(x, 0^\beta y) = L(x,y)$.

Hence, to prove the above lemma it suffices to show how to compute, given any such instance (n, x, y) and threshold ρ, an instance x', y' of $\mathrm{LCS}^{\gamma'}(\boldsymbol{\alpha}, \Sigma)$ (for some $\gamma' \geq 1$) and an integer τ in time $\mathcal{O}(n)$ such that $L(x', y') = L(x,y) + \tau$.

We do this successively for alphabet sizes $|\Sigma| = 2, |\Sigma| = 3, |\Sigma| = 4$, and $|\Sigma| \geq 5$. To this end, the following basic building block will be instantiated with different parameters. Recall that in Lemma 5.5.3, we defined strings $a = (01)^{R+S}$ and $b = 0^R(01)^S$ with the properties $L(a,b) = |b| = R + 2S$ and $d(a,b) = \Theta(R \cdot S)$.

Lemma 5.8.2 (Basic building block)**.** *Let x, y be given strings. Given $\alpha, \beta, R, S \geq 0$, we set $\ell := |x| + |y|$, and define*

$$\begin{aligned} x' &:= a\ 1^\alpha\ 0^\ell\ x &&= (01)^{R+S}\ 1^\alpha\ 0^\ell\ x \\ y' &:= b\ 0^\beta\ 0^\ell\ y &&= 0^R(01)^S\ 0^\beta\ 0^\ell\ y. \end{aligned}$$

Then we have $L(x', y') = L(a,b) + \ell + L(x, 0^\beta y) = R + 2S + \ell + L(x, 0^\beta y)$. If $L(x, 0^\beta y) = L(x,y)$ then we even have $L(x', y') = R + 2S + \ell + L(x,y)$.

Proof. Clearly, $L(x', y') \geq L(a,b) + L(0^\ell, 0^\ell) + L(x, 0^\beta y) = (R + 2S) + \ell + L(x, 0^\beta y)$ since $L(a,b) = |b| = R + 2S$ by Lemma 5.5.3. To prove a corresponding upper bound, note that we can partition $y' = wz$ such that $L(x', y') = L(a1^\alpha, w) + L(0^\ell x, z)$. Consider first the case that z is a subsequence of y. Then

$$L(x', y') = L(a1^n, w) + L(0^\ell x, z) \leq L(a1^n, y') + L(0^\ell x, y),$$

since w, z are subsequences of y', y, respectively. Using the simple fact that $L(u,v) \leq \sum_{\sigma \in \Sigma} \min\{\#_\sigma(u), \#_\sigma(v)\}$ for any strings u, v, we obtain

$$\begin{aligned} L(x', y') &\leq (\#_0(a1^n) + \#_1(y')) + (\#_0(y) + \#_1(0^\ell x)) \\ &= (R + S) + (S + \#_1(y)) + \#_0(y) + \#_1(x) \\ &\leq (R + 2S) + \ell + L(x, 0^\beta y), \end{aligned}$$

since $\ell \geq |x| + |y| \geq \#_0(x) + \#_0(y) + \#_1(y)$. It remains to consider the case that z is not a subsequence of y and hence w is a subsequence of $b0^{\beta+\ell}$. By Lemma 5.5.3(iii), we can without loss of generality assume that w is a subsequence of b, since $L(a1^\alpha, b0^{\beta+\ell}) = L(a,b)$. We write $z = z'z''$ such that z'' is a subsequence of $0^{\ell+\beta}y$ and maximal with this property. Hence, wz' is a subsequence of b. Using the facts $L(u,v) \leq |v|$ and

$L(u, v'v'') \leq |v'| + L(u, v'')$, we bound

$$L(x', y') = L(a1^n, w) + L(0^\ell x, z'z'') \leq |w| + (|z'| + L(0^\ell x, z'')),$$

Since wz' is a subsequence of b and z'' is a subsequence of $0^{\ell+\beta}y$, this yields

$$L(x', y') \leq |b| + L(0^\ell x, 0^{\ell+\beta}y) = (R + 2S) + \ell + L(x, 0^\beta y),$$

where we used greedy prefix matching. This finishes the proof. □

We now instantiate the basic building block to prove Lemma 5.8.1 for $\Sigma = \{0, 1\}$. Note that in the remainder, we again simply write p for the target value $\lceil n^{\alpha_p} \rceil$ of each parameter $p \in \mathcal{P}$, while the parameter value attained by any strings x, y is denoted by $p(x, y)$, as in Section 5.6. Note that the additional guarantees for (n, x, y) are satisfied by Lemma 5.6.2.

Lemma 5.8.3. *Consider a parameter setting $(\boldsymbol{\alpha}, \{0, 1\})$ satisfying Table 5.2 with $\alpha_\delta = \alpha_m$. Let (n, x, y) be an instance of $\mathrm{LCS}^\gamma_\leq(\boldsymbol{\alpha}, \{0, 1\})$ with $|x|, |y| \leq \gamma \cdot L$ and $\#_1(y) \leq \gamma \cdot d/L$ satisfying $L(x, 0^{\beta'}y) = L(x, y)$ for any $\beta' \geq 0$. We obtain strings x', y' from Lemma 5.8.2 (recall that in this lemma we set $\ell := |x| + |y|$), where we choose*

$$R := L, \quad S := \lfloor d/L \rfloor, \quad \beta := \tilde{m} := \max\{m, 2|x|\}, \quad \alpha := \tilde{n} := \max\{n, \tilde{m} + |y|\}.$$

Then, setting $\kappa := \lfloor M/n \rfloor$, the strings defined by

$$\begin{aligned} x'' :=\ & 1^\kappa\, x' &=\ & 1^\kappa\, a\, 1^{\tilde{n}}\, 0^\ell\, x &=\ & 1^\kappa\, (01)^{R+S}\, 1^{\tilde{n}}\, 0^\ell\, x, \\ y'' :=\ & 1^\kappa\, y' &=\ & 1^\kappa\, b\, 0^{\ell+\tilde{m}}\, y &=\ & 1^\kappa\, 0^R (01)^S\, 0^{\ell+\tilde{m}}\, y. \end{aligned}$$

are an instance of $\mathrm{LCS}^{\gamma'}(\boldsymbol{\alpha}, \{0, 1\})$ (for some constant $\gamma' \geq 1$) and can be computed in time $\mathcal{O}(n)$, together with an integer τ such that $L(x'', y'') = \tau + L(x, y)$.

Proof. Note that

$$L(x'', y'') = \kappa + L(x', y') = \kappa + (R + 2S) + \ell + L(x, y), \tag{5.4}$$

where the first equality follows from greedy prefix matching and the second follows from Lemma 5.8.2. Thus by setting $\tau = \kappa + (R + 2S) + \ell$, we have that $L(x'', y'') = \tau + L(x, y)$. Clearly, x'', y'', and τ can be computed in time $\mathcal{O}(n)$, and $\Sigma(x'', y'') = \{0, 1\}$.

We first verify that $|x|, |y|, \ell, R, S, |a|, |b|, \kappa = \mathcal{O}(L)$. By assumption, $|x|, |y| = \mathcal{O}(L)$ and thus $\ell = |x| + |y| = \mathcal{O}(L)$. By the parameter relation $d \leq |\Sigma| \cdot L^2 = 2L^2$, we note that $d/L = \mathcal{O}(L)$ and hence by choice of R, S, we have $|a|, |b| = \Theta(R + S) = \Theta(L + d/L) = \Theta(L)$. Furthermore, the parameter relation $M \leq 2Ln$ implies $\kappa \leq M/n \leq 2L$. Since $L(x, y) \leq |x| = \mathcal{O}(L)$, the bound $L(x'', y'') = \kappa + R + 2S + \ell + L(x, y) = R + \mathcal{O}(L) = \Theta(L)$ follows directly from (5.4).

Observe that $\tilde{n}$ is chosen such that $|x''| \geq |y''|$. Also, $\tilde{m} = \Theta(m)$ and $\tilde{n} = \Theta(n)$. Since $L \leq m \leq n$, we thus have $|x''| = \kappa + |a| + \tilde{n} + \ell + |x| = \tilde{n} + \mathcal{O}(L) = \Theta(n)$ and $|y| = \kappa + |b| + \tilde{m} + \ell + |y| = \tilde{m} + \mathcal{O}(L) = \Theta(m)$.

Note that by (5.4), $\delta(x'', y'') = (\tilde{m} + |y|) - L(x, y) \geq \tilde{m} - |x| \geq m/2$. Hence, $\delta(x'', y'') = \Theta(m) = \Theta(\delta)$ (by the assumption $\alpha_\delta = \alpha_m$). Moreover, since $\delta = \Theta(m)$, for some constant $\varepsilon > 0$ we have $\Delta = \delta + (n - m) \geq \varepsilon m + n - m = n - (1 - \varepsilon)m \geq n - (1 - \varepsilon)n = \Omega(n)$ (where we used the parameter relation $m \leq n$). Since also $\Delta \leq n$ we have $\Delta = \Theta(n)$. By the same argument, using $\delta(x'', y'') = \Theta(m) = \Theta(m(x'', y''))$ and

$n(x'', y'') = \Theta(n)$ as shown above, we obtain $\Delta(x'', y'') = \Theta(n(x'', y'')) = \Theta(n)$, and thus $\Delta(x'', y'') = \Theta(\Delta)$.

For M, observe that $\#_1(x'') = \kappa + \#_1(a) + \tilde{n} + \#_1(x) = \tilde{n} + \mathcal{O}(L) = \Theta(n)$. Moreover, $\#_1(y) = \mathcal{O}(d/L)$ (by assumption) and $\#_1(b) = S = \mathcal{O}(d/L)$ yield $\#_1(y'') = \kappa + \#_1(b) + \#_1(y) = \Theta(M/n) + \mathcal{O}(d/L)$ (here $\kappa = \Theta(M/n)$ follows from the parameter relation $M \geq n$). This yields $\#_1(x'') \cdot \#_1(y'') = \Theta(M) + \mathcal{O}(dn/L) = \Theta(M)$ (using the parameter relation $M \geq nd/(5L)$). Also note that $\#_0(x'') = \#_0(a) + \ell + \#_0(x) = \mathcal{O}(L)$ and $\#_0(y'') = \#_0(b) + \ell + \tilde{m} + \#_0(y) = \tilde{m} + \mathcal{O}(L) = \mathcal{O}(m)$. This yields $\#_0(x'') \cdot \#_0(y'') = \mathcal{O}(Lm) = \mathcal{O}(M)$ (using the parameter relation $M \geq Lm/4$). Combining these bounds, we obtain $M(x'', y'') = \#_0(x'') \cdot \#_0(y'') + \#_1(x'') \cdot \#_1(y'') = \Theta(M)$. Note that the last two parameter relations used here exploited that we have $\Sigma = \{0, 1\}$.

It remains to determine the number of dominant pairs. Since $L(x', y') = \Theta(L)$ (as argued above) and $\#_1(y') = \mathcal{O}(d/L)$, Lemma 5.4.8 yields $d(x', y') \leq 5L(x', y') \cdot \#_1(y') = \mathcal{O}(L \cdot d/L) = \mathcal{O}(d)$. For a corresponding lower bound, from Observation 5.5.2 and Lemma 5.5.3 we obtain $d(x', y') \geq d(a, b) \geq R \cdot S = \Omega(d)$. By Lemma 5.5.1, the claim now follows from $d(x'', y'') = \kappa + d(x', y') = \mathcal{O}(L) + \Theta(d) = \Theta(d)$, where we use $\kappa = \mathcal{O}(L)$ and the parameter relation $d \geq L$. □

The case $\Sigma = \{0, 1, 2\}$ is similar to $\{0, 1\}$, except that we use the new symbol 2 to pad the parameter n, we use symbol 1 to pad m, and we have to swap the constructions for x'' and y''.

Lemma 5.8.4. *Consider a parameter setting $(\boldsymbol{\alpha}, \{0, 1, 2\})$ satisfying Table 5.2 with $\alpha_\delta = \alpha_m$. Let (n, x, y) be an instance of $\mathrm{LCS}^{\gamma}_{\leq}(\boldsymbol{\alpha}, \{0, 1\})$ with $|x|, |y| \leq \gamma \cdot L$ and $\#_1(y) \leq \gamma \cdot d/L$ satisfying $L(x, 0^{\beta'} y) = L(x, y)$ for any $\beta' \geq 0$. We obtain strings x', y' from Lemma 5.8.2 (recall that in this lemma, we set $\ell := |x| + |y|$), where we choose*

$$R := L, \qquad S := \lfloor d/L \rfloor, \qquad \beta := 0, \qquad \alpha := m.$$

Then, setting $\kappa := \lfloor M/n \rfloor$ and $\tilde{n} := \max\{n, \kappa + |a| + m + |x|\}$, the strings defined by

$$\begin{aligned} x'' := \quad & 2^{\tilde{n}}\, y' \quad = \quad 2^{\tilde{n}}\, b \quad 0^{\ell}\, y \quad = \quad 2^{\tilde{n}}\, 0^{R} (01)^{S} \quad 0^{\ell}\, y, \\ y'' := \quad & 2^{\kappa}\, x' \quad = \quad 2^{\kappa}\, a\, 1^{m}\, 0^{\ell}\, x \quad = \quad 2^{\kappa}\, (01)^{R+S}\, 1^{m}\, 0^{\ell}\, x. \end{aligned}$$

are an instance of $\mathrm{LCS}^{\gamma'}(\boldsymbol{\alpha}, \{0, 1, 2\})$ (for some constant $\gamma' \geq 1$) and can be computed in time $\mathcal{O}(n)$, together with an integer τ such that $L(x'', y'') = \tau + L(x, y)$.

Proof. Note that unlike the case $\{0, 1\}$, the string x now appears in y'' and y appears in x'', so the constructions are swapped. This is necessary to realize m and M, using the parameter relation $M \geq md/(80L)$ that holds for $\Sigma = \{0, 1, 2\}$. Observe that as usual, $|x''| \geq |y''|$.

We first compute

$$L(x'', y'') = L(2^{\tilde{n}}, 2^{\kappa}) + L(y', x') = \kappa + (R + 2S) + \ell + L(x, y), \tag{5.5}$$

where the first equality follows from the Disjoint Alphabets Lemma and the second equality from greedy prefix matching and Lemma 5.8.2. Thus, by setting $\tau = \kappa + (R + 2S) + \ell$, we have $L(x'', y'') = \tau + L(x, y)$. Clearly, x'', y'', and τ can be computed in time $\mathcal{O}(n)$, and $\Sigma(x'', y'') = \{0, 1, 2\}$.

As in the case $\{0, 1\}$, we have $|x|, |y|, \ell, R, S, |a|, |b|, \kappa = \mathcal{O}(L)$. Thus, by (5.5), we have $L(x'', y'') = R + \mathcal{O}(L) = \Theta(L)$. Furthermore, note that $\tilde{n} = \Theta(n)$. Thus,

$|y''| = \kappa+|a|+m+\ell+|x| = m+\mathcal{O}(L) = \Theta(m)$ and $|x''| = \tilde{n}+|b|+\ell+|y| = \tilde{n}+\mathcal{O}(L) = \Theta(n)$. Since $L(x,y) \leq |x| = \mathcal{O}(L)$, the bound $L(x'',y'') = R + \mathcal{O}(L) = \Theta(L)$ follows directly from (5.5).

By (5.5), we see that $\delta(x'',y'') = |y''| - L(x'',y'') = R + m + (|x| - L(x,y)) \geq m$. Hence $\delta(x'',y'') = \Theta(m)$. Thus, $\Delta(x'',y'') = \delta(x'',y'') + (|x''| - |y''|) = \Theta(n)$ follows as in the case $\{0,1\}$.

For M, observe that $\#_1(y'') = \#_1(a) + m + \#_1(x) = m + \mathcal{O}(L) = \Theta(m)$. Moreover, $\#_1(y) = \mathcal{O}(d/L)$ (by assumption) yields $\#_1(x'') = \#_1(b)+\#_1(y) = S+\mathcal{O}(d/L) = \mathcal{O}(d/L)$. Also note that $\#_0(y'') = \#_0(a) + \ell + \#_0(x) = \mathcal{O}(L)$ and $\#_0(x'') = \#_0(b) + \ell + \#_0(y) = \mathcal{O}(L)$. Since furthermore $\#_2(y'') = \Theta(M/n)$ (by the parameter relation $M \geq n$) and $\#_2(x'') = \Theta(n)$, we conclude that $M(x'',y'') = \sum_{\sigma\in\{0,1,2\}} \#_\sigma(x'') \cdot \#_\sigma(y'') = \mathcal{O}(dm/L + L^2) + \Theta(M)$. By the parameter relations $M \geq md/(80L)$ (using that $\Sigma = \{0,1,2\}$) and $M \geq L^2/|\Sigma| = \Omega(L^2)$, this yields $M(x'',y'') = \Theta(M)$.

For the remaining parameter d, by the disjoint alphabets lemma and Lemma 5.5.1 we have $d(x'',y'') = d(2^{\tilde{n}}, 2^\kappa) + d(y',x') = \kappa + d(x',y')$ (using symmetry $d(x,y) = d(y,x)$). The remaining arguments are the same as in the case $\{0,1\}$. □

In the case $\Sigma = \{0,1,2,3\}$ we can use the new symbol 3 to pad m (instead of using symbol 1, as in the previous case). Note that now x appears in x'' and y in y'', as in the case $\{0,1\}$.

Lemma 5.8.5. *Consider a parameter setting $(\boldsymbol{\alpha}, \{0,1,2,3\})$ satisfying Table 5.2 with $\alpha_\delta = \alpha_m$. Let (n,x,y) be an instance of $\mathrm{LCS}^\gamma_\leq(\boldsymbol{\alpha}, \{0,1\})$ with $|x|,|y| \leq \gamma \cdot L$ and $\#_1(y) \leq \gamma \cdot d/L$ satisfying $L(x, 0^{\beta'}y) = L(x,y)$ for any $\beta' \geq 0$. We obtain strings x', y' from Lemma 5.8.2 (recall that in this lemma, we set $\ell := |x| + |y|$), where we choose*

$$R := L, \qquad S := \lfloor d/L \rfloor, \qquad \beta := 0, \qquad \alpha := 0.$$

Then, setting $\kappa := \lfloor M/n \rfloor$ and $\tilde{n} := \max\{n, m + \kappa + |y|\}$, the strings defined by

$$\begin{aligned} x'' :=\ & 3\ \ 2^{\tilde{n}}\ x' &=\ & 3\ \ 2^{\tilde{n}}\ a\ 0^\ell\ x &=\ & 3\ \ 2^{\tilde{n}}\ (01)^{R+S}\ 0^\ell\ x, \\ y'' :=\ & 3^m\ 2^\kappa\ y' &=\ & 3^m\ 2^\kappa\ b\ 0^\ell\ y &=\ & 3^m\ 2^\kappa\ 0^R(01)^S\ 0^\ell\ y, \end{aligned}$$

are an instance of $\mathrm{LCS}^{\gamma'}(\boldsymbol{\alpha}, \{0,1,2,3\})$ (for some constant $\gamma' \geq 1$) and can be computed in time $\mathcal{O}(n)$, together with an integer τ such that $L(x'',y'') = \tau + L(x,y)$.

Proof. We compute

$$L(x'',y'') = L(3,3^m) + L(2^{\tilde{n}}, 2^\kappa) + L(x',y') = 1 + \kappa + (R+2S) + \ell + L(x,y), \quad (5.6)$$

where the first equality follows from the Disjoint Alphabets Lemma and the second follows from greedy prefix matching and Lemma 5.8.2. Hence, by setting $\tau = 1+\kappa+R+2S+\ell$, we have $L(x'',y'') = \tau + L(x,y)$. Clearly, x'', y'', and τ can be computed in time $\mathcal{O}(n)$, and $\Sigma(x'',y'') = \{0,1,2,3\}$.

As for the cases $\{0,1\}$ and $\{0,1,2\}$, we have $|x|,|y|,\ell,R,S,|a|,|b|,\ell,\kappa = \Theta(L)$. Note that by choice of $\tilde{n}$, we have again $|x''| \geq |y''|$ and $\tilde{n} = \Theta(n)$. Hence, $|x''| = 1 + \tilde{n} + |a| + \ell + |x| = \tilde{n} + \mathcal{O}(L) = \Theta(n)$ and $|y''| = m + \kappa + |b| + \ell + |y| = m + \mathcal{O}(L) = \Theta(m)$. Since $L(x,y) \leq |x| = \mathcal{O}(L)$, the bound $L(x'',y'') = R+\mathcal{O}(L) = \Theta(L)$ follows directly from (5.6). Note that (5.6) also implies that $\delta(x'',y'') = |y''|-L(x'',y'') = m-1+|x|-L(x,y) \geq m-1$ and hence $\delta(x'',y'') = \Theta(m)$. Thus, $\Delta(x'',y'') = \delta(x'',y'') + (|x''| - |y''|) = \Theta(n)$ follows as for the case $\{0,1\}$.

For M, observe that $|a0^\ell x|, |b0^\ell y| = \mathcal{O}(L)$ implies that $M(a0^\ell x, b0^\ell y) = \mathcal{O}(L^2)$. By the Disjoint Alphabets Lemma, we obtain

$$M(x'', y'') = M(2^{\tilde{n}}, 2^\kappa) + M(3, 3^m) + M(a0^\ell x, b0^\ell y) = \kappa\tilde{n} + \mathcal{O}(m + L^2) = \Theta(M),$$

where we used $\kappa\tilde{n} = \Theta(M/n \cdot n) = \Theta(n)$ (note that $M \geq n$ implies $\kappa = \Theta(M/n)$) and the parameter relations $M \geq n \geq m$ and $M \geq L^2/|\Sigma| = \Omega(L^2)$.

For the remaining parameter d, as in the case $\{0,1\}$ we show that $d(x', y') = \Theta(d)$. Thus, the Disjoint Alphabets Lemma and Lemma 5.5.1 prove that $d(x'', y'') = d(3, 3^m) + d(2^{\tilde{n}}, 2^\kappa) + d(x', y') = 1 + \kappa + d(x', y') = d(x', y') + \mathcal{O}(L) = \Theta(d)$ using $\kappa = \mathcal{O}(L)$ and the parameter relation $d \geq L$. □

Finally, observe that for any parameter setting $(\boldsymbol{\alpha}, \Sigma)$ with $|\Sigma| \geq 5$ satisfying Table 5.2, also the parameter setting $(\boldsymbol{\alpha}, \{0,1,2,3\})$ satisfies Table 5.2. Hence, the following lemma transfers the hardness of LCS$(\boldsymbol{\alpha}, \{0,1,2,3\})$ to LCS$(\boldsymbol{\alpha}, \Sigma)$.

Lemma 5.8.6. *Let $\boldsymbol{\alpha}$ be a parameter setting satisfying Table 5.2 with $\alpha_\delta = \alpha_m$. Let Σ be an alphabet of size $|\Sigma| \geq 5$. If there is an $\mathcal{O}(n^\beta)$-time algorithm for* LCS$(\boldsymbol{\alpha}, \Sigma)$*, then also* LCS$(\boldsymbol{\alpha}, \{0,1,2,3\})$ *admits an $\mathcal{O}(n^\beta)$-time algorithm.*

Proof. Given an instance (x, y) of LCS$(\boldsymbol{\alpha}, \{0,1,2,3\})$ with $n := |x|$, we show how to compute, in time $\mathcal{O}(n)$, an instance (x', y') of LCS$(\boldsymbol{\alpha}, \Sigma)$ such that $L(x', y') = 1 + L(x, y)$. The claim then follows from applying the $\mathcal{O}(n^\beta)$-time algorithm on x', y' (and subtracting 1 from the result).

Without loss of generality, let $\Sigma = \{0, \ldots, \sigma\}$ with $\sigma \geq 4$. Define $x' := wx$ and $y' = w^R y$, where $w = 4 \ldots \sigma$ and $w^R = \sigma \ldots 4$. Then by the Disjoint Alphabets and Crossing Alphabets Lemmas (Lemmas 5.3.5 and 5.6.3), we obtain $L(x', y') = L(w, w^R) + L(x, y) = 1 + L(x, y)$. It remains to show that (x', y') is an instance of LCS$(\boldsymbol{\alpha}, \Sigma)$. By the Crossing Alphabets Lemma, for all parameters $p \in \{d, M, n, m\}$ we have $p(w, w^R) = \sum_{\sigma'=4}^{\sigma} p(\sigma', \sigma') = |\Sigma| - 4$, and hence the Disjoint Alphabets Lemma yields $p(x', y') = p(w, w^R) + p(x, y) = |\Sigma| - 4 + p(x, y) = \Theta(p(x, y))$, by the parameter relations $n \geq m \geq |\Sigma|$ and $M \geq d \geq |\Sigma|$. For L we obtain $L(x', y') = L(w, w^R) + L(x, y) = 1 + L(x, y) = \Theta(L(x, y))$. For $p \in \{\delta, \Delta\}$ this yields $p(x', y') = (|w| - 1) + p(x, y) = \Theta(p(x, y))$, since $\alpha_\Delta \geq \alpha_\delta = \alpha_m \geq \alpha_\Sigma$ (by the assumption $\alpha_\delta = \alpha_m$ and the parameter relations $\Delta \geq \delta$ and $m \geq |\Sigma|$) and thus $\Delta(x, y) \geq \delta(x, y) \geq \Omega(|w| - 1)$. Hence, (x', y') has the same parameters as (x, y) up to constant factors, so all parameter relations satisfied by (x, y) are also satisfied by (x', y'). Since clearly x', y' use alphabet Σ, indeed (x', y') is an instance of LCS$(\boldsymbol{\alpha}, \Sigma)$. □

Lemmas 5.8.3, 5.8.4, 5.8.5, and 5.8.6 of this section, together with the construction of hard strings in LCS$_\leq(\boldsymbol{\alpha}, \{0,1\})$ in Lemma 5.6.2, prove hardness of LCS$(\boldsymbol{\alpha}, \Sigma)$ for any constant alphabet size in the case $\alpha_\delta = \alpha_m$, i.e., Lemma 5.8.1.

5.8.2 Large LCS, Alphabet Size At Least 3

In this section, we study the case that $\alpha_L = \alpha_m$ (and $\alpha_\delta, \alpha_\Delta$ may be small). Additionally, we assume that $|\Sigma| \geq 3$. In this regime, Theorem 5.2.3 follows from the following statement (and Lemma 5.3.1).

Lemma 5.8.7. *Let $(\boldsymbol{\alpha}, \Sigma)$ be a parameter setting satisfying Table 5.2 with $\alpha_L = \alpha_m$ and $|\Sigma| \geq 3$. There is a constant $\gamma \geq 1$ such that any algorithm for* LCS$^\gamma(\boldsymbol{\alpha}, \Sigma)$ *takes time* $\min\{d, \delta m, \delta\Delta\}^{1-o(1)}$ *unless* OVH *fails.*

By the following lemma, it suffices to prove the result for $\Sigma = \{0,1,2\}$ (note that for any $(\boldsymbol{\alpha}, \Sigma)$ satisfying Table 5.2 with $\alpha_L = \alpha_m$ and $|\Sigma| \geq 4$, also $(\boldsymbol{\alpha}, \{0,1,2\})$ satisfies Table 5.2, since the only additional constraint $\alpha_M \geq \alpha_m + \alpha_d - \alpha_L$ for ternary alphabets simplifies, by $\alpha_L = \alpha_m$, to the constraint $\alpha_M \geq \alpha_d$, which is satisfied by $\boldsymbol{\alpha}$).

Lemma 5.8.8. *Let $\boldsymbol{\alpha}$ be a parameter setting satisfying Table 5.2 with $\alpha_L = \alpha_m$. Let Σ be an alphabet of size $|\Sigma| \geq 4$. If there is an $\mathcal{O}(n^\beta)$-time algorithm for* LCS$(\boldsymbol{\alpha}, \Sigma)$*, then also* LCS$(\boldsymbol{\alpha}, \{0,1,2\})$ *admits an $\mathcal{O}(n^\beta)$-time algorithm.*

Proof. Given an instance (x, y) of LCS$(\boldsymbol{\alpha}, \{0,1,2\})$ with $n := |x|$, we show how to compute in time $\mathcal{O}(n)$ an instance (x', y') of LCS$(\boldsymbol{\alpha}, \Sigma)$ such that $L(x', y') = |\Sigma| - 3 + L(x, y)$. The claim then follows from applying the $\mathcal{O}(n^\beta)$-time algorithm on x', y' (and subtracting $|\Sigma| - 3$ from the result).

Without loss of generality, let $\Sigma = \{0, \ldots, \sigma\}$ with $\sigma \geq 3$. Define $x' := wx$ and $y' = wy$, where $w = 3 \ldots \sigma$. Then by the Disjoint Alphabets Lemma (Lemma 5.3.5), we obtain $L(x', y') = L(w, w) + L(x, y) = |\Sigma| - 3 + L(x, y)$. It remains to show that (x', y') is an instance of LCS$(\boldsymbol{\alpha}, \Sigma)$. By the Disjoint Alphabets Lemma, for all parameters $p \in \mathcal{P}^* = \{n, m, L, \delta, \Delta, |\Sigma|, M, d\}$ we have $p(x', y') = p(w, w) + p(x, y) = \sum_{\sigma'=3}^{\sigma} p(\sigma', \sigma') + p(x, y)$. For $p \in \{n, m, L, M, d\}$ we have $p(\sigma', \sigma') = 1$ and thus $p(x', y') = |\Sigma| - 3 + p(x, y) = \Theta(p(x, y))$ by the assumption $\alpha_L = \alpha_m$ and the parameter relations $n \geq m \geq |\Sigma|$ and $M \geq d \geq |\Sigma|$. For $p \in \{\delta, \Delta\}$ this yields $p(x', y') = 0 + p(x, y) = p(x, y)$. Hence, (x', y') has the same parameters as (x, y) up to constant factors, so all parameter relations satisfied by (x, y) are also satisfied by (x', y'). Since clearly x', y' use alphabet Σ, indeed (x', y') is an instance of LCS$(\boldsymbol{\alpha}, \Sigma)$. □

To prepare the proof of the main result in this section, we adapt the construction of Lemma 5.8.2 to obtain a desired value for d (and, in later sections, δ). Recall that Lemma 5.5.3 defines $a = (01)^{R+S}$, $b = 0^R(01)^S$ with $L(a, b) = |b| = R + 2S$ and $d(a, b) = \Theta(R \cdot S)$.

Lemma 5.8.9 (Basic building block II)**.** *Given strings x, y and $R, S, \ell, \beta \geq 0$ with $\ell \geq R + |x| + |y|$ we define*

$$\begin{aligned} x' &:= a\ 0^\ell\ x &&= (01)^{R+S}\ 0^\ell\ x, \\ y' &:= 0^\beta\ b\ 0^\ell\ y &&= 0^\beta\ 0^R(01)^S\ 0^\ell\ y. \end{aligned}$$

Assume that $S \geq |x|$ or $L(x, 0^\beta y) = L(x, y)$. Then we have $L(x', y') = R + 2S + \ell + L(x, y)$.

Proof. Clearly, $L(x', y') \geq L(a, b) + L(0^\ell, 0^\ell) + L(x, y) = R + 2S + \ell + L(x, y)$, since $L(a, b) = |b| = R + 2S$. To prove the corresponding upper bound, we partition $y' = y_1 y_2$ such that $L(x', y') = L(a, y_1) + L(0^\ell x, y_2)$. Consider first the case that y_2 is a subsequence of y. Then

$$L(x', y') \leq L(a, y_1) + L(0^\ell x, y) \leq |a| + |y| = 2(R + S) + |y| \leq (R + 2S) + \ell + L(x, y),$$

since $\ell \geq R + |y|$.

It remains to consider the case that y_2 is not a subsequence of y and hence y_1 is a subsequence of $0^\beta b 0^\ell$. By Lemma 5.5.3(iii), we can without loss of generality assume that y_1 is a subsequence of $0^\beta b$, since $L(a, 0^\beta b 0^\ell) = |b| = L(a, 0^\beta b)$. Hence, we can partition $0^\beta b = y_1 z$ with $L(x', y') \leq L(a, y_1) + L(0^\ell x, z 0^\ell y)$. We bound

$$L(a, y_1) \leq \min\{\#_0(a), \#_0(y_1)\} + \min\{\#_1(a), \#_1(y_1)\} \leq (R + S) + \#_1(y_1).$$

Observe that $L(0^\ell x, z0^\ell y) \le \#_1(z) + L(0^\ell x, 0^{\#_0(z)+\ell} y)$, since each "1" in z can increase the LCS by at most 1. By greedy prefix matching, we obtain $L(0^\ell x, z0^\ell y) \le \#_1(z) + \ell + L(x, 0^{\#_0(z)} y)$. With the assumption $L(x, 0^\beta y) = L(x, y)$ for any $\beta \ge 0$, this yields

$$\begin{aligned} L(x', y') &\le L(a, y_1) + L(0^\ell x, z0^\ell y) \\ &\le (R + S + \#_1(y_1)) + (\#_1(z) + \ell + L(x, y)) = R + 2S + \ell + L(x, y), \end{aligned}$$

since $0^\beta b = y_1 z$ and hence $\#_1(y_1) + \#_1(z) = \#_1(0^\beta b) = S$.

With the alternative assumption $S \ge |x|$, we bound $L(0^\ell x, z0^\ell y) \le |0^\ell x| \le \ell + S$. If $\#_1(y_1) = 0$ this yields

$$L(x', y') \le L(a, y_1) + L(0^\ell x, z0^\ell y) \le (R + S) + (\ell + S) \le R + 2S + \ell + L(x, y).$$

Otherwise, if $\#_1(y_1) \ge 1$, by inspecting the structure of $0^\beta b = 0^{\beta+R}(01)^S$ we see that z is a subsequence of $(01)^{S-\#_1(y_1)}$. Consider an LCS of $0^\ell x, z0^\ell y$. If no "1" in z is matched by the LCS, then we obtain

$$L(0^\ell x, z0^\ell y) \le L(0^\ell x, 0^{\#_0(z)+\ell} y) \le \#_0(z) + L(0^\ell x, 0^\ell y) = \#_0(z) + \ell + L(x, y),$$

where we used the fact $L(u, vw) \le |v| + L(uw)$ and greedy prefix matching. Otherwise, if a "1" in z is matched by the LCS to a "1" in $0^\ell x$, i.e., to a "1" in x, then the 0^ℓ-block of $0^\ell x$ is matched to a subsequence of z and hence

$$L(0^\ell x, z0^\ell y) \le L(0^\ell, z) + L(x, z0^\ell y) \le \#_0(z) + |x| \le \#_0(z) + \ell + L(x, y),$$

where we used $\ell \ge |x|$. Since z is a subsequence of $(01)^{S-\#_1(y_1)}$ and thus $\#_0(z) \le S - \#_1(y_1)$, in both cases we obtain

$$\begin{aligned} L(x', y') &\le L(a, y_1) + L(0^\ell x, z0^\ell y) \\ &\le (R + S + \#_1(y_1)) + (\#_0(z) + \ell + L(x, y)) \le R + 2S + \ell + L(x, y), \end{aligned}$$

which finally proves the claim. □

Lemma 5.8.10. *Consider x', y' as in Lemma 5.8.9 with $\beta = 0$, and assume $S \ge |x|$. Then*

$$R \cdot S \le d(x', y') \le (R + 1)(4R + 6S + \ell) + d(x, y).$$

Proof. Note that the simple fact $L(u'u'', v) \le |u''| + L(u', v)$ implies that for any strings w, z and any i, we have $L(w, z) \le |w[(i + 1)..|w|]| + L(w[1..i], z) = |w| - i + L(w[1..i], z)$, and hence $L(w[1..i], z) \ge i - (|w| - L(w, z))$. Recall that $L(a, b) = |b| = R + 2S$ by Lemma 5.5.3(iii). This yields $L(a[1..i], b) \ge i - (|a| - (R + 2S)) = i - R$ and $L(a, b[1..j]) \ge j - (|b| - |b|) = j$.

The claimed lower bound follows from $d(x', y') \ge d(a, b) \ge R \cdot S$ by Observation 5.5.2 and Lemma 5.5.3(iv). For the upper bound, we consider all possible prefixes $\tilde{x} := x'[1..i], \tilde{y} := y'[1..j]$ of x', y' and count how many of them correspond to dominant pairs. Clearly, for $i \le |a|, j \le |b|$, there are $d(a, b)$ dominant pairs.

By the above observation, for any $i \le |a|$, we have $L(a[1..i], b) \ge i - R$. Hence, any dominant pair of the form (i, j) satisfies $i - R \le L(a[1..i], b) \le |a[1..i]| = i$. By Observation 5.4.2, there are at most $R + 1$ such dominant pairs for fixed i. Thus, there are at most $|a| \cdot (R + 1)$ dominant pairs with $i \le |a|$ and $j > |b|$. Similarly, for $j \le |b|$, we have $L(a, b[1..j]) \ge j$. Hence, there are no dominant pairs with $i > |a|$ and $j \le |b|$,

since already the prefix a of $x'[1..i]$ includes $b[1..j]$ as a subsequence. In total, there are at most $d(a,b) + |a| \cdot (R+1)$ dominant pairs with $i \leq |a|$ or $j \leq |b|$.

Let $i = |a| + k$, $j = |b| + k$ with $k \in [\ell]$. Then $L(\tilde{x}, \tilde{y}) = L(a0^k, b0^k) = L(a,b) + k = |b| + k$ by greedy suffix matching and Lemma 5.5.3(iii). As any such choice could correspond to a dominant pair, we count at most ℓ dominant pairs. Analogously to above, for $i = |a| + k$, there can be at most $i - L(a0^k, b0^k) \leq R$ dominant pairs with $j > |b| + k$. Symmetrically, for $j = |b| + k$, there are at most $j - L(a0^k, b0^k) = 0$ dominant pairs with $i > |a| + k$. This yields at most $(R+1) \cdot \ell$ dominant pairs with $|a| < i \leq |a| + \ell$ or $|b| < j \leq |b| + \ell$.

It remains to count dominant pairs with $i = |a| + \ell + \tilde{i}$ and $j = |b| + \ell + \tilde{j}$, with $\tilde{i} \in [|x|], \tilde{j} \in [|y|]$. Here, Lemma 5.8.9 bounds $L(\tilde{x}, \tilde{y}) = L(a0^\ell x[1..\tilde{i}], b0^\ell y[1..\tilde{j}]) = L(a,b) + \ell + L(x[1..\tilde{i}], y[1..\tilde{j}])$. Hence, the dominant pairs of this form are in one-to-one correspondence to the dominant pairs of x, y.

Summing up all dominant pairs, we obtain

$$\begin{aligned} d(x', y') &\leq d(a,b) + (R+1)(|a| + \ell) + d(x,y) \\ &\leq (R+1)(4R + 6S + \ell) + d(x,y), \end{aligned}$$

since $|a| = 2R + 2S$ and Lemma 5.5.3(iv) yields $d(a,b) \leq 2(R+1)(R+2S)$. □

Finally, to pad δ (and later, in Section 5.8.3, Δ), we need the following technical lemma.

Lemma 5.8.11. *Let x, y be arbitrary and μ, ν such that $\nu \geq \mu + |y|$. We define*

$$\begin{aligned} x' &:= 0^\mu\ 1^\nu\ 0^\mu\ x, \\ y' &:= \quad 1^\nu\ 0^\mu\ y. \end{aligned}$$

Then $L(x', y') = \mu + \nu + L(x,y)$ and $d(x', y') = 2\mu + \nu + \#_1(y) + d(x,y)$.

Proof. Note that for any prefix w, z of $0^\mu x, 0^\mu y$, we have

$$L(0^\mu 1^\nu w, 1^\nu z) = \nu + L(w,z), \tag{5.7}$$

by Lemma 5.5.6 (swapping the role of "0"s and "1"s). In particular, we obtain $L(x', y') = \nu + L(0^\mu x, 0^\mu y) = \mu + \nu + L(x,y)$ by greedy prefix matching.

For the second statement, we consider all possible prefixes $\tilde{x} := x'[1..i], \tilde{y} := y'[1..j]$ of x', y' and count how many of them correspond to dominant pairs. Note that these prefixes have to end in the same symbol, since any dominant pair is a matching pair. Recall that $\tilde{x} := x'[1..i], \tilde{y} := y'[1..j]$ gives rise to a dominant pair if and only if $L(\tilde{x}, \tilde{y}) > L(x'[1..i-1], \tilde{y})$ and $L(\tilde{x}, \tilde{y}) > L(\tilde{x}, y'[1..j-1])$.

- $\tilde{x} = 0^\mu 1^\nu w, \tilde{y} = 1^\ell$ (with w non-empty prefix of $0^\mu x$, $\ell \in [\nu]$): These prefixes do not correspond to a dominant pair, since $L(0^\mu 1^\nu w, 1^\ell) = L(0^\mu 1^\ell, 1^\ell) = \ell$ is obtained already by a shorter prefix of $\tilde{x}$.
- $\tilde{x} = 0^\mu 1^\nu w, \tilde{y} = 1^\nu z$ (with w non-empty prefix of $0^\mu x$, z non-empty prefix of $0^\mu y$): These prefixes correspond to a dominant pair if and only if w, z correspond to a dominant pair of $0^\mu x, 0^\mu y$, since by (5.7) we have $L(\tilde{x}, \tilde{y}) = \nu + L(w,z)$. This yields $d(0^\mu x, 0^\mu y)$ dominant pairs, which by Lemma 5.5.1 evaluates to $\mu + d(x,y)$.
- $\tilde{x} = 0^k, \tilde{y} = 1^\mu 0^\ell$ (with $k, \ell \in [\mu]$): Clearly, $L(\tilde{x}, \tilde{y}) = \min\{k, \ell\}$. It follows that $\tilde{x}, \tilde{y}$ corresponds to a dominant pair if and only if $k = \ell$. This yields exactly μ dominant pairs.

- $\tilde{x} = 0^\mu 1^k, \tilde{y} = 1^\ell$ (with $k, \ell \in [\nu]$): Analogously to above, $L(\tilde{x}, \tilde{y}) = \min\{k, \ell\}$, hence this corresponds to a dominant pair if and only if $k = \ell$. This yields exactly ν dominant pairs.
- $\tilde{x} = 0^k, \tilde{y} = 1^\nu 0^\mu z$ (with $k \in [\mu]$, z non-empty prefix of y): We have $L(\tilde{x}, \tilde{y}) = L(0^k, 1^\nu 0^k) = k$, hence these prefixes do not correspond to a dominant pair, since the LCS is already obtained for a shorter prefix of $\tilde{y}$.
- $\tilde{x} = 0^\mu 1^k, \tilde{y} = 1^\nu 0^\mu z$ (with $k \in [\nu]$, z non-empty prefix of y): Since we can either match some "1" of the 1^ν-block in $\tilde{y}$ to a "1" in $\tilde{x}$ (necessarily discarding the initial 0^μ-block of $\tilde{x}$) or delete the complete 1^ν-block, we obtain

$$\begin{aligned} L(0^\mu 1^k, 1^\nu 0^\mu z) &= \max\{L(1^k, 1^\nu 0^\mu z), L(0^\mu 1^k, 0^\mu z)\} \\ &= \max\{k, \mu + L(1^k, z)\} = \max\{k, \mu + \min\{k, \#_1(z)\}\}. \end{aligned}$$

  Consider the case $L(\tilde{x}, \tilde{y}) = L(x'[1..i], y'[1..j]) = k$. Then also $L(x'[1..i], y'[1..(j-1)]) = k$, and hence (i, j) is no dominant pair. If, however, $L(\tilde{x}, \tilde{y}) = \mu + \min\{k, \#_1(z)\}$, then this corresponds to a dominant pair if and only if $k = \#_1(z)$ (and z ends in "1"): if $k > \#_1(z)$, then also $L(x'[1..(i-1)], y'[1..j]) = \mu + \#_1(z)$, if $k < \#_1(z)$, then also $L(x'[1..i], y'[1..(j-1)]) = \mu + k$. Thus, there are exactly $\min\{\nu, \#_1(y)\} = \#_1(y)$ such dominant pairs.

In total, we have counted $\nu + 2\mu + \#_1(y) + d(x, y)$ dominant pairs.

□

We start with the basic building block from Lemma 5.8.9 and then further pad the strings to obtain the desired n, m, Δ, δ, M as follows. Note that the guarantee $|y| \leq |x| = \mathcal{O}(\min\{\Delta, m\})$ is satisfied by Lemma 5.6.8.

Lemma 5.8.12. *Let $(\boldsymbol{\alpha}, \{0,1,2\})$ be a parameter setting satisfying Table 5.2 with $\alpha_L = \alpha_m$. Let (n, x, y) be an instance of $\mathrm{LCS}^\gamma_\leq(\boldsymbol{\alpha}, \{0,1\})$ with $|y| \leq |x| \leq \gamma \cdot \min\{\Delta, m\}$. We set*

$$S = \max\{m, |x|\}, \qquad R = \lfloor d/m \rfloor,$$

to instantiate the basic building block $x' = a0^\ell x = (01)^{R+S}0^\ell x$ and $y' = b0^\ell y = 0^R(01)^S 0^\ell y$ of Lemma 5.8.9 with $\ell := R + S + |x| + |y|$. Moreover, we define $\kappa := \lfloor M/n \rfloor$ and $\tilde{m} := \max\{m, \delta + 2R + 2S + \ell + |x|\}$ to further pad the instance to

$$\begin{aligned} x'' &= 2^\kappa\ 2^\Delta\ 1^{\tilde{m}}\ 0^\delta\ a\ 0^\ell\ x, \\ y'' &= 2^\kappa\ 0^\delta\ 1^{\tilde{m}}\ 0^\delta\ b\ 0^\ell\ y. \end{aligned}$$

Then x'', y'' is an instance of $\mathrm{LCS}^{\gamma'}(\boldsymbol{\alpha}, \{0,1,2\})$ for some constant $\gamma' \geq 1$ and can be computed in time $\mathcal{O}(n)$, together with an integer τ such that $L(x'', y'') = \tau + L(x, y)$.

Proof. We first use the Disjoint Alphabets Lemma and greedy prefix matching to obtain

$$\begin{aligned} L(x'', y'') &= \kappa + L(1^{\tilde{m}}0^\delta x', 0^\delta 1^{\tilde{m}} 0^\delta y') \\ &= \kappa + \tilde{m} + \delta + L(x', y') = \kappa + \tilde{m} + \delta + R + 2S + \ell + L(x, y), \end{aligned} \tag{5.8}$$

where we used Lemma 5.8.11 for the second equality (with the roles of x, y swapped) and Lemma 5.8.9 for the last equality. Observe that x'', y'' and $\tau = \kappa + \tilde{m} + \delta + R + 2S + \ell$ can be computed in time $\mathcal{O}(n)$.

It remains to verify that x, y is an instance of $\mathrm{LCS}^{\gamma'}(\boldsymbol{\alpha}, \{0,1,2\})$ for some $\gamma' \geq 1$. We first observe that $S = \Theta(m)$ by $|x| = \mathcal{O}(m)$ and $R = \mathcal{O}(S)$ by the parameter relation $d \leq Lm \leq m^2$. Since also $|y| = \mathcal{O}(m)$, we conclude $R, S, |x|, |y|, \ell = \mathcal{O}(m)$. Observe that $\kappa = \mathcal{O}(M/n) = \mathcal{O}(m)$ by the relation $M \leq mn$ and $\tilde{m} = \Theta(m)$ (using that $R, S, \ell, |x| = \mathcal{O}(m)$ and the parameter relation $\delta \leq m$). Thus, $|x''| = \kappa + \Delta + \tilde{m} + \delta + 2(R + S) + \ell + |x| = \Theta(m + \Delta) + \mathcal{O}(m) = \Theta(n)$. Similarly, $|y''| = \kappa + \tilde{m} + \delta + R + 2S + \ell + |y| = \Theta(m + \delta) + \mathcal{O}(m) = \Theta(m)$. By (5.8), we also have $L(x'', y'') = \tilde{m} + \mathcal{O}(m) = \Theta(m)$, as desired.

Note that by $\Delta \geq \delta$, $|a| \geq |b|$ and $|x| \geq |y|$, we indeed have $|x''| \geq |y''|$. Furthermore, using the relation $d \leq 2L(\Delta + 1) = \mathcal{O}(m\Delta)$, we obtain $R = \mathcal{O}(d/m) = \mathcal{O}(\Delta)$. By (5.8), we obtain $\Delta(x'', y'') = \Delta + R + (|x| - L(x, y)) = \Delta + \mathcal{O}(\Delta) = \Theta(\Delta)$ since $|x| = \mathcal{O}(\Delta)$. Similarly, (5.8) yields $\delta(x'', y'') = \delta + (|y| - L(x, y)) = \delta + \mathcal{O}(\delta) = \Theta(\delta)$, since $|y| - L(x, y) = \delta(x, y) = \mathcal{O}(\delta)$.

For the dominant pairs, we apply the disjoint alphabets lemma, Lemma 5.5.1 and Lemma 5.8.11 to compute

$$d(x'', y'') = \kappa + d(1^{\tilde{m}}0^{\delta}x', 0^{\delta}1^{\tilde{m}}0^{\delta}y') = \kappa + \tilde{m} + 2\delta + \#_1(x') + d(x', y'). \tag{5.9}$$

Lemma 5.8.10 yields the lower bound $d(x', y') \geq R \cdot S = \Omega(d)$ and the corresponding upper bound

$$d(x', y') \leq (R + 1)(4R + 6S + \ell) + d(x, y) = \mathcal{O}(R \cdot S + d(x, y)) = \mathcal{O}(d).$$

Thus, (5.9) yields $d(x'', y'') = \Theta(d) + \mathcal{O}(m) = \Theta(d)$, where we used that $\kappa, \tilde{m}, \delta, |x'| = \mathcal{O}(m)$ and that $d \geq L = \Omega(m)$ by $\alpha_L = \alpha_m$.

For M, we count $\#_2(x'') = \Delta + \kappa$ and $\#_2(y'') = \kappa$, as well as $\#_0(y''), \#_1(y'') \leq |y''| = \mathcal{O}(m)$, $\#_0(x'') \leq \delta + |x'| = \mathcal{O}(m)$ and $\#_1(x'') \leq \tilde{m} + |x'| = \mathcal{O}(m)$. Thus, $M(x'', y'') = (\Delta + \kappa)\kappa + \mathcal{O}(m^2)$ and by $M \geq L^2/|\Sigma| = \Omega(m^2)$ (since $\alpha_L = \alpha_M$), it suffices to prove that $(\Delta + \kappa)\kappa = \Theta(M)$ to verify $M(x'', y'') = \Theta(M)$. Indeed, we have $\kappa = \Theta(M/n)$, since $M \geq n$. If $\alpha_\Delta < 1$, we have $\alpha_m = \alpha_n = 1$ and $M = \Omega(m^2)$ together with the relation $M \leq mn = \mathcal{O}(m^2)$ implies $M = \Theta(m^2)$. Thus, $\kappa = \Theta(m)$ and hence $(\kappa + \Delta)\kappa = \Theta(m^2) = \Theta(M)$. If $\alpha_\Delta \geq \alpha_m$, then $\alpha_\Delta = 1$ and hence $\Delta + \kappa = \Delta + \mathcal{O}(m) = \Theta(n)$, which implies $(\kappa + \Delta)\kappa = \Theta(n \cdot M/n) = \Theta(M)$. Finally, note that indeed $\Sigma(x'', y'') = \{0, 1, 2\}$. □

Combining Lemma 5.8.12 with Lemma 5.6.8 finally proves Lemma 5.8.7.

5.8.3 Large LCS, Alphabet Size 2

In this section, we study the case that $\alpha_L = \alpha_m$ (and $\alpha_\delta, \alpha_\Delta$ may be small) for the case of binary alphabets, i.e., $\Sigma = \{0, 1\}$. In this regime, Theorem 5.2.3 follows from the following statement (and Lemma 5.3.1).

Lemma 5.8.13. *Let $(\boldsymbol{\alpha}, \{0,1\})$ be a parameter setting satisfying Table 5.2 with $\alpha_L = \alpha_m$. There is a constant $\gamma \geq 1$ such that any algorithm for $\mathrm{LCS}^{\gamma}(\boldsymbol{\alpha}, \{0,1\}$ takes time $\min\{d, \delta\Delta, \delta M/n\}^{1-o(1)}$ unless* OVH *fails.*

We present different constructions for three subcases, that we discuss shortly in the following paragraphs and in detail in the remainder of this section. We hope that this short discussion conveys enough intuition about the "complexity" of the task to make it believable that our lengthy and technical case distinction is indeed necessary.

Case 1: $\alpha_\Delta \le \alpha_m = \alpha_L$. Here, $n = \Theta(m)$ and it follows that *any* binary strings x, y satisfy $M(x,y) = \Theta(m^2)$, so M poses no constraints, in particular there are no constraints on the numbers of "0"s and "1"s in the constructed strings. On the other hand, the potentially small value of Δ renders some of our gadgets useless (e.g., Lemma 5.8.25). Since δ may be small, we use the hardness construction from Section 5.6.2 (for large LCS).

Otherwise, we have $\alpha_\Delta > \alpha_m$ and thus $\Delta = \Theta(n) \gg m$. Note that any string x of length n contains at least $n/2$ "0"s or "1"s, say x contains many "1"s. Then to obtain $\Theta(M)$ matching pairs, y must contain at most $\mathcal{O}(M/n)$ "1"s. Thus, we need to pay close attention to the number of "1"s in the constructed string y. We split the case $\alpha_\Delta > \alpha_m$ into two subcases. *Case 2:* $\alpha_\Delta > \alpha_m = \alpha_L$ *and* $\alpha_\delta \ge \alpha_M - 1$. Here, the constraint on $\#_1(y)$ is stronger than the constraint on δ, and we use the hardness construction from Section 5.6.1 (for small LCS), since it introduces few "1"s. *Case 3:* $\alpha_\Delta > \alpha_m = \alpha_L$ *and* $\alpha_\delta < \alpha_M - 1$. Here, the constraint on δ is stronger than the constraint on $\#_1(y)$, and we use the hardness construction from Section 5.6.2 (for large LCS), since it keeps δ small.

Case 1: $\alpha_\Delta \le \alpha_m = \alpha_L$

Since $n = \Delta + L$, the assumptions $\alpha_L = \alpha_m$ and $\alpha_\Delta \le \alpha_m$ imply $n = \Theta(m)$. Together with the parameter relations $L^2/|\Sigma| \le M \le 2Ln$ and $\Sigma = \{0,1\}$ we obtain $M = \Theta(m^2)$. In particular, in this regime the $\tilde{\mathcal{O}}(\delta\Delta)$ time bound beats $\tilde{\mathcal{O}}(\delta M/n)$, and Lemma 5.8.13 simplifies to the following result.

Lemma 5.8.14. *Let* $(\boldsymbol{\alpha}, \Sigma)$ *be a parameter setting satisfying Table 5.2 with* $\alpha_L = \alpha_m$ *and* $\alpha_\Delta \le \alpha_m$. *There is a constant* $\gamma \ge 1$ *such that any algorithm for* $\mathrm{LCS}^\gamma(\boldsymbol{\alpha}, \Sigma)$ *takes time* $\min\{d, \delta\Delta\}^{1-o(1)}$ *unless* OVH *fails.*

We instantiate the parameters of Lemma 5.8.9 to create a string with the desired number of dominant pairs. The remaining parameters will be padded in an additional construction. Note that the preconditions, specifically the additional guarantee, are satisfied by Lemma 5.6.8.

Lemma 5.8.15. *Let* $(\boldsymbol{\alpha}, \Sigma)$ *be a parameter setting satisfying Table 5.2 with* $\alpha_L = \alpha_m$ *and* $\alpha_\Delta \le \alpha_m$. *Given any instance* (n, x, y) *of* $\mathrm{LCS}^\gamma_\le(\boldsymbol{\alpha}, \{0,1\})$ *with the additional guarantee* $|y| \le |x| \le \gamma \cdot m$, *we can construct an instance* (n, x', y') *of* $\mathrm{LCS}^{\gamma'}_\le(\boldsymbol{\alpha}, \{0,1\})$ *(for some constant* $\gamma' \ge 1$*) and* τ *in time* $\mathcal{O}(n)$ *such that*

(i) $L(x', y') = \tau + L(x, y)$,

(ii) $d(x', y') = \Theta(d)$.

(iii) $|x'| \ge |y'|$.

Proof. We construct x', y' as in Lemma 5.8.9 with $S = \max\{|x|, m\}$, $R = \lceil d/S \rceil$, $\beta = 0$ and $\ell = (R + S + |x| + |y|)$. Note that indeed $|x'| \ge |y'|$ by $|x| \ge |y|$, $|a| \ge |b|$, and $\beta = 0$. The assumption $|x|, |y| = \mathcal{O}(m)$ yields $S = \Theta(m)$. By the parameter relation $d \le 2(\Delta + 1) \cdot L = \mathcal{O}(\Delta \cdot m)$, we also have $R = \mathcal{O}(d/S + 1) = \mathcal{O}(d/m + 1) = \mathcal{O}(\Delta)$. We conclude that $|x'|, |y'| = \mathcal{O}(R + S + |x| + |y|) = \mathcal{O}(m + \Delta) = \mathcal{O}(m)$, since by assumption $\alpha_\Delta \le \alpha_m$. This yields $L(x', y') = \mathcal{O}(m) = \mathcal{O}(L)$ (by the assumption $\alpha_L = \alpha_m$) and $M(x', y') = \mathcal{O}(m^2) = \mathcal{O}(M)$ (note that $M \ge L^2/|\Sigma| = \Omega(m^2)$ by the assumption $\alpha_L = \alpha_m$).

By Lemma 5.8.9, we have $L(x', y') = R + 2S + \ell + L(x, y)$, satisfying (i). This yields $\delta(x', y') = \delta(x, y) = \mathcal{O}(\delta)$ and $\Delta(x', y') = R + \Delta(x, y) = \mathcal{O}(d/m + \Delta) = \mathcal{O}(\Delta)$, by the

parameter relation $d \leq 2L(\Delta+1) = \mathcal{O}(m\Delta)$. For d, we first observe that $\lceil d/S \rceil = \Theta(d/m)$ (by the parameter relation $d \geq L = \Theta(m)$) and $\ell = \mathcal{O}(R + S + |x| + |y|) = \mathcal{O}(m)$. Lemma 5.8.10 yields the lower bound $d(x', y') \geq R \cdot S = \Omega(d/m \cdot m) = \Omega(d)$ as well as the corresponding upper bound $d(x', y') = \mathcal{O}(R \cdot \ell + d(x, y)) = \mathcal{O}(d/m \cdot m + d) = \mathcal{O}(d)$.

These bounds show that (n, x', y') is an instance of $\mathrm{LCS}^{\gamma'}_{\leq}(\boldsymbol{\alpha}, \{0,1\})$ for a sufficiently large constant $\gamma' \geq 1$. □

We use Lemma 5.8.11 to finally pad δ, Δ, and m.

Lemma 5.8.16. *Let x, y, x', y', τ be as given in Lemma 5.8.15. Then, in time $\mathcal{O}(n)$ we can construct an instance x''', y''' of $\mathrm{LCS}^{\gamma''}(\boldsymbol{\alpha}, \{0,1\})$ (for some constant $\gamma'' \geq 1$) and an integer τ' such that $L(x''', y''') = \tau' + L(x', y') = (\tau + \tau') + L(x, y)$.*

Proof. As an intermediate step, let $\tilde{m} := \max\{m, |x'|, |y'|, \Delta\}$ and construct

$$\begin{aligned} x'' &:= 0^{\Delta}\, 1^{\tilde{m}+\Delta}\, 0^{\Delta}\, x', \\ y'' &:= \quad\; 1^{\tilde{m}+\Delta}\, 0^{\Delta}\, y'. \end{aligned}$$

We obtain the final instance as

$$\begin{aligned} x''' &:= \quad 1^{5\tilde{m}+\delta}\, 0^{\delta}\, x'', \\ y''' &:= 0^{\delta}\, 1^{5\tilde{m}+\delta}\, 0^{\delta}\, y''. \end{aligned}$$

Note that by definition of $\tilde{m}$, we have $\tilde{m}+\Delta \geq \Delta+|y'|$ and $\delta+5\tilde{m} \geq \delta+(3\Delta+\tilde{m}+|x'|) = \delta + |x''|$, satisfying the conditions of Lemma 5.8.11. Hence, this lemma yields

$$L(x''', y''') = 5\tilde{m} + 2\delta + L(x'', y'') = 6\tilde{m} + 2\delta + 2\Delta + L(x', y'). \tag{5.10}$$

Clearly, x', y', τ and hence also x'', y'' and $\tau' := 6\tilde{m} + 2\delta + 2\Delta$ can be computed in time $\mathcal{O}(n)$.

We now verify all parameters. Clearly, $\Sigma(x''', y''') = \{0,1\}$. Since by assumption $\alpha_\Delta \leq \alpha_m = 1$, we have $|x'| = \mathcal{O}(n) = \mathcal{O}(m)$, $|y'| = \mathcal{O}(m)$, and $\Delta = \mathcal{O}(m)$. This implies $\tilde{m} = \Theta(m)$ and consequently $|x'''| = 6\tilde{m} + 2\delta + 3\Delta + |x'| = 6\tilde{m} + \mathcal{O}(m) = \Theta(m) = \Theta(n)$ (using $\delta \leq \Delta$ and $\alpha_\Delta \leq \alpha_m = 1$). Similarly, $|y'''| = 6\tilde{m} + 3\delta + 2\Delta + |y'| = 6\tilde{m} + \mathcal{O}(m) = \Theta(m)$. By (5.10), we have $L(x''', y''') = 6\tilde{m} + 2\delta + 2\Delta + L(x', y') = 6\tilde{m} + \mathcal{O}(L) = \Theta(L)$ (using the assumption $\alpha_L = \alpha_m$).

Note that $|x'''| \geq |y'''|$ follows from $\Delta \geq \delta$ and $|x'| \geq |y'|$ (by Lemma 5.8.15(iii)). Hence, (5.10) provides $\Delta(x''', y''') = \Delta + \Delta(x', y') = \Theta(\Delta)$ and $\delta(x''', y''') = \delta + \delta(x', y') = \Theta(\delta)$.

For the number of matching pairs, it holds that $M(x''', y''') = \#_0(x''')\#_0(y''') + \#_1(x''')\#_1(y''') = \Theta(m^2) = \Theta(M)$, where the last bound follows from $M \geq L^2/|\Sigma| = \Omega(m^2)$ and $M \leq 2Ln = \mathcal{O}(m^2)$ by $\alpha_L = \alpha_m = 1$. For the number of dominant pairs, we apply Lemma 5.8.11 to bound

$$\begin{aligned} d(x''', y''') &= 3\delta + 5\tilde{m} + \#_1(x'') + d(x'', y'') \\ &= 3\delta + 5\tilde{m} + (\tilde{m} + \Delta + \#_1(x')) + (3\Delta + \tilde{m} + \#_1(y') + d(x', y')) \\ &= 7\tilde{m} + 3(\delta + \Delta) + \#_1(x') + \#_1(y') + d(x', y') = \Theta(d) + \mathcal{O}(m) = \Theta(d), \end{aligned}$$

where the last two bounds follow from $\tilde{m}, |x'|, |y'|, \delta, \Delta = \mathcal{O}(m)$, $d(x', y') = \Theta(d)$ by Lemma 5.8.15 and the parameter relation $d \geq L = \Omega(m)$. □

Combining Lemmas 5.8.15 and 5.8.16 with Lemma 5.6.8 finally proves Lemma 5.8.14.

Case 2: $\alpha_\Delta > \alpha_m = \alpha_L$ **and** $\alpha_\delta \geq \alpha_M - 1$

In this section, we consider the case where $\alpha_L = \alpha_m < \alpha_\Delta$ and $\alpha_\delta \geq \alpha_M - 1$. In this case, we have $\Delta = \Theta(n) \gg m$ and $M/n = \mathcal{O}(\delta)$. Since $M/n = \mathcal{O}(m) = \mathcal{O}(\Delta)$, the fastest known algorithm runs in time $\tilde{\mathcal{O}}(\min\{d, \delta M/n\}+n)$. Consequently, in this regime Lemma 5.8.13 simplifies to the following statement.

Lemma 5.8.17. *Let* $(\boldsymbol{\alpha}, \{0,1\})$ *be a parameter setting satisfying Table 5.2 with* $\alpha_L = \alpha_m < \alpha_\Delta$ *and* $\alpha_M - 1 \leq \alpha_\delta$*. There is a constant* $\gamma \geq 1$ *such that any algorithm for* $\mathrm{LCS}^\gamma(\boldsymbol{\alpha}, \Sigma)$ *takes time* $\min\{d, \delta M/n\}^{1-o(1)}$ *unless* OVH *fails.*

To obtain this result, it is infeasible to simply pad our hard instances (n, x, y) of $\mathrm{LCS}_\leq(\boldsymbol{\alpha}, \{0,1\})$ to $\mathrm{LCS}(\boldsymbol{\alpha}, \{0,1\})$, since the desired running time bound $\min\{d, \delta M/n\}$ is not monotone. In other words, for $\mathrm{LCS}_\leq(\boldsymbol{\alpha}, \{0,1\})$ we have a lower bound of $\min\{\delta\Delta, \delta m, d\}^{1-o(1)}$ (see Lemma 5.6.8) which can be higher than the running time $\mathcal{O}(n + \delta M/n)$ of our new algorithm (Theorem 5.2.4) and thus a direct padding would violate SETH. Moreover, we even cannot start from an instance as constructed in Lemma 5.6.8, since this would generate too many "1"s. Instead, we use instances of a different parameter setting $\mathrm{LCS}_\leq(\boldsymbol{\alpha}', \{0,1\})$ with $\alpha'_\delta = \alpha'_m$, i.e., we invoke Lemma 5.6.2.

Observation 5.8.18. *Let* $(\boldsymbol{\alpha}, \{0,1\})$ *be a parameter setting satisfying Table 5.2 with* $\alpha_L = \alpha_m < \alpha_\Delta$ *and* $\alpha_M - 1 \leq \alpha_\delta$*. Then* $\boldsymbol{\alpha}' := \boldsymbol{\alpha}'(\boldsymbol{\alpha})$ *defined by*

$$\begin{aligned} \alpha'_d &= \min\{\alpha_d, \alpha_\delta + \alpha_M - 1\}, & \alpha'_M &= 2\alpha'_L, \\ \alpha'_d - \alpha'_L &= \min\{\alpha_M - 1, \alpha_d/2\}, & \alpha'_\Delta &= 1, \\ \alpha'_m &= \alpha'_\delta = \alpha'_L, & \alpha'_\Sigma &= 0, \end{aligned}$$

yields a parameter setting $(\boldsymbol{\alpha}', \{0,1\})$ *satisfying Table 5.2. The definition of* $\boldsymbol{\alpha}'$ *implies*

$$\alpha'_L = \min\{\alpha_\delta, \max\{\alpha_d - \alpha_M + 1, \alpha_d/2\}\}. \tag{5.11}$$

Moreover, there is some constant $\gamma \geq 1$ *such that no algorithm solves* $\mathrm{LCS}^\gamma_\leq(\boldsymbol{\alpha}', \{0,1\})$ *in time* $n^{\alpha'_d(1-o(1))} = \min\{d, \delta M/n\}^{1-o(1)}$ *unless* OVH *fails. This holds even restricted to instances* (n, x, y) *with* $|x|, |y| \leq \gamma \cdot n^{\alpha'_L} = \mathcal{O}(\min\{\delta, \max\{dn/M, \sqrt{d}\}\})$ *and* $\#_1(y) \leq \gamma \cdot n^{\alpha'_d - \alpha'_L} = \mathcal{O}(\min\{M/n, \sqrt{d}\})$ *satisfying* $L(x, 0^\beta y) = L(x, y)$ *for all* $\beta \geq 0$*.*

Proof. We first prove (5.11). Consider the case that $\alpha_d/2 \leq \alpha_M - 1$. Then $\alpha_d \leq 2(\alpha_M - 1) \leq \alpha_\delta + (\alpha_M - 1)$, where we used the assumption $\alpha_M - 1 \leq \alpha_\delta$. Thus, $\alpha'_d = \alpha_d$ and by definition

$$\alpha'_L = \alpha'_d - \min\{\alpha_M - 1, \alpha_d/2\} = \alpha_d - \alpha_d/2 = \alpha_d/2.$$

From $\alpha_d/2 \leq \alpha_M - 1$, it follows that $\alpha_d - \alpha_M + 1 \leq \alpha_d/2$ and $\alpha_d/2 \leq \alpha_M - 1 \leq \alpha_\delta$, hence $\alpha'_L = \alpha_d/2 = \min\{\alpha_\delta, \max\{\alpha_d - \alpha_M + 1, \alpha_d/2\}\}$, as desired.

Consider the remaining case that $\alpha_d/2 > \alpha_M - 1$. Then by definition

$$\alpha'_L = \alpha'_d - (\alpha_M - 1) = \min\{\alpha_d - \alpha_M + 1, \alpha_\delta\}.$$

Since $\alpha_d/2 > \alpha_M - 1$ implies $\alpha_d - \alpha_M + 1 \geq \alpha_d/2$, this indeed yields

$$\alpha'_L = \min\{\max\{\alpha_d - \alpha_M + 1, \alpha_d/2\}, \alpha_\delta\},$$

as desired, concluding the proof of (5.11).

Checking all constraints from Table 5.2 is straight-forward, except for the inequalities $0 \leq \alpha'_L \leq 1$ and $\alpha'_d \leq 2\alpha'_L$. From (5.11), $0 \leq \alpha'_L \leq 1$ follows immediately by the parameter relations $\alpha_\delta, \alpha_d \geq 0$ and $\alpha_\delta \leq \alpha_m \leq 1$. For the other inequality, note that $\alpha'_d \leq 2\alpha'_L$ is equivalent to $\min\{\alpha_M - 1, \alpha_d/2\} = \alpha'_d - \alpha'_L \leq \alpha'_L = \min\{\alpha_\delta, \max\{\alpha_d - \alpha_M + 1, \alpha_d/2\}\}$, which directly follows from the assumption $\alpha_M - 1 \leq \alpha_\delta$ and the trivial fact that $\alpha_d/2 \leq \max\{\alpha_d - \alpha_M + 1, \alpha_d/2\}$.

The last statement directly follows from Lemma 5.6.2. □

It remains to pad strings x, y of $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha}', \{0,1\})$ to $\mathrm{LCS}(\boldsymbol{\alpha}, \{0,1\})$. The first step is the following construction which pads δ and d.

Lemma 5.8.19. *Let $(\boldsymbol{\alpha}, \{0,1\})$ be a parameter setting satisfying Table 5.2 with $\alpha_L = \alpha_m < \alpha_\Delta$ and $\alpha_M - 1 \leq \alpha_\delta$, and define $\boldsymbol{\alpha}'$ as in Observation 5.8.18. Let (n, x, y) be an instance of $\mathrm{LCS}^{\gamma}_{\leq}(\boldsymbol{\alpha}', \{0,1\})$ with $|x|, |y| \leq \gamma \cdot \min\{\delta, \max\{dn/M, \sqrt{d}\}\}$ and $\#_1(y) \leq \gamma \cdot \min\{M/n, \sqrt{d}\}$ satisfying $L(x, 0^\beta y) = L(x, y)$ for all $\beta \geq 0$. We set*

$$S = \lfloor \min\{M/n, \sqrt{d}\} \rfloor, \qquad R = \lfloor d/S \rfloor, \qquad \ell = |x| + |y| + R + S, \qquad \beta = \delta,$$

and define, as in Lemma 5.8.9,

$$\begin{aligned} x' &:= a\, 0^\ell\, x &&= (01)^{R+S}\, 0^\ell\, x, \\ y' &:= 0^\beta\, b\, 0^\ell\, y &&= 0^\beta\, 0^R (01)^S\, 0^\ell\, y. \end{aligned}$$

Then x', y' is an instance of $\mathrm{LCS}^{\gamma'}_{\leq}(\boldsymbol{\alpha}, \{0,1\})$ (for some $\gamma' \geq 1$) with the additional guarantees that

(i) $|x'|, |y'| = \mathcal{O}(m)$,

(ii) $\#_1(y') = \mathcal{O}(M/n)$ *and* $\#_1(y') \cdot |b0^\ell y| = \mathcal{O}(d)$.

(iii) $d(x', y') = \Theta(d)$,

(iv) $|y'| - L(x', y') = \Theta(\delta)$,

(v) In time $\mathcal{O}(n)$ we can compute a number τ such that $L(x', y') = \tau + L(x, y)$.

Proof. Note that $S \leq \sqrt{d}$ implies that $S \leq R$. Note also that by assumption, $|x|, |y| = \mathcal{O}(\min\{\delta, R\})$ and hence $R, S, \ell, |x|, |y| = \mathcal{O}(R)$. Furthermore, $R = \Theta(d/S) = \Theta(d/Mn + \sqrt{d}) = \mathcal{O}(m)$, where the first bound follows from $\alpha_d \geq 0$ and $\alpha_M \geq 1$and the second follows from the parameter relations $d/Mn \leq 5L = \mathcal{O}(m)$ and $d \leq Lm \leq m^2$. Hence $|x'| = \mathcal{O}(R) = \mathcal{O}(m)$. Additionally, $\tilde{\delta} = \Theta(\delta)$ and thus $|y'| = \mathcal{O}(\delta + m) = \mathcal{O}(m)$ by $\delta \leq m$; thus we have proven (i). This also implies $L(x', y') \leq \mathcal{O}(m) = \mathcal{O}(L)$ since $\alpha_L = \alpha_m$.

Note that $\#_1(y) = S + \#_1(y) = \mathcal{O}(S)$ by definition of S and assumption on $\#_1(y)$. We compute $d(x', y') \leq 5L(x', y') \cdot \#_1(y') \leq 5|x'| \cdot \#_1(y') = \mathcal{O}(R \cdot S) = \mathcal{O}(d)$. Furthermore, we obtain by Observation 5.5.2 and Lemma 5.5.3 that $d(x', y') \geq d(a, 0^\beta b) \geq R \cdot S = \Omega(d)$. Note that in particular $\#_1(y') = \mathcal{O}(S) = \mathcal{O}(M/n)$ and $\#_1(y') \cdot |b0^\ell y| = \mathcal{O}(S \cdot R) = \mathcal{O}(d)$, proving (ii). The bound $M(x', y') \leq \mathcal{O}(m^2) = \mathcal{O}(M)$ trivially follows from $|x'|, |y'| = \mathcal{O}(m)$ and $M \geq L^2/|\Sigma| = \Omega(m^2)$ since $\alpha_L = \alpha_m$.

By Lemma 5.8.9, we have $L(x', y') = R + 2S + \ell + L(x, y)$, which immediately yields (v). Moreover, we obtain $\delta(x', y') = \delta + \delta(x, y) = \Theta(\delta)$, since $\delta(x, y) \leq |x| = \mathcal{O}(\delta)$, proving (iv). Finally, $\Delta(x', y') \leq |x'| = \mathcal{O}(m) = \mathcal{O}(\Delta)$ by the assumption $\alpha_\Delta > \alpha_m$. □

To finally pad all remaining parameters, we first prepare the following technical tool.

Lemma 5.8.20. *Let $x = 1^\kappa 0^\mu w$ and $y = 0^\mu 0^\nu z$ with $\mu > |z|$. Then it holds that $L(x,y) = \mu + L(w, 0^\nu z)$, as well as $d(x,y) \geq d(w, 0^\nu z)$ and $d(x,y) \leq \min\{\kappa, |z|\} + \mu + d(1^\kappa 0^\mu, z) + d(w, 0^\nu z)$.*

Proof. By Lemma 5.5.6, we have that $L(x,y) = \mu + L(w, 0^\nu z)$.

This immediately shows that $d(x,y) \geq d(w, 0^\nu z)$, since the above statement implies, for any prefixes $\tilde{w}, \tilde{z}$ of $w, 0^\nu z$, that $L(1^\kappa 0^\mu \tilde{w}, 0^\mu \tilde{z}) = \mu + L(\tilde{w}, \tilde{z})$ and hence any k-dominant pair (i,j) of w and $0^\nu z$ gives rise to a $(\mu+k)$-dominant pair $(\kappa+\mu+i, \mu+j)$ of x and y.

For the upper bound, we count the number of prefixes $\tilde{x}, \tilde{y}$ of x, y corresponding to dominant pairs. Note that $\tilde{x}, \tilde{y}$ have to end in the same symbol to be a dominant pair. Consider first the case that $\tilde{x} = 1^k$. Hence we must have $\tilde{y} = 0^\mu 0^\nu \tilde{z}$ for some prefix $\tilde{z} = z[1..\ell]$ of $\tilde{z}$. Clearly, $L(\tilde{x}, \tilde{y}) = \min\{k, \#_1(\tilde{z})\}$. Hence, $\tilde{x}, \tilde{y}$ corresponds to a dominant pair if and only if $\#_1(\tilde{z}) = \#_1(z[1..\ell]) = k$ and $\#_1(z[1..\ell-1]) < k$, i.e., $\tilde{z}$ is determined by the k-th occurrence of "1" in z. Thus, there can be at most $\min\{\kappa, \#_1(z)\} \leq \min\{\kappa, |z|\}$ such dominant pairs.

Consider the case that $\tilde{x} = 1^\kappa 0^k$ with $k \in [\mu]$. We separately regard the following types of prefixes of y.

- $\tilde{y} = 0^\ell$ with $\ell \in [\mu + \nu]$: By greedy suffix matching, $L(\tilde{x}, \tilde{y}) = L(1^\kappa 0^k, 0^\ell) = \min\{k, \ell\}$, hence as above there can be at most μ dominant pairs, since there are only μ choices for k.

- $\tilde{y} = 0^\mu 0^\nu \tilde{z}$: We have $L(\tilde{x}, \tilde{y}) = \max\{k, L(1^\kappa 0^k, \tilde{z})\}$. To see that this holds, note that the longest common subsequence either includes none of the ones of 1^κ of $\tilde{y}$, in which case 0^k is the LCS of $\tilde{y}$ and $\tilde{x}$, or otherwise it matches at least one 1 in $\tilde{y}$, which means that the LCS deletes all "0"s preceding the first "1" in $\tilde{y}$, i.e., the whole $0^{\mu+\nu}$ block in y'.

  If $L(\tilde{x}, \tilde{y}) = k$, then $\tilde{x}, \tilde{y}$ cannot correspond to a dominant pair since already the prefix 0^k of $\tilde{y}$ satisfies $L(\tilde{x}, 0^k) = k = L(\tilde{x}, \tilde{y})$. Hence $\tilde{x}, \tilde{y}$ can only correspond to a dominant pair if $L(\tilde{x}, \tilde{y}) = L(1^\kappa 0^k, \tilde{z})$ and hence $1^\kappa 0^k, \tilde{z}$ correspond to a dominant pair of $1^\kappa 0^\mu$, z. This yields at most $d(1^\kappa 0^\mu, z)$ dominant pairs.

Finally, consider the case that $\tilde{x} = 1^\kappa 0^\mu \tilde{w}$ with $\tilde{w}$ a prefix of w. There are no dominant pairs for $\tilde{y} = 0^\ell$ with $\ell \in [\mu]$: Already for the prefix $1^\kappa 0^\ell$ of $\tilde{x}$, we have $L(1^\kappa 0^\ell, 0^\ell) = \ell = L(\tilde{x}, \tilde{y})$, hence these prefixes cannot correspond to dominant pairs. It remains to consider $\tilde{y} = 0^\mu \tilde{z}$ for a prefix $\tilde{z}$ of $0^\nu z$. Again by Lemma 5.5.6, we have $L(\tilde{x}, \tilde{y}) = \mu + L(\tilde{w}, \tilde{z})$ and hence such dominant pairs are in one-to-one correspondence with the dominant pairs of w and $0^\nu z$. This yields at most $d(w, 0^\nu z)$ further dominant pairs.

By summing up over the three cases, we conclude that there are at most $\min\{\kappa, |z|\} + \mu + d(1^\kappa 0^\mu, z) + d(w, 0^\nu z)$ dominant pairs. □

We can finally pad to $\mathrm{LCS}(\boldsymbol{\alpha}, \{0,1\})$.

Lemma 5.8.21. *Let x, y, x', y', τ be as in Lemma 5.8.19. We set $\kappa := \lfloor M/n \rfloor$, $\tilde{\Delta} := \max\{\Delta, |y'|\}$, and $\tilde{m} := \max\{m, |y'|\}$ and define*

$$\begin{aligned} x'' &= 1^{\kappa+\tilde{\Delta}}\, 0^{\tilde{m}}\, x', \\ y'' &= 1^{\kappa} \quad 0^{\tilde{m}}\, y'. \end{aligned}$$

Then x'', y'' is an instance of $\mathrm{LCS}^{\gamma'}(\boldsymbol{\alpha}, \{0,1\})$ for some constant $\gamma' \geq 1$. Moreover, we can compute a number τ' in time $\mathcal{O}(n)$ such that $L(x'', y'') = \tau' + L(x, y)$.

Proof. Note that $\kappa = \mathcal{O}(M/n) = \mathcal{O}(m)$ by the parameter relation $M \leq mn$. Furthermore, Lemma 5.8.19(i) yields $|x'|, |y'| = \mathcal{O}(m)$ and hence $\tilde{m} = \Theta(m)$ and $\tilde{\Delta} = \Theta(\Delta)$ (since $\alpha_m \leq \alpha_\Delta = 1$). Thus, $|x''| = \kappa + \tilde{\Delta} + \tilde{m} + |x'| = \Theta(\Delta) + \mathcal{O}(m) = \Theta(n)$ since $\alpha_\Delta = 1$. Furthermore, $|y''| = \kappa + \tilde{m} + |y'| = \tilde{m} + \mathcal{O}(m) = \Theta(m)$.

Observe that $\tilde{\Delta}$ has been defined such that $|x''| \geq |y''|$. By greedy prefix matching and Lemma 5.8.20, we obtain

$$L(x'', y'') = \kappa + L(1^{\tilde{\Delta}}0^{\tilde{m}}x', 0^{\tilde{m}}y') = \kappa + \tilde{m} + L(x', y'). \tag{5.12}$$

Since $L(x', y') = \tau + L(x, y)$, we satisfy the last claim by setting $\tau' := \kappa + \tilde{m} + \tau$. Moreover, $L(x'', y'') = \tilde{m} + \mathcal{O}(m) = \Theta(m) = \Theta(L)$ since $|x'|, |y'| \leq \mathcal{O}(m)$ and $\alpha_L = \alpha_m$. Furthermore, (5.12) yields $\Delta(x'', y'') = \tilde{\Delta} + (|x'| - L(x', y')) = \Theta(\Delta) + \mathcal{O}(m) = \Theta(\Delta)$ and $\delta(x'', y'') = |y'| - L(x', y') = \Theta(\delta)$ by Lemma 5.8.19(iv).

For the dominant pairs, Lemma 5.5.1 yields $d(x'', y'') = \kappa + d(1^{\tilde{\Delta}}0^{\tilde{m}}x', 0^{\tilde{m}}y')$. To bound the latter term, note that Lemma 5.8.20 yields $d(1^{\tilde{\Delta}}0^{\tilde{m}}x', 0^{\tilde{m}}y') \geq d(x', y') = \Omega(d)$ by Lemma 5.8.19(iii). For the upper bound, we first recall that $y' = 0^{\tilde{\delta}}b0^{\ell}y$ and that $\#_1(y') \cdot |b0^{\ell}y| = \mathcal{O}(d)$ by Lemma 5.8.19(ii). Hence we have $d(1^{\tilde{\Delta}}0^{\tilde{m}}, b0^{\ell}y) \leq 5 \cdot L(1^{\tilde{\Delta}}0^{\tilde{m}}, b0^{\ell}y) \cdot \#_1(b0^{\ell}y) = \mathcal{O}(|b0^{\ell}y| \cdot \#_1(y')) = \mathcal{O}(d)$. We can finally compute, using Lemma 5.8.20,

$$\begin{aligned} d(1^{\tilde{\Delta}}0^{\tilde{m}}x', 0^{\tilde{m}}y') &\leq \min\{\tilde{\Delta}, |y'|\} + \tilde{m} + d(1^{\tilde{\Delta}}0^{\tilde{m}}, b0^{\ell}y) + d(x', y') \\ &\leq |y'| + \tilde{m} + \mathcal{O}(d) + d(x', y') = \mathcal{O}(d), \end{aligned}$$

where the last bound follows from $|y'|, \tilde{m} = \mathcal{O}(m) = \mathcal{O}(d)$ by the relation $d \geq L = \Omega(m)$ (since $\alpha_L = \alpha_m$) and $d(x', y') = \mathcal{O}(d)$ by Lemma 5.8.19(iii).

It remains to count the number of matching pairs. We have $\#_0(x''), \#_0(y'') \leq |x'| + |y'| + \tilde{m} = \mathcal{O}(m)$, as well as $\#_1(x'') = \kappa + \tilde{\Delta} + \#_1(x') = \Theta(\Delta) + \mathcal{O}(m) = \Theta(n)$ (since $\alpha_\Delta = 1$) and $\#_1(y'') = \kappa + \#_1(y') = \kappa + \mathcal{O}(M/n) = \Theta(M/n)$ by Lemma 5.8.19(ii). Thus $M(x'', y'') = \#_1(x'')\#_1(y'') + \#_0(x'')\#_0(y'') = \Theta(M) + \mathcal{O}(m^2) = \Theta(M)$, where the last bound follows from $M \geq L^2/|\Sigma| = \Omega(m^2)$ since $\alpha_L = \alpha_m$. □

By combining Lemmas 5.8.19 and 5.8.21 with Observation 5.8.18, we finally obtain Lemma 5.8.17.

Case 3: $\alpha_\Delta > \alpha_m = \alpha_L$ and $\alpha_\delta \leq \alpha_M - 1$

We consider the final case, where $\alpha_L = \alpha_m < \alpha_\Delta$ and $\alpha_\delta \leq \alpha_M - 1$. Here, we have $\Delta = \Theta(n) \gg m$ and $\delta = \mathcal{O}(M/n)$. Since $M/n = \mathcal{O}(m) = \mathcal{O}(\Delta)$, the fastest known algorithm runs in time $\tilde{\mathcal{O}}(\min\{d, \delta M/n\} + n)$. Consequently, in this regime Lemma 5.8.13 simplifies to the following statement.

Lemma 5.8.22. *Let $(\boldsymbol{\alpha}, \{0,1\})$ be a parameter setting satisfying Table 5.2 with $\alpha_L = \alpha_m < \alpha_\Delta$ and $\alpha_\delta \leq \alpha_M - 1$. There is a constant $\gamma \geq 1$ such that any algorithm for $\mathrm{LCS}^\gamma(\boldsymbol{\alpha}, \Sigma)$ takes time $\min\{d, \delta M/n\}^{1-o(1)}$ unless* OVH *fails.*

As in the previous case, to prove this result we cannot simply pad instances of $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha}, \{0,1\})$ to $\mathrm{LCS}(\boldsymbol{\alpha}, \{0,1\})$, since the desired running time bound $\min\{d, \delta M/n\}$ is not monotone. Instead, we start from instances of a suitably chosen different parameter setting $\mathrm{LCS}_{\leq}(\boldsymbol{\alpha}', \{0,1\})$ with $\alpha'_L = \alpha'_m$, i.e., we invoke Lemma 5.6.8.

Observation 5.8.23. *Let $(\boldsymbol{\alpha}, \{0,1\})$ be a non-trivial parameter setting satisfying Table 5.2 with $\alpha_L = \alpha_m < \alpha_\Delta$ and $\alpha_\delta \leq \alpha_M - 1$. Define $\boldsymbol{\alpha}' := \boldsymbol{\alpha}'(\boldsymbol{\alpha})$ by*

$$\begin{aligned} \alpha'_\delta &= \min\{\alpha_\delta, \alpha_d/2\}, & \alpha'_M &= 1 + \alpha'_m, \\ \alpha'_L = \alpha'_m &= \min\{\alpha_M - 1, \alpha_d - \alpha'_\delta\}, & \alpha'_\Delta &= 1, \\ \alpha'_d &= \min\{\alpha_\delta + \alpha_M - 1, \alpha_d\}, & \alpha'_\Sigma &= 0, \end{aligned}$$

Then the parameter setting $(\boldsymbol{\alpha}', \{0,1\})$ satisfies Table 5.2. Furthermore, there is some constant $\gamma \geq 1$ such that no algorithm solves $\mathrm{LCS}^\gamma_\leq(\boldsymbol{\alpha}', \{0,1\})$ in time $n^{\alpha'_d(1-o(1))} = \min\{d, \delta M/n\}^{1-o(1)}$ unless OVH *fails. This holds even restricted to instances (n, x, y) with $|x|, |y| \leq \gamma \cdot n^{\alpha'_m} = \mathcal{O}(\min\{M/n, \max\{d/\delta, \sqrt{d}\}\})$.*

Proof. We only discuss the inequalities from Table 5.2 that are not straight-forward to verify. To see $\alpha'_\delta \leq \alpha'_m$, note that $\alpha_\delta \leq \alpha_M - 1$ by assumption and $\alpha_d/2 = \alpha_d - \alpha_d/2 \leq \alpha_d - \alpha'_\delta$. The inequality $\alpha'_L \leq \alpha'_d$ follows from $\alpha_M - 1 \leq \alpha_M - 1 + \alpha_\delta$ and $\alpha_d - \alpha'_\delta \leq \alpha_d$. Furthermore, $\alpha'_d \leq 2\alpha'_L$ follows from $\alpha_\delta + \alpha_M - 1 \leq 2(\alpha_M - 1)$ (by assumption) and $\alpha_d = 2(\alpha_d - \alpha_d/2) \leq 2(\alpha_d - \alpha'_\delta)$. From $\alpha'_d \leq 2\alpha'_L$ and $\alpha'_L = \alpha'_m$ we also obtain $\alpha'_M = 1 + \alpha'_m = 1 + \alpha'_L \geq 1 + \alpha'_d - \alpha'_L$, which corresponds to the parameter relation $M \geq nd/(5L)$ that only holds for $\Sigma = \{0,1\}$. Finally, $\alpha'_M \geq \alpha'_d$ follows from $\alpha'_M = 1 + \alpha'_m = \min\{\alpha_M, 1 + \alpha_d - \alpha'_\delta\} \geq \min\{\alpha_M, \alpha_d\}$ by $\alpha'_\delta \leq \alpha_\delta \leq 1$ and similarly $\alpha'_d = \min\{\alpha_\delta + \alpha_M - 1, \alpha_d\} \leq \min\{\alpha_M, \alpha_d\}$.

Lemma 5.6.8 shows that some $\gamma \geq 1$ exists such that $\mathrm{LCS}^\gamma_\leq(\boldsymbol{\alpha}', \{0,1\})$ cannot be solved in time $\min\{n^{\alpha'_d}, n^{\alpha'_\delta + \alpha'_m}, n^{\alpha'_\delta + \alpha'_\Delta}\}^{(1-o(1))}$, even restricted to instances (n, x, y) with $|x|, |y| \leq \gamma \cdot n^{\alpha'_m} = \mathcal{O}(\min\{M/n, \max\{d/\delta, \sqrt{d}\}\})$. We simplify the running time bound by noting that $\alpha'_\Delta = 1 \geq \alpha'_m$, so that $n^{\alpha'_\delta + \alpha'_m} \leq n^{\alpha'_\delta + \alpha'_\Delta}$. Moreover, we have $\alpha'_\delta + \alpha'_m = \alpha'_d = \min\{\alpha_\delta + \alpha_M - 1, \alpha_d\}$. Indeed, if $\alpha_\delta \leq \alpha_d/2$, we have $\alpha'_\delta = \alpha_\delta$ and hence $\alpha'_\delta + \alpha'_m = \min\{\alpha_\delta + \alpha_M - 1, \alpha_d\} = \alpha'_d$. Otherwise, $\alpha_d/2 < \alpha_\delta \leq \alpha_M - 1$, forcing $\alpha'_\delta = \alpha_d/2$ and $\alpha'_m = \alpha_d/2$, which yields $\alpha'_\delta + \alpha'_m = \alpha_d = \min\{\alpha_\delta + \alpha_M - 1, \alpha_d\} = \alpha'_d$. Thus, $\min\{\alpha'_d, \alpha'_\delta + \alpha'_m, \alpha'_\delta + \alpha'_\Delta\} = \alpha'_d$ and the lower bound simplifies to $n^{\alpha'_d(1-o(1))} = \min\{\delta M/n, d\}^{1-o(1)}$. □

In this section, to pad the number of dominant pairs, we will construct instances $x' = a0^\ell x = (01)^{R+S}0^\ell x$, $y' = b0^\ell y = 0^R(01)^S 0^\ell y$, where we choose R, S proportional to $n^{\alpha_R}, n^{\alpha_S}$ with

$$\alpha_S := \min\{\alpha_M - 1, \max\{\alpha_d - \alpha_\delta, \alpha_d/2\}\}, \qquad \alpha_R := \alpha_d - \alpha_S.$$

Note that the use of α_S, α_R is a slight abuse of notation, since R, S are not actual input parameters, but α_S, α_R depend only on $\boldsymbol{\alpha}$. We will later set $R = c \cdot n^{\alpha_R}$, $S = c' \cdot n^{\alpha_S}$ with suitably large constants c, c'. However, depending on whether $\alpha_S \leq \alpha_R$ or $\alpha_S > \alpha_R$, we will extend and analyze the basic construction differently. We start with the simpler case of $\alpha_S \leq \alpha_R$.

Lemma 5.8.24 (Construction for $\alpha_S \leq \alpha_R$)**.** *Let $(\boldsymbol{\alpha}, \{0,1\})$ be a parameter setting satisfying Table 5.2 with $\alpha_L = \alpha_m < \alpha_\Delta$, $\alpha_\delta \leq \alpha_M - 1$, and $\alpha_S \leq \alpha_R$. We define $\boldsymbol{\alpha}'$ as in Observation 5.8.23. Given an instance (n, x, y) of $\mathrm{LCS}^\gamma_\leq(\boldsymbol{\alpha}', \{0,1\})$ with $|y| \leq |x| = \mathcal{O}(\min\{M/n, \max\{d/\delta, \sqrt{d}\}\})$, we use the parameters*

$$\begin{aligned} S &= \max\{|x|, \lfloor n^{\alpha_S} \rfloor\}, & R &= \lfloor d/S \rfloor, \\ \ell &= R + S + |x| + |y|, & \beta &= \delta, \end{aligned}$$

to instantiate the strings $x' = a0^\ell x = (01)^{R+S}0^\ell x$ *and* $y' = 0^\beta b0^\ell y = 0^\beta 0^R(01)^S 0^\ell y$ *as in Lemma 5.8.9. We furthermore set* $\tilde{m} := \max\{m, \delta + R + 2S + \ell + |y|\}$ *and* $\kappa := \lfloor M/n \rfloor$ *and define*

$$\begin{aligned} x'' &:= 1^{\Delta+\kappa}\, 0^{\tilde{m}}\, x' &&= 1^{\Delta+\kappa}\, 0^{\tilde{m}} \quad a\, 0^\ell\, x &&= 1^{\Delta+\kappa}\, 0^{\tilde{m}} \quad (01)^{R+S}\, 0^\ell\, x, \\ y'' &:= 1^{\kappa} \quad 0^{\tilde{m}}\, y' &&= 1^{\kappa} \quad 0^{\tilde{m}}\, 0^\delta\, b\, 0^\ell\, y &&= 1^{\kappa} \quad 0^{\tilde{m}}\, 0^\delta\, 0^R(01)^S\, 0^\ell\, y. \end{aligned}$$

Then x'', y'' *is an instance of* $\mathrm{LCS}^{\gamma'}(\boldsymbol{\alpha}, \{0,1\})$ *for some constant* $\gamma' \geq 1$ *that can be constructed in time* $\mathcal{O}(n)$ *together with some integer* τ *such that* $L(x'', y'') = \tau + L(x, y)$.

Proof. By greedy prefix matching and Lemma 5.5.6, we obtain

$$L(x'', y'') = \kappa + L(1^\Delta 0^{\tilde{m}} x', 0^{\tilde{m}} y') = \kappa + \tilde{m} + L(x', y') = \kappa + \tilde{m} + R + 2S + \ell + L(x, y), \quad (5.13)$$

where the last equality follows from Lemma 5.8.9, which applies since $S \geq |x|$. Clearly, x'', y'' and $\tau = \kappa + \tilde{m} + R + 2S + \ell$ can be computed in time $\mathcal{O}(n)$.

Let us verify that x, y is an instance of $\mathrm{LCS}^{\gamma'}(\boldsymbol{\alpha}, \{0,1\})$ for some $\gamma' \geq 1$. By assumption, $|x|, |y| = \mathcal{O}(n^{\alpha_S})$, and hence $S = \Theta(n^{\alpha_S}) = \Theta(\min\{M/n, \max\{d/\delta, \sqrt{d}\}\})$. Thus, $R = \Theta(d/S) = \Theta(n^{\alpha_R}) = \mathcal{O}(dn/M + \min\{\delta, \sqrt{d}\}) = \mathcal{O}(m)$, where the last bound follows from the parameter relations $M \geq nd/(5L) = \Omega(nd/m)$ (since $\alpha_L = \alpha_m$) and $\delta \leq m$. By assumption $\alpha_S \leq \alpha_R$, we have $S = \mathcal{O}(R) = \mathcal{O}(m)$. Furthermore, we have $\kappa = \mathcal{O}(M/n) = \mathcal{O}(m)$ by the relation $M \leq mn$ and $\tilde{m} = \Theta(m)$ by $R, S, |x|, |y|, \ell = \mathcal{O}(m)$ and $\delta \leq m$. Thus, $|x''| = \kappa + \Delta + \tilde{m} + 2(R + S) + \ell + |x| = \Theta(\Delta + m) = \Theta(n)$ and $|y''| = \kappa + \tilde{m} + \delta + R + 2S + \ell + |y| = \Theta(m)$. By (5.13), we also conclude that $L(x'', y'') = \tilde{m} + \mathcal{O}(m) = \Theta(m) = \Theta(L)$ by the assumption $\alpha_L = \alpha_m$.

Note that by $\Delta \geq \delta$, $|a| \geq |b|$ and $|x| \geq |y|$, we have $|x''| \geq |y''|$. Hence by (5.13), $\Delta(x'', y'') = \Delta + R + (|x| - L(x, y)) = \Theta(\Delta)$, since $R, |x| = \mathcal{O}(m) = \mathcal{O}(\Delta)$ by $\alpha_\Delta > \alpha_m$. Likewise, (5.13) yields $\delta(x'', y'') = \delta + (|y| - L(x, y)) = \delta + \delta(x, y) = \Theta(\delta)$ since $\delta(x, y) = \mathcal{O}(\delta)$ by $\delta(x, y) = \mathcal{O}(n^{\alpha'_\delta})$ and $\alpha'_\delta \leq \alpha_\delta$.

For the dominant pairs, we first compute

$$d(1^\Delta 0^{\tilde{m}} x', 0^{\tilde{m}} y') \geq d(x', y') \geq d(a, 0^\delta b) \geq R \cdot S = \Omega(d),$$

using Lemma 5.8.20, Observation 5.5.2, and Lemma 5.5.3. For a corresponding upper bound, we use Lemma 5.8.20 to obtain

$$d(1^\Delta 0^{\tilde{m}} x', 0^{\tilde{m}} y') = d(1^\Delta 0^{\tilde{m}} x', 0^{\tilde{m}} 0^\delta b 0^\ell y) \leq |y'| + \tilde{m} + d(1^\Delta 0^{\tilde{m}}, b0^\ell y) + d(x', y').$$

By Lemma 5.4.8, we have $d(1^\Delta 0^{\tilde{m}}, b0^\ell y) \leq 5 \cdot L(1^\Delta 0^{\tilde{m}}, b0^\ell y) \cdot \#_1(b0^\ell y) = \mathcal{O}(|b0^\ell y| \cdot (S + |y|)) = \mathcal{O}(R \cdot S) = \mathcal{O}(d)$. Since $|y'| + \tilde{m} = \mathcal{O}(m) = \mathcal{O}(d)$ (using $d \geq L = \Omega(m)$ since $\alpha_L = \alpha_m$) and $d(x', y') \leq 5 \cdot L(x', y') \cdot \#_1(y') = \mathcal{O}(|x'| \cdot \#_1(y')) = \mathcal{O}(R \cdot S) = \mathcal{O}(d)$, we conclude that $d(1^\Delta 0^{\tilde{m}} x', 0^{\tilde{m}} y') = \Theta(d)$. Finally, Lemma 5.5.1 yields $d(x'', y'') = \kappa + d(1^\Delta 0^{\tilde{m}} x', 0^{\tilde{m}} y') = \Theta(d) + \mathcal{O}(m) = \Theta(d)$, as desired.

It remains to count the matching pairs. Note that $\#_0(x''), \#_0(y'') = \mathcal{O}(\tilde{m} + |x'| + |y'|) = \mathcal{O}(m)$. Furthermore $\#_1(x'') = \Delta + \kappa + \#_1(x') = \Theta(\Delta) + \mathcal{O}(m) = \Theta(n)$ (since $\alpha_\Delta > \alpha_m$ implies $\alpha_\Delta = 1$) and $\#_1(y'') = \kappa + S + \#_1(y) = \Theta(\kappa) = \Theta(M/n)$, where we used $S, |y| = \mathcal{O}(M/n)$ and $\kappa = \Theta(M/n)$ (since $M \geq n$). Thus, $M(x'', y'') = \#_1(x'')\#_1(y'') + \#_0(x'')\#_1(y'') = \Theta(n \cdot M/n) + \mathcal{O}(m^2) = \Theta(M)$ using that $M \geq L^2/|\Sigma| = \Omega(m^2)$ by $\alpha_L = \alpha_m$. Note that indeed $\Sigma(x'', y'') = \{0, 1\}$. □

Before giving the construction for the case $\alpha_S > \alpha_R$, we present a technical lemma that is similar to the dominant pair reduction technique of Lemma 5.5.8.

Lemma 5.8.25. *Let $x' = a0^\ell x = (01)^{R+S}0^\ell x$, $y' = b0^\ell y = 0^R(01)^S0^\ell y$ be an instance of Lemma 5.8.2 and $R \geq |y| - L(x,y)$. We set $t := R + |y'| + 1$ and define*

$$\begin{aligned} \bar{x} &:= \quad 1^t\, 0^t\, y'\, 0^R\, 1^{t+\Delta}\, 0^t\, x', \\ \bar{y} &:= \qquad\qquad 0^R\, 1^t \quad 0^t\, y'. \end{aligned}$$

Then

(i) $L(\bar{x},\bar{y}) = R + 2t + L(x',y')$,

(ii) $d(\bar{x},\bar{y}) \leq (2t + |y'|)(R+1) + R^2$,

(iii) $d(\bar{x},\bar{y}) \geq R \cdot S$.

Proof. We first prove the following property.

(∗) For any prefixes $\tilde{x},\tilde{y}$ of x',y', we have

$$L(1^t0^ty'\,0^R\,1^{t+\Delta}0^t\tilde{x}, 0^R\,1^t0^t\tilde{y}) = \max\{2t + |\tilde{y}|, R + 2t + L(\tilde{x},\tilde{y})\}.$$

Note that the lower bound immediately follows from either matching $1^t0^t\tilde{y}$ with $1^t0^ty'$, or by matching $0^R1^t0^t\tilde{y}$ with $0^R1^{t+\Delta}0^t\tilde{x}$. For the upper bound, fix an LCS and observe that it cannot match symbols in the 1^t-prefix of $\bar{x}$ and symbols in the 0^R-prefix of $\bar{y}$, so at least one of the two prefixes stays unmatched. Thus,

$$\begin{aligned} &L(1^t0^ty'\,0^R\,1^{t+\Delta}0^t\tilde{x}, 0^R\,1^t0^t\tilde{y}) \\ &\quad\leq \max\{L(0^ty'\,0^R\,1^{t+\Delta}0^t\tilde{x}, 0^R\,1^t0^t\tilde{y}),\, L(1^t0^ty'\,0^R\,1^{t+\Delta}0^t\tilde{x}, 1^t0^t\tilde{y})\} \\ &\quad= \max\{R + L(0^{t-R}y'\,0^R\,1^{t+\Delta}0^t\tilde{x},\, 1^t0^t\tilde{y}),\, 2t + |\tilde{y}|\}, \end{aligned}$$

where the second line follows from greedy prefix matching. Setting $\hat{x} := 0^{t-R}y'\,0^R\,1^{t+\Delta}0^t\tilde{x}$ and $\hat{y} := 1^t0^t\tilde{y}$, it remains to provide the upper bound $R + L(\hat{x},\hat{y}) \leq \max\{2t + |\tilde{y}|, R + 2t + L(\tilde{x},\tilde{y})\}$ to prove (∗). Assume that an LCS z of $\hat{x},\hat{y}$ matches less than $t-R$ symbols of the 1^t-prefix of $\hat{y}$. Then $|z| \leq t - R + L(\hat{x}, 0^t\tilde{y}) \leq 2t - R + |\tilde{y}|$, yielding $R + L(\hat{x},\hat{y}) \leq 2t + |\tilde{y}|$. Hence, assume instead that at least $t - R$ symbols of the 1^t-prefix of $\hat{y}$ are matched. Since the number of "1"s in the prefix $0^{t-R}y'0^R$ of $\hat{x}$ is only $\#_1(y') \leq |y'| < t - R$, all zeroes of this prefix have to be deleted, resulting in

$$\begin{aligned} L(\hat{x},\hat{y}) = |z| &\leq L(1^{\#_1(y')+t+\Delta}0^t\tilde{x}, 1^t0^t\tilde{y}) \\ &= t + L(1^{\#_1(y')+R+\Delta}0^t\tilde{x}, 0^t\tilde{y}) \\ &= 2t + L(\tilde{x},\tilde{y}), \end{aligned}$$

where the second line follows from greedy prefix matching and the third follows from Lemma 5.5.6. Thus, we have verified $R + L(\hat{x},\hat{y}) \leq \max\{2t + |\tilde{y}|, R + 2t + L(\tilde{x},\tilde{y})\}$, which implies (∗).

As an immediate consequence, (∗) yields $L(\bar{x},\bar{y}) = \max\{2t + |y'|, R + 2t + L(x',y')\} = R + 2t + L(x',y')$ since $R \geq |y| - L(x,y) = |y'| - L(x',y')$ (where the equality follows from Lemma 5.8.2). This proves (i).

For (ii), an application of Lemma 5.5.7 yields $d(\bar{x},\bar{y}) \leq (2t + |y'|)(R+1) + R^2$.

Finally, for (iii) we consider, analogous to Lemma 5.5.3, for any $0 \le s \le S$ and $s \le r \le R + s$, the prefixes $\tilde{x} = (01)^r$ of x' and $\tilde{y} = 0^R(01)^s$ of y'. Then by $(*)$,

$$\begin{aligned} L(1^t0^t y'\, 0^R\, 1^{t+\Delta}0^t\tilde{x}, 0^R\, 1^t0^t\tilde{y}) &= \max\{2t + |\tilde{y}|, R + 2t + L(\tilde{x}, \tilde{y})\} \\ &= \max\{2t + R + 2s, R + 2t + r + s\} = (R + 2t) + r + s, \end{aligned}$$

where we used Lemma 5.5.3$(*)$ for the second equality and $r \ge s$ for the last equality. Analogously to the proof of Lemma 5.5.3(ii), this yields $d(\bar{x}, \bar{y}) \ge R \cdot S$, since any feasible choice of r, s gives rise to a unique dominant pair. □

We can now give the construction for $\alpha_S > \alpha_R$. Recall that

$$\alpha_S := \min\{\alpha_M - 1, \max\{\alpha_d - \alpha_\delta, \alpha_d/2\}\}, \qquad \alpha_R := \alpha_d - \alpha_S.$$

Lemma 5.8.26 (Construction for $\alpha_S > \alpha_R$)**.** *Let $(\boldsymbol{\alpha}, \{0,1\})$ be a parameter setting satisfying Table 5.2 with $\alpha_L = \alpha_m < \alpha_\Delta$, $\alpha_\delta \le \alpha_M - 1$, and $\alpha_S > \alpha_R$. We define $\boldsymbol{\alpha}'$ as in Observation 5.8.23. Given an instance (n, x, y) of $\mathrm{LCS}^{\gamma}_{\le}(\boldsymbol{\alpha}', \{0,1\})$ with $|y| \le |x| = \mathcal{O}(\min\{M/n, \max\{d/\delta, \sqrt{d}\}\})$, we can construct an instance $x^{(4)}, y^{(4)}$ of $\mathrm{LCS}^{\gamma'}(\boldsymbol{\alpha}, \{0,1\})$ (for some constant $\gamma' \ge 1$) in time $\mathcal{O}(n)$ together with some integer τ such that $L(x^{(4)}, y^{(4)}) = \tau + L(x, y)$.*

Proof. We first set $\ell_1 := |x|$ to define

$$\begin{aligned} x^{(1)} &:= \quad 0^{\ell_1}\, x, \\ y^{(1)} &:= 1^\delta\, 0^{\ell_1}\, y, \end{aligned}$$

which pads the parameter δ. For convenience, define $\delta_1 := |y^{(1)}| - L(x^{(1)}, y^{(1)})$. We use the parameters

$$S := \lfloor n^{\alpha_S} \rfloor, \qquad R := \max\{\lfloor n^{\alpha_R} \rfloor, \delta_1\}, \qquad \ell_2 := |x^{(1)}| + |y^{(1)}|,$$

to define, as in Lemma 5.8.2,

$$\begin{aligned} x^{(2)} &:= a\, 0^{\ell_2}\, x^{(1)}, \\ y^{(2)} &:= b\, 0^{\ell_2}\, y^{(1)}. \end{aligned}$$

We then use the dominant pair reduction trick of Lemma 5.8.25, which additionally pads Δ, and define

$$\begin{aligned} x^{(3)} &:= 1^{\ell_3}\, 0^{\ell_3}\, y^{(2)}\, 0^R\, 1^{\ell_3+\Delta}\, 0^{\ell_3}\, x^{(2)}, \\ y^{(3)} &:= \qquad\qquad\quad 0^R\, 1^{\ell_3} \quad 0^{\ell_3}\, y^{(2)}, \end{aligned}$$

where $\ell_3 := R + |y^{(2)}| + 1$. The final instance is then constructed as

$$\begin{aligned} x^{(4)} &= 1^\kappa\, 0^m\, x^{(3)}, \\ y^{(4)} &= 1^\kappa\, 0^m\, y^{(3)}, \end{aligned}$$

where $\kappa := \lfloor M/n \rfloor$.

We first compute

$$\begin{aligned} L(x^{(4)}, y^{(4)}) &= \kappa + m + L(x^{(3)}, y^{(3)}) \\ &= \kappa + m + R + 2\ell_3 + L(x^{(2)}, y^{(2)}) \\ &= \kappa + m + 2R + 2S + \ell_2 + 2\ell_3 + L(x^{(1)}, x^{(1)}) \\ &= \kappa + m + 2R + 2S + \ell_1 + \ell_2 + 2\ell_3 + L(x, y), \end{aligned} \tag{5.14}$$

where we used greedy prefix matching in the first line, Lemma 5.8.25(i) in the second, Lemma 5.8.2 in the third, and Lemma 5.5.6 in the last line. Note that $x^{(4)}, y^{(4)}$, and $\tau := \kappa + m + 2R + 2S + \ell_1 + \ell_2 + 2\ell_3$ can be computed in time $\mathcal{O}(n)$.

It remains to verify that $x^{(4)}, y^{(4)}$ is an instance of $\mathrm{LCS}^{\gamma'}(\boldsymbol{\alpha}, \{0,1\})$ for some $\gamma' \geq 1$. We first observe that $|x|, |y| = \mathcal{O}(n^{\alpha_S})$ and hence $|x^{(1)}|, |y^{(1)}| = \mathcal{O}(n^{\alpha_S} + \delta)$. Note that by definition $S = \Theta(n^{\alpha_S})$.

Assume for contradiction that $\alpha_R < \alpha_\delta$. Note that by definition $\alpha_R = \alpha_d - \alpha_S = \max\{\alpha_d - \alpha_M + 1, \min\{\alpha_\delta, \alpha_d/2\}\}$ and hence $\alpha_R < \alpha_\delta$ only holds if $\alpha_d - \alpha_M + 1 \leq \alpha_R = \alpha_d/2 < \alpha_\delta$. But then $\alpha_d - \alpha_\delta \leq \alpha_d/2 < \alpha_\delta \leq \alpha_M - 1$. This forces $\alpha_S = \alpha_d/2$ by definition, which contradicts the assumption $\alpha_S > \alpha_R$. We therefore obtain $\alpha_R \geq \alpha_\delta$.

Note that $\delta_1 = |y^{(1)}| - L(x^{(1)}, y^{(1)}) = \delta + \delta(x, y)$ by Lemma 5.5.6. Since $\delta(x, y) \leq \mathcal{O}(n^{\alpha'_\delta})$ and $\alpha'_\delta \leq \alpha_\delta$, this yields $\delta_1 = \Theta(\delta)$, and hence $R = \Theta(n^{\alpha_R})$. Thus, $R = \mathcal{O}(S)$, since $\alpha_R < \alpha_S$. It is immediate that $|x^{(2)}|, |y^{(2)}| = \mathcal{O}(R + S + |x^{(1)}| + |y^{(1)}|) = \mathcal{O}(S)$. Furthermore, it follows that $|x^{(3)}| = \Delta + \mathcal{O}(S)$ and $|y^{(3)}| = \mathcal{O}(S)$. Finally $|y^{(4)}| = m + \mathcal{O}(M/n + S) = \Theta(m)$, where the last bound follows from $S = \mathcal{O}(M/n) = \mathcal{O}(m)$ by the parameter relation $M \leq mn$. Likewise, $|x^{(4)}| = m + \Delta + \mathcal{O}(M/n + S) = \Theta(m + \Delta) = \Theta(n)$. Finally, $L(x^{(4)}, y^{(4)}) = m + \mathcal{O}(M/n + S) = \Theta(m) = \Theta(L)$ by (5.14) and $\alpha_L = \alpha_m$.

Since $|x| \geq |y|$, $|a| \geq |b|$ and $\Delta \geq \delta$ it is easy to see that $|x^{(4)}| \geq |y^{(4)}|$. Hence (5.14) yields

$$\Delta(x^{(4)}, y^{(4)}) = 2\ell_3 + |y^{(2)}| + \Delta + R + \Delta(x, y) = \Delta + \mathcal{O}(m) = \Theta(\Delta),$$

since in particular $\Delta(x, y) \leq |x| = \mathcal{O}(m)$. Similarly, $\delta(x^{(4)}, y^{(4)}) = \delta + \delta(x, y) = \Theta(\delta)$ as above.

For the number of dominant pairs, Lemma 5.8.25(iii) yields $d(x^{(3)}, y^{(3)}) \geq R \cdot S = \Omega(d)$. From Lemma 5.8.25(ii), the corresponding upper bound $d(x^{(3)}, y^{(3)}) \leq (2\ell_3 + |y^{(2)}|) \cdot (R + 1) + R^2 = \mathcal{O}(S \cdot R + R^2) = \mathcal{O}(d)$ follows, since $R = \mathcal{O}(S)$ by $\alpha_R < \alpha_S$. Thus, by Lemma 5.5.1 we obtain $d(x^{(4)}, y^{(4)}) = \kappa + m + d(x^{(3)}, y^{(3)}) = \Theta(d) + \mathcal{O}(m) = \Theta(d)$ by $d \geq L = \Omega(m)$ since $\alpha_L = \alpha_m$.

It remains to count the number of matching pairs. We have $\#_1(y^{(4)}) = \kappa + \#_1(y^{(3)}) = \Theta(M/n)$, since $\kappa = \Theta(M/n)$ by the parameter relation $M \geq n$, and $\#_1(y^{(3)}) \leq |y^{(3)}| = \mathcal{O}(S) = \mathcal{O}(M/n)$. Since $|y^{(4)}| = \mathcal{O}(m)$, we have $\#_0(y^{(4)}) = \mathcal{O}(m)$. Note that $\#_1(x^{(4)}) = \kappa + 2\ell_3 + \Delta + |x^{(2)}| + |y^{(2)}| = \Delta + \mathcal{O}(S + \kappa) = \Theta(n)$, since $\alpha_\Delta > \alpha_m$ implies $\alpha_\Delta = 1$. Finally, $\#_0(x^{(4)}) = m + 2\ell_3 + R + \#_0(y^{(2)}) + \#_0(x^{(2)}) = \mathcal{O}(m)$. Thus, we obtain $M(x^{(4)}, y^{(4)}) = \#_1(x^{(4)}) \cdot \#_1(y^{(4)}) + \#_0(x^{(4)}) \cdot \#_0(y^{(4)}) = \Theta(n \cdot M/n) + \mathcal{O}(m^2) = \Theta(M)$ by the relation $M \geq L^2/|\Sigma| = \Omega(m^2)$, since $\alpha_L = \alpha_m$. □

Note that combining Lemmas 5.8.24 and 5.8.26 with Observation 5.8.23 yields Lemma 5.8.22.

Chapter 6

Algorithms

In this chapter, we provide two algorithmic results. The first is a generalization of Hirscherg's $\tilde{\mathcal{O}}(n + \delta L)$-time algorithm for LCS to Edit Distance, which we provide here for completeness (Section 6.1). The second algorithm is a novel LCS algorithm for binary alphabets, which runs particularly fast when the number of matching pairs M and $\delta = m - L$ are small (Section 6.2).

6.1 Generalization of Hirscherg's Algorithm

For completeness, we prove a generalization of the algorithm of Hirschberg [Hir77] from LCS to Edit Distance. Recall that the trivial dynamic programming algorithm computes a table storing all prefix distances $\boldsymbol{\delta}_{\text{Edit}}(x[1..i], y[1..j])$. In contrast, we build a dynamic programming table storing, for any index j and any cost k, the minimal index i with $\boldsymbol{\delta}_{\text{Edit}}(x[1..i], y[1..j]) - c_{\text{del-x}}(i - j) = k$. For some intuition, note that for $i \geq j$ at least $i - j$ symbols in $x[1..i]$ have to be deleted so that the cost $\boldsymbol{\delta}_{\text{Edit}}(x[1..i], y[1..j])$ is at least $c_{\text{del-x}}(i - j)$. Thus, it is reasonable to "normalize" the cost by subtracting $c_{\text{del-x}}(i - j)$. As we will see, the normalized cost is bounded by $\mathcal{O}(m)$ (the length of the smaller of the two strings), which reduces the table size to $\mathcal{O}(m^2)$.

Theorem 6.1.1. *Let the cost parameters $c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}}$ be any positive integers. Then* $\text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$ *can be solved in time $\mathcal{O}((n + m^2) \log |\Sigma|)$ on strings of length n, m with $n \geq m$ over alphabet Σ.*

Note that it is easy to ensure $\Sigma \subseteq [n + m]$ after $\mathcal{O}(n \log(\min\{|\Sigma|, n\}))$ preprocessing.[1] Thus, the running time is at most $\mathcal{O}((n + m^2) \log n) = \tilde{\mathcal{O}}(n + m^2)$, and Theorem 2.1.4 follows from the above theorem and Fact 4.4.2. Our algorithm is designed for the pointer machine model; a slightly better running is achievable on the Word RAM.

Consider strings x and y over alphabet Σ of lengths n and m, respectively, and assume without loss of generality that $n \geq m$. For convenience, we set $\min \emptyset := \infty$. For any index $i \in \{0, \ldots, n\}$ and symbol $\sigma \in \Sigma$, we define

$$\text{Next}^x_{=\sigma}(i) := \min\{i' \mid i < i' \leq n \text{ and } x[i'] = \sigma\},$$
$$\text{Next}^x_{\neq\sigma}(i) := \min\{i' \mid i < i' \leq n \text{ and } x[i'] \neq \sigma\}.$$

We argue that a data structure can be built in $\mathcal{O}(n \log |\Sigma|)$ preprocessing time supporting $\text{Next}^x_{=\sigma}(i)$ and $\text{Next}^x_{\neq\sigma}(i)$ queries in time $\mathcal{O}(\log |\Sigma|)$. A simple solution with worse running time is to precompute all answers to all possible queries $\text{Next}^x_{=\sigma}(i)$ and $\text{Next}^x_{\neq\sigma}(i)$, with $i \in \{0, \ldots, n\}$, $\sigma \in \Sigma$, in time $\mathcal{O}(|\Sigma|n)$ by one scan from $x[n]$ to $x[1]$. To improve

[1] To compress the alphabet we build a balanced binary search tree T whose nodes correspond to Σ (by simply adding all symbols of x and y to T). Then we replace each symbol by its index in some fixed ordering of the nodes of T.

the preprocessing time for $\text{Next}^x_{\neq\sigma}(i)$, note that $\text{Next}^x_{\neq\sigma}(i) = i+1$ for all $\sigma \neq x[i+1]$. Thus, we only have to precompute all values $\text{Next}^x_{\neq x[i+1]}(i)$ for $1 \leq i \leq n$ (which can be done in time $\mathcal{O}(n)$ by one scan from $x[n]$ to $x[1]$), then $\text{Next}^x_{\neq\sigma}(i)$ can be queried in time $\mathcal{O}(1)$. For $\text{Next}^x_{=\sigma}(i)$, for any $i \in \{0,\dots,n\}$ we build a dictionary D_i storing $\text{Next}^x_{=\sigma}(i)$ for each $\sigma \in \Sigma$. Note that D_{i-1} and D_i differ only for the symbol $x[i+1]$. Thus, we can use persistent search trees [Dri+89] as dictionary data structures, resulting in a preprocessing time of $\mathcal{O}(n \log|\Sigma|)$ for building $D_0,\dots,D_n$ and a lookup time of $\mathcal{O}(\log|\Sigma|)$ for querying $\text{Next}^x_{=\sigma}(i)$. Using such a Next data structure, we can formulate our dynamic programming algorithm, see Algorithm 1.

Algorithm 1 Solving $\text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$ in time $\mathcal{O}((n+m^2)\log|\Sigma|)$.

Assumption: $c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}}$ are positive integers
Input: strings x, y of length n, m, $n \geq m$
Output: $\boldsymbol{\delta}_{\text{Edit}}(x,y)$

$M \leftarrow (c_{\text{del-x}} + c_{\text{del-y}})m$
Implicitly set $I[j,k] \leftarrow \infty$ for all j and all $k < 0$ or $k > M$
$I[0,0] \leftarrow 0$
$I[0,k] \leftarrow \infty$ for $0 < k \leq M$
for $j = 1,\dots,m$ **do**
  for $k = 0,\dots,M$ **do**
    $I[j,k] \leftarrow \min\{I[j-1, k-c_{\text{del-x}}-c_{\text{del-y}}],$
      $\text{Next}^x_{=y[j]}(I[j-1, k-c_{\text{match}}]),$
      $\text{Next}^x_{\neq y[j]}(I[j-1, k-c_{\text{subst}}])\}$
return $c_{\text{del-x}}(n-m) + \min\{0 \leq k \leq M \mid I[m,k] < \infty\}$.

Since $\text{Next}^x_{=\sigma}$ and $\text{Next}^x_{\neq\sigma}$ can be queried in time $\mathcal{O}(\log|\Sigma|)$ and $M = (c_{\text{del-x}} + c_{\text{del-y}})m = \mathcal{O}(m)$, Algorithm 1 runs in time $\mathcal{O}(m^2 \log|\Sigma|)$. Together with the preprocessing time for the Next data structure, we obtain a total running time of $\mathcal{O}((n+m^2)\log|\Sigma|)$. It remains to argue correctness.

Correctness. We prove that the dynamic programming table $I[j,k]$ admits the following interpretation.

Lemma 6.1.2. *Algorithm 1 computes for any $j \in [m]$, $k \in \mathbb{Z}$, the quantity*

$$I[j,k] = \min\{0 \leq i \leq n \mid \boldsymbol{\delta}_{\text{Edit}}(x[1..i], y[1..j]) - c_{\text{del-x}}(i-j) = k\}.$$

Proof. Let $R[j,k] := \min\{0 \leq i \leq n \mid \boldsymbol{\delta}_{\text{Edit}}(x[1..i], y[1..j]) - c_{\text{del-x}}(i-j) = k\}$ be the right hand side of the statement.

The statement is true for $j = 0$, since for the empty string ϵ, we have $\boldsymbol{\delta}_{\text{Edit}}(x[1..i], \epsilon) = c_{\text{del-x}} \cdot i$, so that $R[0,k] = 0$ for $k = 0$, and ∞ otherwise, which is exactly how we initialize $I[0,k]$.

We show that $R[j,k] = \infty$ for $k < 0$ or $k > M$, which is also implicitly assumed for $I[j,k]$ in Algorithm 1. Note that for $i \geq j$ we have to delete at least $i-j$ symbols in $x[1..i]$ when traversing it with $y[1..j]$, which implies $\boldsymbol{\delta}_{\text{Edit}}(x[1..i], y[1..j]) \geq c_{\text{del-x}}(i-j)$. Since additionally for $i < j$ the term $-c_{\text{del-x}}(i-j)$ is positive, we have $\boldsymbol{\delta}_{\text{Edit}}(x[1..i], y[1..j]) - c_{\text{del-x}}(i-j) \geq 0$ for all i, j. Thus, for no $k < 0$, we have $\boldsymbol{\delta}_{\text{Edit}}(x[1..i], y[1..j]) - c_{\text{del-x}}(i-j) = k$, implying $R[j,k] = \infty$ in this case. Moreover,

$\boldsymbol{\delta}_{\text{Edit}}(x[1..i], y[1..j]) \leq c_{\text{del-x}} \cdot i + c_{\text{del-y}} \cdot j$, which implies $\boldsymbol{\delta}_{\text{Edit}}(x[1..i], y[1..j]) - c_{\text{del-x}}(i-j) \leq (c_{\text{del-y}} + c_{\text{del-x}})j \leq M$. Thus, we also have $R[j,k] = \infty$ for $k > M$.

It remains to show the statement for $j > 1$ and $0 \leq k \leq M$. Inductively, we can assume that the statement holds for $j-1$. We show that $R[j,k]$ satisfies the same recursive equation as $I[j,k]$ in Algorithm 1. Let $i := R[j,k]$ and consider an optimal traversal T of $(x[1..i], y[1..j])$. We obtain a traversal T' by removing the last operation in T.

If the last operation in T is a deletion in x, then T' is an optimal traversal of $(x[1..i-1], y[1..j])$ with cost $\boldsymbol{\delta}_{\text{Edit}}(x[1..i-1], y[1..j]) = \boldsymbol{\delta}_{\text{Edit}}(x[1..i], y[1..j]) - c_{\text{del-x}}$. Thus, we can decrease i to $i-1$ while keeping $k = \boldsymbol{\delta}_{\text{Edit}}(x[1..i], y[1..j]) - c_{\text{del-x}}(i-j) = \boldsymbol{\delta}_{\text{Edit}}(x[1..i-1], y[1..j]) - c_{\text{del-x}}(i-1-j)$. This contradicts minimality of $i = R[j,k]$, so the last operation in T cannot be a deletion in x.

If the last operation in T is a deletion in y, then T' is an optimal traversal of $(x[1..i], y[1..j-1])$ with cost $\boldsymbol{\delta}_{\text{Edit}}(x[1..i], y[1..j-1]) = \boldsymbol{\delta}_{\text{Edit}}(x[1..i], y[1..j]) - c_{\text{del-y}}$. Thus, we have

$$\begin{aligned}
&R[j,k] \\
&= \min\{0 \leq i \leq n \mid \boldsymbol{\delta}_{\text{Edit}}(x[1..i], y[1..j]) - c_{\text{del-x}}(i-j) = k\} \\
&= \min\{0 \leq i \leq n \mid \boldsymbol{\delta}_{\text{Edit}}(x[1..i], y[1..j-1]) - c_{\text{del-x}}(i-(j-1)) = k - c_{\text{del-y}} - c_{\text{del-x}}\} \\
&= R[j-1, k - c_{\text{del-x}} - c_{\text{del-y}}].
\end{aligned}$$

If the last operation in T is a matching of $x[i]$ and $y[j]$, then T' is an optimal traversal of $(x[1..i-1], y[1..j-1])$ with cost $\boldsymbol{\delta}_{\text{Edit}}(x[1..i-1], y[1..j-1]) = \boldsymbol{\delta}_{\text{Edit}}(x[1..i], y[1..j]) - c_{\text{match}}$. This implies $\boldsymbol{\delta}_{\text{Edit}}(x[1..i-1], y[1..j-1]) - c_{\text{del-x}}((i-1)-(j-1)) = k - c_{\text{match}}$, so that $i-1$ is a candidate for $R[j-1, k-c_{\text{match}}]$. Let $i' := R[j-1, k-c_{\text{match}}]$ and note that $i' \leq i-1$. As $x[i] = y[j]$, we obtain $i \geq \text{Next}^x_{=y[j]}(i') =: i^*$. In the following we show $i = i^*$. By definition of i' we have $\boldsymbol{\delta}_{\text{Edit}}(x[1..i'], y[1..j-1]) - c_{\text{del-x}}(i'-j+1) = k - c_{\text{match}}$. Hence, $\boldsymbol{\delta}_{\text{Edit}}(x[1..i^*], y[1..j]) \leq c_{\text{match}} + c_{\text{del-x}}(i^*-i'-1) + \boldsymbol{\delta}_{\text{Edit}}(x[1..i'], y[1..j-1]) = k + c_{\text{del-x}}(i^*-j)$. We even have equality, since otherwise $\boldsymbol{\delta}_{\text{Edit}}(x[1..i], y[1..j]) \leq \boldsymbol{\delta}_{\text{Edit}}(x[1..i^*], y[1..j]) + c_{\text{del-x}}(i-i^*) < k + c_{\text{del-x}}(i-j)$, contradicting the definition of i. Thus, i^* is a candidate for $R[j,k]$, implying that we also have $i \leq i^*$. Hence, we have $R[j,k] = i^* = \text{Next}^x_{=y[j]}(i') = \text{Next}^x_{=y[j]}(R[j-1, k-c_{\text{match}}])$.

We argue analogously if the last operation in T is a substitution of $x[i]$ and $y[j]$. This yields

$$\begin{aligned}
R[j,k] = \min\{&R[j-1, k-c_{\text{del-x}}-c_{\text{del-y}}], \\
&\text{Next}^x_{=y[j]}(R[j-1, k-c_{\text{match}}]), \\
&\text{Next}^x_{\neq y[j]}(R[j-1, k-c_{\text{subst}}])\}.
\end{aligned}$$

Hence, $R[j,k]$ satisfies the same recursion as $I[j,k]$, and we proved $R[j,k] = I[j,k]$ for all j,k. □

We can finally prove correctness of Algorithm 1, thus proving Theorem 6.1.1.

Lemma 6.1.3. *Algorithm 1 correctly computes* $\boldsymbol{\delta}_{\text{Edit}}(x,y)$.

Proof. Among all optimal traversals of (x,y), pick a traversal T that ends with the maximal number d of deletions in x, and set $i := n-d$. Observe that i is minimal with $\boldsymbol{\delta}_{\text{Edit}}(x[1..i], y[1..m]) + c_{\text{del-x}}(n-i) = \boldsymbol{\delta}_{\text{Edit}}(x,y)$, which is equivalent to

$$\boldsymbol{\delta}_{\text{Edit}}(x[1..i], y[1..m]) - c_{\text{del-x}}(i-m) = \boldsymbol{\delta}_{\text{Edit}}(x,y) - c_{\text{del-x}}(n-m) =: k.$$

Thus, $i = I[m,k] < \infty$, which implies that the return value of Algorithm 1 is at most $c_{\text{del-x}}(n-m) + k = \boldsymbol{\delta}_{\text{Edit}}(x,y)$.

Moreover, for any k with $I[m,k] < \infty$ there is a $0 \le i \le n$ with $\boldsymbol{\delta}_{\text{Edit}}(x[1..i], y[1..m]) - c_{\text{del-x}}(i-m) = k$. By appending $n-i$ deletions in x to any optimal traversal of $(x[1..i], y[1..m])$, we obtain $\boldsymbol{\delta}_{\text{Edit}}(x,y) \le \boldsymbol{\delta}_{\text{Edit}}(x[1..i], y[1..m]) + c_{\text{del-x}}(n-i) = k + c_{\text{del-x}}(n-m)$. Hence, the return value of Algorithm 1 is also at least $\boldsymbol{\delta}_{\text{Edit}}(x,y)$. □

6.2 Improved LCS Algorithm for Alphabet Size 2

In this section, we prove Theorem 5.2.4, i.e., we consider LCS, assume that $\Sigma = \{0,1\}$ and provide an algorithm running in time $\mathcal{O}(n + \delta M/n)$ (recall that $M = \#\{(i,j) \mid x_i = y_j\}$ and $\delta = m - L$, see Section 5.1). More precisely, for any input x and y, by $\#_0(x) + \#_1(x) = n$ we have $\max\{\#_0(x), \#_1(x)\} \ge n/2$, so without loss of generality assume $\#_1(x) \ge n/2$ (otherwise exchange 0 and 1). Since $M = \#_0(x) \cdot \#_0(y) + \#_1(x) \cdot \#_1(y)$, it follows that $\#_1(y) \le 2M/n$. Hence, it suffices to design an algorithm running in time $\mathcal{O}(n + \delta \cdot \#_1(y))$.

Theorem 6.2.1. *For $\Sigma = \{0,1\}$, LCS admits an $\mathcal{O}(n + \delta \cdot \#_1(y))$-time algorithm.*

We preprocess x in time $\mathcal{O}(n)$ to support the following queries (which are a slight generalization of the queries used in Section 6.1). For $\sigma \in \{0,1\}$, $0 \le i \le n$, and $t \ge 1$, $\text{Next}^t_{=\sigma}(i)$ returns the position of the t-th occurrence of symbol σ after position i in x, i.e., $\text{Next}^t_{=\sigma}(i) = i'$ if and only if $x[i'] = \sigma$ and $\#_\sigma(x[i+1..i']) = t$ (if such a number i' does not exist then $\text{Next}^t_{=\sigma}(i) := \infty$). For convenience, we let $\text{Next}^0_{=\sigma}(i) := i$. For $t = 1$, we also write $\text{Next}^1_{=\sigma}(i) = \text{Next}_{=\sigma}(i)$. It is easy to implement $\text{Next}^t_{=\sigma}$ in time $\mathcal{O}(1)$ using rank/select data structures on the 0's and 1's in x, which can be built in time $\mathcal{O}(n)$ [Jac89; Pat08]. The symbol succeeding i is $\text{Next}(i) := i+1$ if $i+1 \le n$, or ∞ otherwise, which can be computed in time $\mathcal{O}(1)$.

Let $\lambda = \#_1(y)$ and $1 \le j_1 < \ldots < j_\lambda \le m$ be the positions of all 1's in y, and for convenience set $j_0 := 0$. We can assume that the last symbol in each of x and y is a 1, in particular $j_\lambda = m$, because appending symbol 1 to both x and y increases the LCS by exactly 1 (by Lemma 5.5.1). We write z_ℓ for the number of 0's between $j_{\ell-1}$ and j_ℓ in y, i.e., $y = 0^{z_1}10^{z_2}1 \ldots 10^{z_\lambda}1$ with $z_\ell \ge 0$.

Consider the dynamic programming table T that contains for all $0 \le \ell \le \lambda$ and $k \ge 0$ (it remains to fix an upper bound on k) the value

$$T[\ell,k] = \min\{0 \le i \le n \mid L(x[1..i], y[1..j_\ell]) = j_\ell - k\}, \tag{6.1}$$

where we set $\min\emptyset = \infty$. Observe that from T we can read off the LCS length as $L(x,y) = m - \min\{k \mid T[\lambda,k] < \infty\}$. In particular, we may initialize $\tilde{\delta} := 1$, and compute the table T for $0 \le \ell \le \lambda$ and $0 \le k \le \tilde{\delta}$. If there is no $0 \le k \le \tilde{\delta}$ with $T[\lambda,k] < \infty$, then we double $\tilde{\delta}$ and repeat. This exponential search ends once we find a value $\tilde{\delta} \in [\delta, 2\delta)$.

Next we show how to recursively compute $T[\ell,k]$. For $\ell = 0$, we clearly have $T[0,0] = 0$ and $T[0,k] = \infty$ for any $k > 0$. For $\ell > 0$, the following dynamic programming recurrence computes $T[\ell,k]$, as shown in Lemma 6.2.2 below.

$$\begin{aligned} T[\ell,k] = \min\big\{ &\min\{\text{Next}(\text{Next}^{z_\ell - k + k'}_{=0}(T[\ell-1,k'])) \mid \max\{0, k-z_\ell\} \le k' < k\}, \\ &\text{Next}_{=1}(\text{Next}^{z_\ell}_{=0}(T[\ell-1,k])), \\ &T[\ell-1, k-z_\ell-1]\big\}. \end{aligned} \tag{6.2}$$

Note that the third line only applies if $k - z_\ell - 1 \ge 0$, as $T[\ell',k'] = \infty$ for $k' < 0$.

Let us discuss how to efficiently implement the above algorithm, assuming we already have computed the values $T[\ell-1,k]$, $0 \le k \le \tilde{\delta}$. Clearly, we can evaluate the second and third line in constant time, using the Next data structures that we built in the preprocessing. For the first line, observe that $\mathrm{Next}^t_{=0}(i)$ is the position of the $(t + \#_0(x[1..i]))$-th 0 in x. Hence, $\mathrm{Next}^{z_\ell-k+k'}_{=0}(T[\ell-1,k'])$ is the position of the $(z_\ell - k + k' + \#_0(x[1..T[\ell-1,k']]))$-th 0 in x, so it is minimized if $k' + \#_0(x[1..T[\ell-1,k']])$ is minimized[2]. Thus, it suffices to compute a range minimum query over the interval $[\max\{0, k-z_\ell\}, k)$ on the array $A_\ell[0..\tilde{\delta}]$ with $A_\ell[k'] := k' + \#_0(x[1..T[\ell-1,k']])$. From the answer $A_\ell[r]$ to this range minimum query we can infer $T[\ell,k]$ in time $\mathcal{O}(1)$. Specifically, the first line evaluates to the next symbol after the position of the $(z_\ell - k + A[r])$-th 0 in x, i.e., $\mathrm{Next}(\mathrm{Next}^{z_\ell-k+A_\ell[r]}_{=0}(0))$.

Note that range minimum queries can be performed in time $\mathcal{O}(1)$, after a preprocessing of $\mathcal{O}(|A_\ell|) = \mathcal{O}(\tilde{\delta})$ [Sad07; BDR12], where $|A_\ell|$ is the size of array A_ℓ. Since we can reuse the array A_ℓ for all $0 \le k \le \tilde{\delta}$, we spend (amortized) preprocessing time $\mathcal{O}(1)$ per entry of $T[\ell,\cdot]$. In total, this yields time $\mathcal{O}(\tilde{\delta} \cdot \lambda) = \mathcal{O}(\tilde{\delta} \cdot \#_1(y))$ to build the table T. The exponential search for δ yields time $\mathcal{O}(\delta \cdot \#_1(y))$. Adding the preprocessing time $\mathcal{O}(n)$, we obtain an $\mathcal{O}(n + \delta \cdot \#_1(y))$ algorithm. It remains to prove correctness of the recursive formula (6.2).

Lemma 6.2.2. *Table (6.1) follows the recursive formula (6.2).*

Proof. For any $1 \le \ell \le \lambda$ and $0 \le k \le \tilde{\delta}$ we show that the value $T[\ell,k]$ of (6.1) follows the recursive formula (6.2). Let $i = T[\ell,k]$ and let i' be minimal with

$$L(x[1..i], y[1..j_\ell]) = L(x[1..i'], y[1..j_{\ell-1}]) + L(x[i'+1..i], y[j_{\ell-1}+1..j_\ell]). \tag{6.3}$$

Let $k' = j_{\ell-1} - L(x[1..i'], y[1..j_{\ell-1}])$. Then we claim $i' = T[\ell-1,k']$. Indeed, since i' satisfies the condition $L(x[1..i'], y[1..j_{\ell-1}]) = j_{\ell-1} - k'$ of (6.1) we have $i' \ge T[\ell-1,k']$. Moreover, if we had $i' > T[\ell-1,k']$ then we could replace i' by $T[\ell-1,k']$, as both values satisfy the condition $L(x[1..i'], y[1..j_{\ell-1}]) = j_{\ell-1} - k'$, contradicting minimality of i'.

Let $r = L(x[i'+1..i], y[j_{\ell-1}+1..j_\ell])$. By (6.3), we have $j_\ell - k = j_{\ell-1} - k' + r$, and we obtain $r = 1 + z_\ell - k + k'$ using $z_\ell = j_\ell - j_{\ell-1} - 1$. Note that $i \ge i'$ is the smallest value attaining $L(x[i'+1..i], y[j_{\ell-1}+1..j_\ell]) = r$. Indeed, if there was a smaller value $i' \le i^* < i$ with $L(x[i'+1..i^*], y[j_{\ell-1}+1..j_\ell]) = r$, then $L(x[1..i^*], y[1..j_\ell]) \ge L(x[1..i'], y[1..j_{\ell-1}]) + L(x[i'+1..i^*], y[j_{\ell-1}+1..j_\ell]) = L(x[1..i'], y[1..j_{\ell-1}]) + L(x[i'+1..i], y[j_{\ell-1}+1..j_\ell]) = L(x[1..i], y[1..j_\ell]) = j_\ell - k$. Then there also exists $0 \le i'' \le i^* < i$ with equality, i.e., $L(x[1..i''], y[1..j_\ell]) = j_\ell - k$. Indeed, if $L(x[1..i^*], y[1..j_\ell]) > j_\ell - k$ then we can repeatedly reduce i^* by 1, this reduces $L(x[1..i^*], y[1..j_\ell])$ by at most 1, and we eventually reach $j_\ell - k$ since $L(x[1..t], y[1..j_\ell]) = 0$ for $t = 0$. However, existence of $i'' < i$ contradicts minimality of $i = T[\ell,k]$.

Now we show that i is one of the terms on the right hand side of (6.2), considering three cases.

Case 1: If $1 \le r < z_\ell + 1$, then the LCS of $x[i'+1..i]$ and $y[j_{\ell-1}+1..j_\ell] = 0^{z_\ell}1$ consists of $r-1$ 0's and one additional symbol which is 0 or 1. Thus, the smallest i attaining r is $\mathrm{Next}(\mathrm{Next}^{r-1}_{=0}(i'))$, accounting for $r-1$ 0's and one additional symbol. Since $r - 1 = z_\ell - k + k'$ and $i' = T[\ell-1,k']$, we have shown that $i = T[\ell,k]$ is of the form $\mathrm{Next}(\mathrm{Next}^{z_\ell-k+k'}_{=0}(T[\ell-1,k']))$ for some k'. Observe that $1 \le r < z_\ell + 1$ implies $k - z_\ell \le k' < k$. We clearly also have $k' \ge 0$. This corresponds to the first line of (6.2).

[2]Here we interpret $\#_0(x[1..\infty])$ as ∞.

Case 2: If $r = z_\ell + 1$ then $x[i' + 1..i]$ contains $y[j_{\ell-1} + 1..j_\ell] = 0^{z_\ell}1$. Thus, $i = \text{Next}_{=1}(\text{Next}_{=0}^{z_\ell}(i'))$, accounting for z_ℓ 0's followed by a 1. In this case, we have $k' = k + r - z_\ell - 1 = k$ so that $i = T[\ell, k]$ is of the form $\text{Next}_{=1}(\text{Next}_{=0}^{z_\ell}(T[\ell - 1, k]))$. This corresponds to the second line of (6.2).

Case 3: If $r = 0$ then $i = i'$, since the smallest value $i \geq i'$ attaining $L(x[i' + 1..i], y[j_{\ell-1} + 1..j_\ell]) = 0$ is i'. In this case, we have $k' = k - z_\ell - 1$, and we obtain $T[\ell, k] = i = i' = T[\ell - 1, k'] = T[\ell - 1, k - z_\ell - 1]$. This only applies if $k - z_\ell - 1 \geq 0$. This corresponds to the third line of (6.2).

This case distinction shows that i is one of the terms on the right hand side of (6.2). Also observe that we have $i \leq \text{Next}_{=1}(\text{Next}_{=0}^{z_\ell}(T[\ell - 1, k]))$, since the number $\text{Next}_{=1}(\text{Next}_{=0}^{z_\ell}(T[\ell - 1, k]))$ is part of the set of which $i = T[\ell, k]$ is the minimum. Similarly, we have $i \leq \text{Next}(\text{Next}_{=0}^{z_\ell - k + k'}(T[\ell - 1, k']))$ for any $\max\{0, k - z_\ell\} \leq k' < k$, and $i \leq T[\ell - 1, k - z_\ell - 1]$ if $k - z_\ell - 1 \geq 0$. This proves that i is the minimum over all expressions on the right hand side of (6.2), proving that $i = T[\ell, k]$ follows the recursive formula (6.2). □

Chapter 7

Palindromic and Tandem Subsequences

In this chapter, we prove quadratic-time hardness of Longest Palindromic Subsequence (LPS) and Longest Tandem Subsequence (LTS) by presenting reductions from LCS. This proves Theorem 2.1.5. We will use the following simple facts about LCS (proven in Section 3.5), where we again regard LCS as a minimization problem by writing $\boldsymbol{\delta}_{\mathrm{LCS}}(x, y) := |x| + |y| - 2L(x, y)$; recall that $L(x, y)$ denotes the length of any LCS of x and y. In the whole chapter, we let Σ be any alphabet with $0, 1 \in \Sigma$.

Fact 3.5.3. *Let z, w be arbitrary strings and $\ell, k \in \mathbb{N}_0$. Then we have*

(i) $\boldsymbol{\delta}_{\mathrm{LCS}}(1^k z, 1^k w) = \boldsymbol{\delta}_{\mathrm{LCS}}(z, w)$,

(ii) $\boldsymbol{\delta}_{\mathrm{LCS}}(1^k z, w) \geq \boldsymbol{\delta}_{\mathrm{LCS}}(z, w) - k$, *and*

(iii) $\boldsymbol{\delta}_{\mathrm{LCS}}(0^\ell z, 1^k w) \geq \min\{k, \boldsymbol{\delta}_{\mathrm{LCS}}(z, 1^k w) + \ell\}$.

We obtain symmetric statements by flipping all bits and by reversing all involved strings.

Note that in this chapter, we write $x = (x[1], \ldots, x[|x|]) = \bigcirc_{i=1}^{|x|} x[i]$ for better readability.

7.1 Longest Palindromic Subsequence

We show that computing the length of the longest palindromic subsequence is effectively computationally equivalent to computing the length of the longest common subsequence of two strings. For completeness, we provide the following well-known result which shows that LPS can be reduced to LCS in linear time. Recall that for a string x, we denote its reversed string by $\mathrm{rev}(x)$ and the length of its longest palindromic subsequence by $P(x)$.

Fact 7.1.1 (Folklore)**.** *For any input $x \in \Sigma^*$, we have $P(x) = L(x, \mathrm{rev}(x))$.*

Proof. Let p be a palindromic subsequence of x. Then $p = \mathrm{rev}(p)$ is a common subsequence of x and $\mathrm{rev}(x)$, yielding $L(x, \mathrm{rev}(x)) \geq P(x)$.

For the other direction, let c be any LCS of x and $\mathrm{rev}(x)$ of length ℓ. It remains to show that we can find a palindromic subsequence p of x with $|p| \geq \ell$ (observe that c itself is not necessarily a palindrome). Note that c gives rise to a sequence of pairs $(a_1, b_1), \ldots, (a_\ell, b_\ell)$ such that $a_1 < \cdots < a_\ell$, $b_1 > \cdots > b_\ell$, and $c = (x[a_1], \ldots, x[a_\ell]) = (x[b_1], \ldots, x[b_\ell])$. Define $m := \lfloor \frac{\ell}{2} \rfloor + 1$. If $a_m \leq b_m$, then $a_1 < \cdots < a_m \leq b_m < \cdots < b_1$ and hence $(x[a_1], \ldots, x[a_{m-1}], x[a_m], x[b_{m-1}], \ldots, x[b_1])$ is a palindromic subsequence of x of length $2m - 1 = 2\lfloor \frac{\ell}{2} \rfloor + 1 \geq \ell$. Otherwise, i.e., if $a_m > b_m$, then $b_\ell < \cdots < b_m < a_m < \cdots < a_\ell$ gives rise to the palindromic subsequence $(x[b_\ell], \ldots, x[b_m], x[a_m], \ldots, x[a_\ell])$ of x of length $2(\ell - m + 1) = 2\ell - 2\lfloor \frac{\ell}{2} \rfloor \geq \ell$. □

To prove our lower bound for computing a longest palindromic subsequence of a string x, we present a simple reduction from LCS to LPS, and then appeal to our lower bound for LCS, which is equivalent to Edit$(1,1,0,2)$, see Theorem 2.1.2.

Theorem 7.1.2. *On input $x, y \in \Sigma^*$, we can compute, in time $\mathcal{O}(|x|+|y|)$, a string $z \in \Sigma^*$ and $\kappa \in \mathbb{N}$ such that $P(z) = 3\kappa + 2L(x,y)$.*

Proof. Let $\kappa := 2(\ell_{\mathrm{X}} + \ell_{\mathrm{Y}} + 1)$, where $\ell_{\mathrm{X}} := |x|$, $\ell_{\mathrm{Y}} := |y|$. We define

$$z := \quad x\, 0^{\kappa}\, 1^{\kappa}\, 0^{\kappa}\, \mathrm{rev}(y).$$

Clearly, z and κ can be computed in time $\mathcal{O}(\ell_{\mathrm{X}} + \ell_{\mathrm{Y}})$. Let s be a LCS of x and y. Then $s0^{\kappa}1^{\kappa}0^{\kappa}\mathrm{rev}(s)$ is a palindromic subsequence of z, which proves $P(z) \geq 3\kappa + 2L(x,y)$.

To show $P(z) \leq 3\kappa + 2L(x,y)$, fix a LPS p of z and let ℓ be its length. We define $m := \lfloor \frac{\ell}{2} \rfloor$ and denote by $p_1 = (p[1], \ldots, p[m])$ the first "half" of p. Let $z_1 = (z[1], \ldots, z[i])$ be the shortest prefix of z that contains p_1 as a subsequence and let $z_2 := (z[i+1], \ldots, z[|z|])$ be the remainder of z. Then p_1, which by definition equals $(p[\ell], \ldots, p[\ell - m + 1])$, is a subsequence of $\mathrm{rev}(z_2)$. This shows that if ℓ is even, then $\ell \leq 2L(z_1, \mathrm{rev}(z_2))$. If ℓ is odd, we may without loss of generality assume that $p[m+1] = z_2[1]$. Hence $\mathrm{rev}(p_1)$ is a subsequence of $z_2' := (z_2[2], \ldots, z_2[|z_2|])$, so that $\ell \leq 2L(z_1, \mathrm{rev}(z_2')) + 1$. It remains to show that (i) $L(z_1, \mathrm{rev}(z_2)) \leq \frac{3}{2}\kappa + L(x,y)$ and (ii) $L(z_1, \mathrm{rev}(z_2')) \leq \frac{3}{2}\kappa + L(x,y) - \frac{1}{2}$.

Assume that $|z_1| \leq \ell_{\mathrm{X}} + \kappa$ or $|z_2| \leq (\ell_{\mathrm{Y}} + 1) + \kappa$, then by $L(u,v) \leq \min\{|u|, |v|\}$ we obtain that $L(z_1, \mathrm{rev}(z_2')) \leq L(z_1, \mathrm{rev}(z_2)) \leq \max\{\ell_{\mathrm{X}}, \ell_{\mathrm{Y}} + 1\} + \kappa < \frac{3}{2}\kappa + L(x,y)$. Hence without loss of generality, $z_1 = x0^{\kappa}1^{a}$ and $z_2 = 1^{a'}0^{\kappa}\mathrm{rev}(y)$ with $a' \geq 1$, where we assume that $a' \geq a$ since the other case is symmetric. Note that (i) and (ii) are equivalent to $\boldsymbol{\delta}_{\mathrm{LCS}}(z_1, \mathrm{rev}(z_2)) \geq \boldsymbol{\delta}_{\mathrm{LCS}}(x,y)$ and $\boldsymbol{\delta}_{\mathrm{LCS}}(z_1, \mathrm{rev}(z_2')) \geq \boldsymbol{\delta}_{\mathrm{LCS}}(x,y)$, respectively. We compute

$$\begin{aligned}
\boldsymbol{\delta}_{\mathrm{LCS}}(z_1, \mathrm{rev}(z_2)) &= \boldsymbol{\delta}_{\mathrm{LCS}}(x\, 0^{\kappa}\, 1^{a}, y\, 0^{\kappa}\, 1^{a'}) \\
&= \boldsymbol{\delta}_{\mathrm{LCS}}(x\, 0^{\kappa}, y\, 0^{\kappa}\, 1^{a'-a}) && \text{(by Fact 3.5.3(i))} \\
&\geq \min\{\kappa, \boldsymbol{\delta}_{\mathrm{LCS}}(x\, 0^{\kappa}, y\, 0^{\kappa})\} && \text{(by Fact 3.5.3(iii))} \\
&= \min\{\kappa, \boldsymbol{\delta}_{\mathrm{LCS}}(x,y)\} = \boldsymbol{\delta}_{\mathrm{LCS}}(x,y). && \text{(by Fact 3.5.3(i)).}
\end{aligned}$$

By replacing a' by $a' - 1 \geq 0$, we obtain $\boldsymbol{\delta}_{\mathrm{LCS}}(z_1, \mathrm{rev}(z_2')) \geq \boldsymbol{\delta}_{\mathrm{LCS}}(x,y)$ by the same calculation. This yields $P(z) = \ell \leq 3\kappa + 2L(x,y)$, as desired.

□

7.2 Longest Tandem Subsequence

As for LPS, our lower bound for LTS follows from a simple reduction from LCS and appealing to our lower bound for LCS of Theorem 2.1.2. Recall that for a given string x, $T(x)$ denotes the length of its longest tandem subsequence.

Theorem 7.2.1. *On input $x, y \in \Sigma^*$, we can compute, in time $\mathcal{O}(|x|+|y|)$, a string $z \in \Sigma^*$ and $\kappa \in \mathbb{N}$ such that $T(z) = 4\kappa + 2L(x,y)$.*

Proof. Let $\kappa := \ell_{\mathrm{X}} + \ell_{\mathrm{Y}}$, where $\ell_{\mathrm{X}} := |x|$ and $\ell_{\mathrm{Y}} := |y|$. We define

$$z := \quad 0^{\kappa}\, x\, 1^{\kappa}\, 0^{\kappa}\, y\, 1^{\kappa}.$$

Clearly, z can be computed in time $\mathcal{O}(\ell_{\mathrm{X}} + \ell_{\mathrm{Y}})$. Let s be a LCS of x and y. Then $t := t'\, t'$ with $t' := 0^{\kappa}s1^{\kappa}$ is a tandem subsequence of z. Hence, we have $T(z) \geq |t| = 4\kappa + 2L(x,y)$.

To show $T(z) \leq 4\kappa + 2L(x,y)$, fix a LTS $t = t'\,t'$ of z. Let i be the smallest index such that t' is a subsequence of $z_1 := (z[1], \ldots, z[i])$ and let $z_2 := (z[i+1], \ldots, z[|z|])$. By choice of t, t' is also a subsequence of z_2, so that $T(z) = 2|t'| \leq 2L(z_1, z_2)$. Thus, it remains to prove that $2L(z_1, z_2) \leq 4\kappa + 2L(x,y)$.

Assume that $|z_1| \leq \kappa + \ell_{\mathrm{X}}$ or $|z_2| \leq \kappa + \ell_{\mathrm{Y}}$. Then, using $L(u,v) \leq \min\{|u|, |v|\}$, we conclude that $2L(z_1, z_2) \leq 2\kappa + 2(\ell_{\mathrm{X}} + \ell_{\mathrm{Y}}) \leq 4\kappa + 2L(x,y)$.

Hence, without loss of generality, we have (i) $z_1 = 0^\kappa x 1^\ell$ and $z_2 = 1^{\ell'} 0^\kappa y 1^\kappa$ or (ii) $z_1 = 0^\kappa x 1^\kappa 0^\ell$ and $z_2 = 0^{\ell'} y 1^\kappa$, for some ℓ, ℓ' with $\ell + \ell' = \kappa$. We only consider case (i), since case (ii) is symmetric. Note that $2L(z_1, z_2) \leq 4\kappa + 2L(x,y)$ is equivalent to $\boldsymbol{\delta}_{\mathrm{LCS}}(z_1, z_2) \geq \boldsymbol{\delta}_{\mathrm{LCS}}(x,y)$. We obtain

$$
\begin{aligned}
\boldsymbol{\delta}_{\mathrm{LCS}}(z_1, z_2) &= \boldsymbol{\delta}_{\mathrm{LCS}}(0^\kappa x 1^\ell, 1^{\ell'} 0^\kappa y 1^\kappa) \\
&\geq \min\{\kappa, \boldsymbol{\delta}_{\mathrm{LCS}}(0^\kappa x 1^\ell, 0^\kappa y 1^\kappa) + \ell'\} && \text{(by Fact 3.5.3(iii))} \\
&= \min\{\kappa, \boldsymbol{\delta}_{\mathrm{LCS}}(x 1^\ell, y 1^\kappa) + \ell'\} && \text{(by Fact 3.5.3(i))} \\
&= \min\{\kappa, \boldsymbol{\delta}_{\mathrm{LCS}}(x, y 1^{\kappa-\ell}) + \ell'\} && \text{(by Fact 3.5.3(i))} \\
&\geq \min\{\kappa, \boldsymbol{\delta}_{\mathrm{LCS}}(x,y) - (\kappa - \ell) + \ell'\} && \text{(by Fact 3.5.3(ii))} \\
&= \min\{\kappa, \boldsymbol{\delta}_{\mathrm{LCS}}(x,y)\} = \boldsymbol{\delta}_{\mathrm{LCS}}(x,y),
\end{aligned}
$$

which proves the desired inequality $2L(z_1, z_2) \leq 4\kappa + 2L(x,y)$. □

Chapter 8

Open Problems and Outlook

In this part, we proved conditional lower bounds for natural polynomial-time problems: Edit Distance for general operation costs (including its special cases Longest Common Subsequence and Levenshtein distance), Dynamic Time Warping, Longest Palindromic Subsequence, and Longest Tandem Subsequence. Our results give strong evidence that the known algorithms for these problems are optimal up to lower order factors, even restricted to binary strings and one-dimensional curves, respectively. We hope that the underlying framework will find application in hardness proofs for further similarity measures, and that the studied problems serve as starting points for further reductions.

Surprisingly, using our framework to derive conditional lower bounds even yields stronger hardness results than "mere SETH-hardness": Recent work by Abboud et al. [Abb+16a] shows that the alignment gadget captures even more powerful variants of the satisfiability problem, namely a variant of satisfiability of branching programs. Interestingly, in this vein, one can independently investigate what the most powerful class of satisfiability variants is that is captured by the alignment gadget (strengthening the hardness of all problems captured in the framework), and find further problems to apply this framework to (broadening the scope of results). Furthermore, our alignment gadgets for LCS and Edit Distance might find further applications even outside of the framework (see, e.g., results improving conditional hardness of the RNA folding problem [Cha15], which exploits our LCS alignment gadget over binary alphabets).

It remains an open question whether constant-factor approximations running in strongly subquadratic time can be ruled out for the above problems assuming SETH. Furthermore, most polynomial-time lower bounds show quadratic-time barriers, and it is challenging to prove matching SETH-based lower bounds for problems with, say, cubic or $\mathcal{O}(n^{3/2})$-time algorithms (only few results are known in this direction, see, e.g., [ABVW15b; PW10]).

Turning to an even finer-grained perspective, we also presented a systematic multivariate study of SETH-based lower bounds, focusing on LCS. Here, we regarded polynomial restrictions of all 7 previously studied input parameters for LCS. Our tight conditional lower bounds completely explain the lack of polynomial time improvements since the early 1990s, except for a special regime on $\Sigma = \{0, 1\}$, for which we designed an improved algorithm matching our lower bound. We conclude that to obtain polynomially faster LCS algorithms, one has to either (1) refute the Strong Exponential Time Hypothesis or (2) design new reasonable and algorithmically tractable input parameters.

These results showcase our paradigm of *multivariate fine-grained complexity* on a classic problem with 7 parameters that display a complex set of relations. We believe that this paradigm can be applied to further problems, yielding similar systematic insights and helping to find algorithmic improvements for natural special cases.

Part II

Approximation Algorithms on Realistic Inputs

Chapter 9

Introduction to Part II

Classically, theoretical computer scientists judge an algorithm's performance by analyzing its behavior on worst-case inputs. This provides an immensely useful paradigm in general, as such a worst-case analysis is well understood and effectively provides the foundation for complexity theory. Still, it suffers from certain drawbacks when it comes to transferring theoretical results to the empirical world of practical applications: For instance, the famous Simplex method for linear programming is known to require exponential time in the worst case[1], yet it performs remarkably well in practice. Thus, worst-case analysis appears to give a too pessimistic view on the practical performance of the Simplex method. For other problems than linear programming, we face long-standing time barriers that could not yet be beaten by any algorithm on *all* instances, however, *practical* instances might have significantly simpler complexity. In both cases, restricting the input to a natural model of "typical", realistic instances might provide insights that more closely reflect an algorithm's performance or a problem's complexity in practice.

In this thesis, we consider two such models: the general paradigm of *smoothed analysis* (applicable to many problems, especially Euclidean optimization problems) and the rather ad-hoc notion of *c-packed curves* (which are particularly useful for curve similarity measures). These approaches do not only constitute models of realistic inputs, but also yield insights into the structure and properties of hard instances for a problem. In the two main chapters of this part, we use these realistic input assumptions to analyze the approximability of two well-known problems: the Euclidean Traveling Salesperson Problem (TSP) and the Fréchet distance.

9.1 Smoothed Analysis of the 2-Opt Heuristic

The Traveling Salesperson Problem (TSP) is one of the best-studied combinatorial optimization problems. In Chapter 10, we are concerned with the following variant, called Euclidean TSP: given points $X \subseteq [0,1]^d$, find the Hamiltonian cycle (i.e., a cycle visiting all points in X, also called a *tour*) of minimal length (i.e., it minimizes the sum of the Euclidean distances of neighboring points on the tour). Even this restricted variant is NP-hard for $d \geq 2$ [Pap77].

While Euclidean TSP famously admits a polynomial-time approximation scheme (PTAS) [Aro98; Mit99], heuristics that are simpler and easier to implement are often used in practice. A very simple and popular heuristic for finding near-optimal tours quickly is 2-Opt: starting from an initial tour, we iteratively replace two edges by two other edges to obtain a shorter tour until we have found a local optimum. Experiments indicate that this heuristic converges quickly and produces solutions that are within a few percent of the optimal solution [JM97; JM02]. In contrast to its success on practical instances, however, 2-Opt performs poorly in the worst case: the worst-case running-time is exponential

[1] This holds for most, if not all known pivoting rules.

even for $d = 2$ [ERV14] and its worst-case approximation ratio of $\mathcal{O}(\log n)$ has an almost matching lower bound of $\Omega(\log n / \log\log n)$ for Euclidean instances [CKT99].

Smoothed Analysis. In order to explain the performance of algorithms whose worst-case performance guarantee does not reflect the empirical performance (in particular, to explain the performance of the Simplex method, as mentioned above), the paradigm of smoothed analysis has been introduced by Spielman and Teng [ST04]. It provides a hybrid of worst-case analysis (which is often too pessimistic) and average-case analysis (which is often dominated by completely random instances that have special properties not shared by typical instances). In smoothed analysis, an adversary specifies a (worst-case) instance, which is then slightly randomly perturbed. The smoothed performance is the expected performance, where the expectation is taken over the random perturbation, on the worst-case choice of the adversary. The motivating assumption of smoothed analysis is that practical instances are often subjected to a small amount of random noise, e.g., resulting from measurement errors or numerical imprecision. Smoothed analysis often allows more realistic conclusions about the performance of an algorithm than mere worst-case or average-case analysis.

Besides as a justification for the Simplex method, smoothed analysis has been applied successfully to explain the running time of the 2-Opt heuristic [ERV14; MV13] as well as other local search algorithms [AMR11; MR13; AV09]. For an overview of this paradigm, we refer to two surveys on smoothed analysis [MR11; ST09].

Much less than for the running time is known about the smoothed approximation performance of algorithms. Karger and Onak have shown that multi-dimensional bin packing can be approximated arbitrarily well for smoothed instances [KO07] and there are frameworks to approximate Euclidean optimization problems such as TSP for smoothed instances [BMR13; CK15]. However, these approaches mostly consider algorithms tailored to solving smoothed instances. With respect to well-established algorithms, we are only aware of analyses of the jump and lex-jump heuristics for scheduling [Bru+14; Ets15] and an upper bound of $\mathcal{O}(\phi^{1/d})$ for the smoothed approximation ratio of 2-Opt in the so-called one-step model [ERV14]. Here, ϕ is an upper bound on the density functions according to which the points are drawn. Translated to Gaussian perturbation, we would obtain an upper bound of $\mathcal{O}(1/\sigma)$ if we truncate the Gaussian distribution such that all points lie in a hypercube of constant side length.

Our Results. In order to explain the practical approximation performance of 2-Opt, we provide an improved smoothed analysis of its approximation ratio in Chapter 10. More precisely, we provide bounds on the quality of the worst local optimum, when the n data points from $[0, 1]^d$ are perturbed by Gaussian distributions of standard deviation σ. In Section 10.3, we prove an upper bound of $\mathcal{O}(\log(1/\sigma))$ that improves significantly upon the direct translation of the bound of Englert et al. [ERV14] to Gaussian perturbations (see Section 10.2 for how to translate the bound to Gaussian perturbations). It smoothly interpolates between the average-case constant approximation ratio and the worst-case bound of $\mathcal{O}(\log n)$.

To complement our upper bound, we show that the lower bound by Chandra et al. [CKT99] remains true for $\sigma = \mathcal{O}(1/\sqrt{n})$ in Section 10.4.

Technical Contributions. To obtain our improved upper bound for the smoothed approximation ratio, we take into account the *unperturbed positions* of the points, which we also call their *origins*. Although this information is not available to the algorithm,

it can be exploited in the analysis. The smoothed analyses of approximation ratios so far [ERV14; Bru+14; CK15; BMR13; Ets15; KO07] essentially ignored this information. While simplifying the analysis, being oblivious to the unperturbed positions seems to be too pessimistic. In fact, we observe that the bound of Englert et al. [ERV14] cannot be improved beyond $\mathcal{O}(1/\sigma)$ if the unperturbed positions of the points are ignored, i.e., if we separately bound the values of the smoothed local and global optima from above and below, respectively, on all instances. Intuitively, the reason for this limitation is that the lower bound for the global optimum is obtained if all points have the same origin, which corresponds to an average-case rather than a smoothed analysis. On the other hand, the upper bound for the local optimum has to hold for all choices of the unperturbed points, most of which yield higher costs for the global optimum than the average-case analysis.

Our approach exploits the structure of the unperturbed instance by an analysis that (1) argues about the global structure of the longest 2-optimal tour using worst-case bounds and (2) carefully reduces local structures of the longest 2-optimal tour to an almost-average case. The first task uses arguments along the lines of the worst-case bound of Chandra et al. [CKT99]. For the second task, we observe that by conditioning a set of points to be perturbed into a small part of the input space, we obtain a random local instance. If we can choose local instances such that the maximum density required to express the resulting (conditional) distributions of the points is small, we can apply the results of Englert et al. [ERV14] to bound a local approximation ratio. Particularly for this task, we exploit information about the origins of the points. Finally, to obtain a global bound after analyzing local instances, we use, among other ideas, tools from average-case analysis such as a certain superadditivity of the length of the optimal tour.

9.2 Realistic Input Curves for the Fréchet Distance

Analyzing the similarity of curves is a well-studied topic in computation geometry, with ample applications in shape matching, speech recognition, signature verification and sequence analysis. A popular measure for this task is the Fréchet distance, which has two classic variants. Roughly speaking, the *continuous Fréchet distance* of two curves π and σ is the minimal length of a leash required to connect a dog to its owner as they walk without backtracking along π and σ, respectively. In the *discrete Fréchet distance* we replace the dog and its owner by two frogs – in each time step, each frog can jump to the next vertex along its curve or stay where it is.

In a seminal paper in 1991, Alt and Godau introduced the continuous Fréchet distance to computational geometry [AG95; God91]. For polygonal curves π and σ with n and m vertices[2], respectively, they presented an $\mathcal{O}(nm \log nm)$ algorithm. The discrete Fréchet distance was defined by Eiter and Mannila [EM94], who presented an $\mathcal{O}(nm)$ algorithm.

Since then, Fréchet distance has become a rich field of research: The literature contains generalizations to surfaces (see, e.g., [AB10]), approximation algorithms for realistic input curves ([Aro+06; AKW04; DHPW12]), the geodesic and homotopic Fréchet distance (see, e.g., [Cha+10; CW10]), and many more variants (see, e.g., [BBW09; DHP13; Mah+11; Ind02]). As a natural measure for curve similarity [Alt09], the Fréchet distance has found applications in various areas such as signature verification (see, e.g., [MP99]), map-matching tracking data (see, e.g., [Bra+05]), and moving objects analysis (see, e.g., [Buc+11]).

Analogous to the situation for Dynamic Time Warping, Edit distance and similar problems discussed in Part I, apart from log-factor improvements [Aga+14; Buc+14] the

[2] We always assume that $m \leq n$.

quadratic complexity of the classic algorithms for the continuous and discrete Fréchet distance are still state-of-the-art. In fact, Bringmann [Bri14] recently showed a conditional lower bound: Assuming the Strong Exponential Time Hypothesis[3] (SETH), there is no algorithm for the (continuous or discrete) Fréchet distance running in time $\mathcal{O}((nm)^{1-\delta})$ for any $\delta > 0$, hence apart from lower order factors of the form $n^{o(1)}$, the classic algorithms are optimal under the assumption SETH.

Realistic Inputs. Already before the conditional lower bound appeared, attempts to obtain faster algorithms for realistic inputs have been made by considering various restricted classes of curves, such as backbone curves [Aro+06], κ-bounded and κ-straight curves [AKW04], and ϕ-low density curves [DHPW12]. The most popular model of realistic inputs are *c-packed curves*, introduced by Driemel et al. [DHPW12]. A curve π is c-packed if for any point $z \in \mathbb{R}^d$ and any radius $r > 0$, the total length of π inside the ball $B(z, r)$ is at most cr, where $B(z, r)$ is the ball of radius r around z. Intuitively, requiring a curve to be c-packed (for small c) prohibits it from displaying particularly complex, unrealistic behaviour on small parts of the input space. This notion has found applications for several generalizations of the Fréchet distance, such as map matching [Che+11], the mean curve problem [HPR11], a variant of the Fréchet distance allowing shortcuts [DHP13], and Fréchet matching queries in trees [GS13].

Together with introducing c-packed curves as models of realistic inputs, Driemel et al. presented a $(1+\varepsilon)$-approximation for the continuous Fréchet distance running in time $\mathcal{O}(cn/\varepsilon + cn \log n)$, which is applicable for curves in any $\mathbb{R}^d$ with constant $d \geq 2$. Assuming SETH, the following lower bounds have been shown for c-packed curves: (1) For sufficiently small constant $\varepsilon > 0$,there is no $(1+\varepsilon)$-approximation running in time $\mathcal{O}((cn)^{1-\delta})$ for any $\delta > 0$ [Bri14]. Thus, for constant ε the algorithm by Driemel et al. is optimal apart from lower order terms of the form $n^{o(1)}$. (2) In any dimension $d \geq 5$ and for varying $\varepsilon > 0$, there is no $(1+\varepsilon)$-approximation running in time $\mathcal{O}((cn/\sqrt{\varepsilon})^{1-\delta})$ for any $\delta > 0$ [Bri14].

Our Results. In Chapter 11, we are interested in better-than-constant approximation algorithms for the Fréchet distance on realistic input curves. Specifically, for any constant $0 < \beta < 1$, we set $\varepsilon = n^{-\beta}$ and ask what running time is necessary to compute a $(1+\varepsilon)$-approximation of the Fréchet distance on c-packed curves. Note that in this regime, the known upper and lower bounds differ by a factor $\sqrt{\varepsilon}^{-1-o(1)} = n^{\beta/2+o(1)}$. We improve upon the algorithm by Driemel et al. [DHPW12] by presenting an algorithm that matches the conditional lower bound of [Bri14].

Theorem 9.2.1. *For any $0 < \varepsilon \leq 1$, we can compute a $(1+\varepsilon)$-approximation of the continuous and discrete Fréchet distance on c-packed curves in time $\tilde{\mathcal{O}}(cn/\sqrt{\varepsilon})$.*

Note that here we use the $\tilde{\mathcal{O}}$-notation to hide polylogarithmic factors in n and $1/\varepsilon$. Specifically, our running time is $\mathcal{O}(\frac{cn}{\sqrt{\varepsilon}} \log(1/\varepsilon) + cn \log n)$ for the discrete variant and $\mathcal{O}(\frac{cn}{\sqrt{\varepsilon}} \log^2(1/\varepsilon) + cn \log n)$ for the continuous variant. This improves upon the algorithm by Driemel et al. for any $\varepsilon \ll 1/\log n$; in particular for $\varepsilon = \Theta(1/n)$ and $c = \Theta(1)$ we are faster by a factor of nearly $\sqrt{n}$. While our algorithm might be too complex to speed up the algorithm by Driemel et al. in practical situations, it clarifies the optimal asymptotic dependence on c,n, and ε – apart from lower order factors $n^{o(1)}$, in dimension $d \geq 5$, and unless SETH fails [Bri14]. Specifically, for any constants $\alpha, \beta \geq 0$, even after restricting

[3]For a definition, see Section 3.3.

$c = n^{\alpha+o(1)}$ and $\varepsilon = n^{-\beta+o(1)}$, any algorithm takes time $\Omega(\min\{cn/\sqrt{\varepsilon}, n^2\}^{1-o(1)})$ in dimension $d \geq 5$ [Bri14], i.e., any algorithm is at most a factor of $n^{o(1)}$ faster than the better of our proposed algorithm ($\tilde{\mathcal{O}}(cn/\sqrt{\varepsilon})$) and the exact algorithm ($\tilde{\mathcal{O}}(n^2)$).

Technical Contributions. Our algorithmic approach is inspired by the conditional lower bound of Bringmann [Bri14]. Inspecting why the construction could not be improved to obtain a matching lower bound, we identified a potentially tractable special case: decide, for curves contained in disjoint balls of radius $\mathcal{O}(\sqrt{\varepsilon})$, whether their Fréchet distance is at most 1 or at least $1+\varepsilon$. Indeed, one can reduce this problem to computing the discrete Fréchet distance $d_{\mathrm{dF}}(\pi, \sigma)$ of *one-dimensional, separated* curves π and σ (see Chapter 11 for a precise definition). Fortunately, this case admits an exact greedy algorithm running in near-linear time. To exploit this special case in the algorithmic framework of Driemel et al. [DHPW12], however, we have to provide a highly non-trivial extension to solve a more general problem: given one-dimensional, separated curves $\pi = (\pi_1, \ldots, \pi_n)$ and $\sigma = (\sigma_1, \ldots, \sigma_m)$, a value δ, and a set of *entries* on π, specified by $1 \leq i_1 < i_2 < \cdots < i_\ell \leq n$, determine all feasible *exits*, i.e., (1) all $1 \leq j \leq n$ such that some subcurve $\pi' := (\pi_{i_k}, \ldots, \pi_j)$ (for some $1 \leq k \leq \ell$ with $i_k \leq j$) satisfies $d_{\mathrm{dF}}(\pi', \sigma) \leq \delta$, and (2) all $1 \leq j \leq m$ such that some subcurve $\pi' := (\pi_{i_k}, \ldots, \pi_n)$ (for some $1 \leq k \leq \ell$) satisfies $d_{\mathrm{dF}}(\pi', (\sigma_1, \ldots, \sigma_j)) \leq \delta$. Solving this problem in near-linear time is our main technical contribution. We provide more intuition for this problem in the beginning of Chapter 11.

9.3 Notes

The contents of Chapter 10 are based on the extended abstract that has been presented at ICALP'15 [KM15], co-authored by Bodo Manthey. The contents of Chapter 11 result from joint work with Karl Bringmann that has previously been presented at ISAAC'15 [BK15a] (with an extended version accessible online at [BK14]) and are included in similar form in Bringmann's PhD thesis [Bri15]. A journal version is currently under submission.

Acknowledgements

The author wishes to express how much he enjoyed collaborating with his co-authors Bodo Manthey and Karl Bringmann on the results in this part, and thanks the anonymous reviewers of our submissions for helpful remarks.

Chapter 10

Smoothed Analysis of the 2-Opt Heuristic

In this chapter, we analyze the approximation performance of the 2-Opt heuristic for Euclidean TSP (for an introduction, see Chapter 9). We start with introducing our notation, basic definitions and the formal description of our model of smoothed analysis in Section 10.1. We proceed to give general upper and lower bounds for the length of 2-optimal tours (i.e., locally optimal tours for the 2-Opt heuristic) in Section 10.2. Our upper bound of $\mathcal{O}(\log(1/\sigma))$ is proved in Section 10.3, which is complemented by an almost matching lower bound in Section 10.4.

10.1 Preliminaries

Throughout this chapter, we consider input in the Euclidean space $[0,1]^d$ and assume the dimension $d \geq 2$ to be a fixed constant. Given a sequence of points $X = (X_1, \ldots, X_n)$ in $\mathbb{R}^d$, we call a collection $T \subseteq [n] \times [n]$ of edges a *tour*, if T is connected and every $i \in [n] = \{1, \ldots, n\}$ has in- and outdegree exactly one in T. (Note that we consider directed tours, which is useful in the analysis, but our distances are always symmetric.) Given any collection of edges S, its length is denoted by $L(S) = \sum_{(u,v)\in S} d(u,v)$, where $d(u,v)$ denotes the Euclidean distance $\|X_u - X_v\|$ between points X_u and X_v. We call a tour T *2-optimal*, if $d(u,v)+d(w,z) \leq d(u,w)+d(v,z)$ for all edge pairs $(u,v),(w,z) \in T$. Equivalently, it is not possible to obtain a shorter tour by replacing (u,v) and (w,z) in a 2-optimal tour T by two new edges. The 2-Opt heuristic replaces a pair of edges (u,v) and (w,z) by (u,w) and (v,z) if this decreases the tour length while this is possible. Thus, it terminates with a 2-optimal tour.

We call a collection $T \subseteq [n]^2$ a *partial 2-optimal tour* if T is a subset of a tour and $d(u,v) + d(w,z) \leq d(u,w) + d(v,z)$ holds for all edges $(u,v),(w,z) \in T$. Our main interests are the traveling salesperson functional $\mathsf{TSP}(X) := \min_{\text{tour } T} L(T)$ and the functional $\mathsf{2OPT}(X) := \max_{\text{2-optimal tour } T} L(T)$ mapping the point set X to the length of the longest 2-optimal tour through X.

We note that the results in Section 10.2 hold for metrics induced by arbitrary norms in $\mathbb{R}^d$ (Lemma 10.2.1 and 10.2.2) or typical ℓ_p norms (Lemma 10.2.3 and 10.2.4), not only for the Euclidean metric. We conjecture that also the upper bound in Section 10.3 holds for more general metrics, while the lower bound in Section 10.4 is probably specific for the Euclidean metric. Still, we think that the construction can be adapted to work for most natural metrics.

Perturbation models. In the Gaussian perturbation model (also called *two-step model*) for smoothed analysis, an adversary specifies points $x_1, \ldots, x_n$ in $[0,1]^d$ that serve as unperturbed *origins*. Each such point x_i is perturbed independently by adding a

normally distributed random variable of mean 0 and standard deviation σ independently to each coordinate. Equivalently, we draw n random noise vectors $Z_i \sim \mathcal{N}(0, \sigma^2)$, where by abuse of notation $\mathcal{N}(0, \sigma^2)$ refers to the multivariate normal distribution with covariance matrix $\operatorname{diag}(\sigma^2)$, to obtain the *perturbed input* $X_1 = x_1 + Z_1, \ldots, X_n = x_n + Z_n$. For compactness, we denote the set of unperturbed points by $\overline{X} = \{x_1, \ldots, x_n\}$ and the set of perturbed points by $X = \{X_1, \ldots, X_n\}$. We write $X \leftarrow \operatorname{pert}_\sigma(\overline{X})$ to make explicit from which point set $\overline{X}$ the points in X are obtained.

Note that we may assume $\sigma \leq 1$ without loss of generality. Indeed, if $\sigma > 1$, we can rescale the instance to be contained in $[0, 1/\sigma]^d$ and perturb the points by Gaussians with standard deviation 1 instead, which gives an equivalent instance. Thus, every upper bound for $\sigma = 1$ carries over to larger values of σ.

The ϕ-bounded perturbation model (also called *one-step model*) lets the adversary directly specify (not necessarily identical) distributions by choosing probability density functions $f_1, \ldots, f_n : [0, 1]^d \to [0, \phi]$. The perturbed input is then generated by independently sampling $X_1 \sim f_1, \ldots, X_n \sim f_n$. Note that the resulting input is always contained in $[0, 1]^d$ and with higher ϕ, the adversary can concentrate points to smaller regions of the input space. Roughly speaking, when translating Gaussian perturbations to the one-step model, ϕ is proportional to σ^{-d} for fixed d.

Technical Tools. The following standard lemma provides a convenient way to bound the deviation of a perturbed point from its mean in the two-step model.

Lemma 10.1.1 (Chi-square bound [ST04, Cor. 2.19])**.** *Let x be a Gaussian random vector in $\mathbb{R}^d$ of standard deviation σ centered at the origin. Then, for $t \geq 3$, we have $\Pr\big[\|x\| \geq \sigma 3\sqrt{d \ln t}\big] \leq t^{-2.9d}$.*

To give large-deviation bounds on sums of independent variables with bounded support, we will make use of a standard Chernoff-Hoeffding bound.

Lemma 10.1.2 (Chernoff-Hoeffding Bound [DP09, Exercise 1.1])**.** *Let $X := \sum_{i=1}^n X_i$, where $X_i, i = 1, \ldots, n$ are independently distributed in $[0, 1]$, and $\mu_L \leq \mathrm{E}[X] \leq \mu_H$. Then, for $0 \leq \varepsilon \leq 1$, we have*

$$\Pr\big[X > (1+\varepsilon) \cdot \mu_H\big] \leq \exp(-(\varepsilon^2/3) \cdot \mu_H),$$
$$\Pr\big[X < (1-\varepsilon) \cdot \mu_L\big] \leq \exp(-(\varepsilon^2/2) \cdot \mu_L).$$

For obtaining lower bounds on the length of optimal tours, we consider the boundary functional $\mathsf{TSP}_B(X)$ that attains the length of the shortest tour through all points in X that is allowed to traverse the boundary of $[0, 1]^d$ at zero cost. For a proof of the following lemma, we refer to the monograph by Yukich [Yuk98].

Lemma 10.1.3 (Boundary Functional [Yuk98, Lemma 3.7])**.** *There is a constant $C > 0$ such that for all sets $X \subseteq [0, 1]^d$ of n points, we have $\mathsf{TSP}_B(X) \geq \mathsf{TSP}(X) - Cn^{\frac{d-2}{d-1}}$.*

10.2 Length of 2-Optimal Tours under Perturbations

In this section, we provide an upper bound for the length of any 2-optimal tour and a lower bound for the length of any global optimum. These two results yield an upper bound of $\mathcal{O}(1/\sigma)$ for the approximation ratio.

Chandra et al. [CKT99] proved a bound on the worst-case length of 2-optimal tours that, in fact, already holds for the more general notion of *partial* 2-optimal tours. For an

intuition why this is true, let us point out that their proof strategy is to argue that not too many long arcs in a tour may have similar directions due to the 2-optimality of the edges, while short edges do not contribute much to the length. The claim then follows from a packing argument. It is straight-forward to verify that it is never required that the collection of edges is closed or connected.

Lemma 10.2.1 (Length of partial 2-optimal tours [CKT99, Theorem 5.1], paraphrased)**.** *There exists a constant c_d such that for every sequence X of n points in $[0,1]^d$, any partial 2-optimal tour has length less than $c_d \cdot n^{1-1/d}$.*

While this bound directly applies to any perturbed instance under the one-step model, Gaussian perturbations fail to satisfy the premise of bounded support in $[0,1]^d$. However, Gaussian tails are sufficiently light to enable us to translate the result to the two-step model by carefully taking care of outliers.

Lemma 10.2.2. *There exists a constant b_d such that for any $\sigma \leq 1$ the following statement holds. For any $\bar{X}$, the probability that any partial 2-optimal tour on $X \leftarrow \mathrm{pert}_\sigma(\overline{X})$ has length greater than $b_d \cdot n^{1-1/d}$, i.e., $\mathsf{2OPT}(X) > b_d \cdot n^{1-1/d}$, is bounded by $\exp(-\Omega(\sqrt{n}))$. Furthermore,*

$$\mathrm{E}_{X \leftarrow \mathrm{pert}_\sigma(\overline{X})}\big[\mathsf{2OPT}(X)\big] \leq b_d \cdot n^{1-1/d}.$$

Proof. By translation, assume without loss of generality that the input points are contained in $[-1/2, 1/2]^d$. We define cubes $C_1 = \ell_1[-1,1]^d, C_2 = \ell_2[-1,1]^d, \dots$ with $\ell_k := 6\sqrt{kd \ln 3}$. The side length of cube C_k is $2\ell_k$. We consider the partitioning of $\mathbb{R}^d$ into the regions C_1 and $C_i \setminus C_{i-1}$ for $i \geq 2$. For some cube C and any tour T, let $E_C(T)$ denote the edges in T that are completely contained in C. For any tour T, the sequence $E_1, E_2, \dots$ defined by $E_1 := E_{C_1}(T)$ and $E_k := E_{C_k}(T) \setminus E_{C_{k-1}}(T)$, for $k \geq 2$, partitions the edges of T. Thus, $L(T) = \sum_{k=1}^{\infty} L(E_k)$.

For any outcome of the perturbed points, let T be the longest 2-optimal tour. Then, each E_k is a partial 2-optimal tour in C_k. Let n_k be the (random) number of points in $\mathbb{R}^d \setminus C_{k-1}$, which is an upper bound on the number of points in $C_k \setminus C_{k-1}$. At most $3n_k$ vertices are incident to the edges E_k, since each such edge is incident to at least one endpoint in $C_k \setminus C_{k-1}$ and every point has degree 2 in T. Since $C_k = \ell_k[-1,1]^d$ is a translated unit cube scaled by $2\ell_k$, Lemma 10.2.1 yields $L(E_k) \leq c_d \cdot (2\ell_k)(3n_k)^{1-1/d}$.

Observe that X_i is not contained in C_k only if its origin has been perturbed by noise of length at least $\ell_k/2$. Thus, let $Z \sim \mathcal{N}(0, \sigma^2)$ and note that $\sigma \leq 1$ implies that $\ell_k/2 \geq 3\sqrt{dk \ln 3}\sigma$. Hence, for each point X_i, Lemma 10.1.1 yields

$$\Pr[X_i \notin C_k] \leq \Pr\left[\|Z\| \geq \frac{\ell_k}{2}\right] \leq 3^{-(2.9d)k}.$$

By linearity of expectation, we conclude $\mathrm{E}[n_k] \leq n3^{-(2.9d)(k-1)}$ for $k \geq 1$. This yields

$$\begin{aligned}\mathrm{E}\big[L(T)\big] &= \sum_{k=1}^{\infty} \mathrm{E}[L(E_k)] \leq \sum_{k=1}^{\infty} c_d \cdot (2\ell_k)(3\,\mathrm{E}[n_k])^{1-1/d} \\ &\leq c_d \cdot 12\sqrt{d \ln 3} \cdot (3n)^{1-1/d} \left(\sum_{k=1}^{\infty} \sqrt{k} 3^{-2.9(d-1)(k-1)}\right) = \mathcal{O}(n^{1-1/d}),\end{aligned}$$

where we used Jensen's inequality for the first inequality.

To derive tail bounds for the length of any 2-optimal tour, let $N_k := n3^{-2.9d(k-1)}$ be the upper bound on $\mathrm{E}[n_k]$ derived above. By the Chernoff bound of Lemma 10.1.2, we

have that

$$\Pr[n_k \geq 2N_k] \leq \exp(-N_k/3).$$

This guarantee is only strong as long as N_k is sufficiently large. Hence, we regard this guarantee only for $1 \leq k \leq k_1$, where k_1 is chosen such that $\sqrt{n} \leq N_{k_1} \leq 3^{2.9d}\sqrt{n}$. Assume that $n_k \leq 2N_k$ for all $1 \leq k \leq k_1$. Then, analogously to the above calculation, the contribution of $E_1, \dots, E_{k_1}$ is bounded by

$$\begin{aligned}\sum_{k=1}^{k_1} L(E_k) &\leq \sum_{k=1}^{k_1} c_d \cdot (2\ell_k)(6N_k)^{1-1/d} \\ &\leq c_d \cdot 12\sqrt{d \ln 3} \cdot (6n)^{1-1/d} \left(\sum_{k=1}^{\infty} \sqrt{k} 3^{-2.9(d-1)(k-1)} \right) = \mathcal{O}(n^{1-1/d}).\end{aligned}$$

Let p_1 denote the probability that some $1 \leq k \leq k_1$ fails to satisfy $n_k \leq 2N_k$. Then,

$$p_1 \leq \sum_{k=1}^{k_1} \Pr[n_k > 2N_k] \leq k_1 \exp(-N_{k_1}/3) = \exp(-\Omega(\sqrt{n})).$$

Let us continue assuming that all $1 \leq k \leq k_1$ satisfy $n_k \leq 2N_k$. Since in particular $n_{k_1} \leq 2N_{k_1}$, at most $n_{k_1} \leq 2 \cdot 3^{2.9d}\sqrt{n}$ vertices remain outside C_{k_1-1}. Let $k_2 := \lceil \sqrt{n} \rceil$. By a union bound,

$$p_2 := \Pr[\exists j : X_j \notin C_{k_2}] \leq n 3^{-2.9d(k_2-1)} = \exp(-\Omega(\sqrt{n})).$$

Assume that the corresponding event holds (i.e., $X \subseteq C_{k_2}$), then the remaining points outside C_{k_1-1} (and hence, outside C_{k_1}) are contained in C_{k_2}. We conclude that, with probability at least $1 - (p_1 + p_2) = 1 - \exp(-\Omega(\sqrt{n}))$,

$$\begin{aligned}\sum_{k=k_1+1}^{\infty} L(E_k) &= \sum_{k=k_1+1}^{k_2} L(E_k) \leq c_d \cdot (2\ell_{k_2})(6N_{k_1})^{1-1/d} \\ &= \mathcal{O}(\sqrt{k_2}(N_{k_1})^{1-1/d}) = \mathcal{O}(n^{\frac{1}{4}+\frac{1}{2}\left(1-\frac{1}{d}\right)}) = \mathcal{O}(n^{1-1/d}).\end{aligned}$$

This finishes the claim, since we have shown that with probability $1 - \exp(-\Omega(\sqrt{n}))$, both the contribution of $E_1, \dots, E_{k_1}$ and $E_{k_1+1}, \dots$ is bounded by $\mathcal{O}(n^{1-1/d})$. □

We complement the bound above by a lower bound on tour lengths of perturbed inputs, making use of the following result by Englert et al. [ERV14] for the one-step model.

Lemma 10.2.3 ([ERV14, Proof of Theorem 1.4])**.** *Let $X_1, \dots, X_n$ be a ϕ-perturbed instance. Then with probability $1 - \exp(-\Omega(n))$, any tour on $X_1, \dots, X_n$ has length at least $\Omega(n^{1-1/d}/\sqrt[d]{\phi})$.*

It also follows from their results that this bound translates to the two-step model consistently with the intuitive correspondence of $\phi \sim \sigma^{-d}$ between the one-step and the two-step model.

Lemma 10.2.4. *Let $X_1, \dots, X_n$ be an instance of points in the unit cube perturbed by Gaussians of standard deviation $\sigma \leq 1$. Then with probability $1 - \exp(-\Omega(n))$ any tour on $X_1, \dots, X_n$ has length at least $\Omega(\sigma n^{1-1/d})$.*

Proof. For readers' convenience, we summarize the arguments of Englert et al. [ERV14, Section 6] who considered truncated Gaussian perturbations: Here, we condition the Gaussian perturbation Z_i for each input point X_i to be contained in $A := [-\alpha, \alpha]^d$ for some $\alpha \geq 1$. Conditioned on this event, the resulting input instance is contained in the cube $C := [-\alpha, 1+\alpha]^d$. By straight-forward calculations, the conditional distribution of each point in C has maximum density bounded by $\mathcal{O}(\alpha^d/\sigma^d)$. Moreover, the probability that the condition fails for a single point is bounded by $\Pr[Z_i \notin A] \leq d\sigma \exp(-\alpha^2/(2\sigma^2))$ for all i. Thus, by choosing $\alpha \geq 1$ sufficiently large, each point has at least constant probability to satisfy the condition $Z_i \in A$.

Given any instance (with Gaussian perturbations which are not truncated), first reveal the (random) subinstance of those points for which the condition $Z_i \in A$ is satisfied and let n' be the number of such points. By the Chernoff bound of Lemma 10.1.2, and $\Pr[Z_i \in A] = \Omega(1)$, we have $n' \geq c \cdot n$ for some $c > 0$ with probability at least $1 - \exp(-\Omega(n))$. If this event occurs, we obtain a random instance of $n' \geq cn$ points and maximum density $\phi = \mathcal{O}(\alpha^d/\sigma^d)$. Hence an application of Lemma 10.2.3 yields that, for some constant $c' > 0$, the probability that a tour of length less than $c' \cdot (n')^{1-1/d}/\sqrt[d]{\phi} = \mathcal{O}((\sigma/\alpha)n^{1-1/d}) = \mathcal{O}(\sigma n^{1-1/d})$ exists is at most $\exp(-\Omega(n)) + \exp(-\Omega(n')) = \exp(-\Omega(n))$. □

Note that Lemmas 10.2.2 and 10.2.4 almost immediately yield the following bound on the approximation performance for the two-step model.[1]

Observation 10.2.5. *Let $X_1, \ldots, X_n$ be an instance of points in the unit cube perturbed by Gaussians of standard deviation $\sigma \leq 1$. Then the approximation performance of 2-Opt is bounded by $\mathcal{O}(1/\sigma)$ in expectation and with probability $1 - \exp(-\Omega(\sqrt{n}))$.*

We remark that this bound is best possible for an analysis of perturbed instances that separately bounds the lengths of any 2-optimal tour from above and gives a lower bound on any optimal tour. To see this, we argue that Lemma 10.2.3, Lemma 10.2.1 (even under ϕ-perturbed input), Lemma 10.2.4 and Lemma 10.2.2 cannot be improved in general. This is straight-forward for Lemma 10.2.3, since n points distributed uniformly at random in a cube of volume $1/\phi$ always have, by scaling and Lemma 10.2.1, a tour of length $\mathcal{O}(n^{1-1/d}/\sqrt[d]{\phi})$. Hence, the lower bound on optimal tours on perturbed instances is tight. To see that the upper bound on any 2-optimal tour is tight, take n uniformly distributed points that have, by Lemma 10.2.3, an optimal tour of length $\Omega(n^{1-1/d})$ with high probability and thus also in expectation.

Naturally, this transfers to the case of Gaussian perturbations, albeit more technical to verify: If we place n identical points in $[0,1]^d$, say at the origin, and perturb them with Gaussians of standard deviation σ, then we may without loss of generality scale the unit cube to $[0, 1/\sigma]^d$ and perturb the points with standard deviation 1 instead. By Lemma 10.2.2, any 2-optimal tour and, thus, any optimal tour on these points has a length of $\mathcal{O}(n^{1-1/d})$ on the scaled instance, since the origins are still contained in the unit cube. Thus, the optimal tour on the original instance has a length of at most $\mathcal{O}(\sigma \cdot n^{1-1/d})$ in expectation and with high probability.

We only sketch that 2-optimal tours can have a length of at least $\Omega(n^{1-1/d})$: We distribute the n (unperturbed) points into $1/\sigma^d$ groups of $\sigma^d n$ points each, and we partition the cube $[0,1]^d$ into $1/\sigma^d$ subcubes of equal side length. Let $c > 0$ be a constant such that with high probability, at least $c\sigma^d n$ points of a group remain in their subcube after perturbation. We call these points successful. Since successful points are identically distributed, conditioned on falling into a compact set, the shortest tour

[1] The large-deviation bound is immediate. For the expected approximation ratio, we additionally make use the worst-case bound of $\mathcal{O}(\log n)$, given in Lemma 10.3.1 below.

through these (at least) $c\sigma^d n$ points has a length of at least $\sigma \cdot c'(\sigma^d n)^{1-1/d} = c'\sigma^d n^{1-1/d}$ for some other constant $c' > 0$ [Yuk98]. (This is just a scaled version of perturbing and truncating a Gaussian of standard deviation 1 to a unit hypercube, which would result in a tour length of $m^{1-1/d}$ for m points.) By closeness of the tour on all points to the boundary functional and geometric superadditivity of the boundary functional (see Yukich [Yuk98] for details), it follows that the optimal tour on all successful points is at least $\Omega\big((1/\sigma^d) \cdot \sigma^d n^{1-1/d}\big) = \Omega(n^{1-1/d})$.

10.3 Upper Bound on the Approximation Performance

In this section, we establish an upper bound on the approximation performance of 2-Opt under Gaussian perturbations. We achieve a bound of $\mathcal{O}(\log 1/\sigma)$. Due to the lower bound presented in Section 10.4, we cannot expect an approximation ratio of $o(\log(1/\sigma)/\log\log(1/\sigma))$. Thus, our bound is almost tight.

As noted in the previous section, to beat $\mathcal{O}(1/\sigma)$ it is essential to exploit the structure of the unperturbed input. This will be achieved by classifying edges of a tour into *long* and *short* edges and bounding the length of long edges by a (worst-case) global argument and short edges locally against the partial optimal tour on subinstances (by a reduction to an (almost-)average case). The local arguments for short edges will exploit how many unperturbed origins lie in the vicinity of a given region.

The global argument bounding long edges follows from the worst-case $\mathcal{O}(\log n)$ bound on the worst-case approximation performance [CKT99] that we rephrase here for our purposes.

Lemma 10.3.1 ([CKT99, Proof of Theorem 4.3])**.** *Let T be a 2-optimal tour and* OPT *denote the length of the optimal traveling salesperson tour T_{OPT}. Let T_i contain the set of all edges in T whose length is in $[\mathrm{OPT}/2^i, \mathrm{OPT}/2^{i-1}]$. Then $L(T_i) = \mathcal{O}(\mathrm{OPT})$. In particular, it follows that $L(T) = \mathcal{O}(\log n) \cdot \mathrm{OPT}$.*

In the proof of our bound of $\mathcal{O}(\log 1/\sigma)$, the above lemma accounts for all edges of length $[\Omega(\sigma), \mathcal{O}(1)]$. A central idea to bound all shorter edges is to apply the one-step model result to small parts of the input space. In particular, we will condition sets of points to be perturbed into cubes of side length σ. The following technical lemma helps to capture what values of ϕ suffice to express the conditional density function of these points depending on the distance of their unperturbed origins to the cube. This allows for appealing to the one-step model result of Lemma 10.2.3.

Lemma 10.3.2. *Let $c \in [0,\sigma]^d$ and $k = (k_1, \ldots, k_d) \in \mathbb{N}_0^d$. Let Y be the random variable $X \sim \mathcal{N}(c, \sigma^2)$ conditioned on $X \in Q := [k_1\sigma, (k_1+1)\sigma] \times \cdots \times [k_d\sigma, (k_d+1)\sigma]$ and f_Y be the corresponding probability density function. Then f_Y is bounded from above by $\exp(\|k\|_1 + (3/2)d)\sigma^{-d}$.*

Proof. Let $f_X(x) = \frac{1}{(2\pi)^{d/2}\sigma^d} \cdot \exp(-\frac{\|x-c\|^2}{2\sigma^2})$ be the probability density function of X. Let $q := \operatorname{argmin}_{z\in Q} \|z - c\|$ be the point in Q that is closest to c. Then, since $f_X(x)$ is rotationally invariant around c and decreasing in $\|x - c\|$, the density $f_X(x)$ inside Q is maximized at $x = q$. Likewise, $q' := \operatorname{argmax}_{z\in Q} \|z - c\|$ minimizes the density inside Q. Since Q is a $(\sigma \times \cdots \times \sigma)$-cube in $\mathbb{R}_{\geq 0}$, $\|q' - c\| \leq \|(q + \sigma\mathbf{1}) - c\|$, where $\mathbf{1} = (1, \ldots, 1) \in \mathbb{R}^d$ denotes the all-ones vector. Given $g(q) := \frac{f_X(q)}{f_X(q+\sigma\mathbf{1})}$, we can thus bound the conditional probability density function f_Y for $x \in Q$ by

$$f_Y(x) = \frac{f_X(x)}{\int_Q f_X(y)dy} \leq \frac{f_X(x)}{f_X(q')\mathrm{vol}(Q)} \leq \frac{f_X(q)}{f_X(q+\sigma\mathbf{1})} \cdot \frac{1}{\sigma^d} = \frac{g(q)}{\sigma^d}.$$

It remains to bound, for $x \in Q$,

$$g(x) = \frac{f(x)}{f(x+\sigma\mathbf{1})} = \exp\left(\frac{\|(x-c)+\sigma\mathbf{1}\|^2 - \|x-c\|^2}{2\sigma^2}\right) = \exp\left(\frac{\|x-c\|_1}{\sigma} + \frac{d}{2}\right).$$

Since for all $x \in Q$, $\|x-c\|_1 \leq (\|k\|_1 + d)\sigma$, we can bound $g(x) \leq \exp(\|k\|_1 + (3/2)d)$, yielding the claim. □

The main result of this section is the following theorem, which will be proved in the remainder of the section.

Theorem 10.3.3. *Let $X = (X_1, \ldots, X_n)$ be an instance of points in $[0,1]^d$ perturbed by Gaussians of standard deviation $\sigma \leq 1$. With probability $1 - \exp(-\Omega(n^{1/2-\varepsilon}))$ for any constant $\varepsilon > 0$, we have $\mathsf{2OPT}(X) = \mathcal{O}(\log(1/\sigma)) \cdot \mathsf{TSP}(X)$. Furthermore,*

$$\mathrm{E}\left[\frac{\mathsf{2OPT}(X)}{\mathsf{TSP}(X)}\right] = \mathcal{O}(\log(1/\sigma)).$$

Since the approximation performance of 2-Opt is bounded by $\mathcal{O}(\log n)$ in the worst-case, we may assume that $1/\sigma = \mathcal{O}(n^{\varepsilon})$ for all constant $\varepsilon > 0$, since otherwise our smoothed result is superseded by Lemma 10.3.1. Furthermore, we may also assume that $1/\sigma = \omega(1)$, since otherwise Observation 10.2.5 already yields the result. In what follows, let T_{OPT} and T be any optimal and longest 2-optimal, respectively, traveling salesperson tour on $X_1, \ldots, X_n$. Furthermore, we let $\mathrm{OPT} = L(T_{\mathrm{OPT}})$ denote the length of the shortest traveling salesperson tour.

10.3.1 Outliers and Long Edges

We will first show that the contribution of almost all points outside $[0,1]^d$ is bounded by $\mathcal{O}(\sigma n^{1-1/d})$ with high probability and in expectation, similar to Lemma 10.2.2. For this, we define growing cubes $A_i := [-a_i, 1+a_i]^d$, where we set $a_i := 3\sigma\sqrt{di\ln(3/\sigma)}$ for $i \geq 1$ and $A_0 = [0,1]^d$. Let n_i be the number of points not contained in A_{i-1}. For every point X_j, Lemma 10.1.1 with $t := (3/\sigma)^i$ bounds $\Pr[X_j \notin A_{i-1}] \leq (\sigma/3)^{2.9d(i-1)}$ (note that we have chosen the a_i such that $t \geq 3$). Thus, $\mathrm{E}[n_i] \leq n(\sigma/3)^{2.9d(i-1)}$. We define E_i as the set of edges of the longest 2-optimal tour T contained in A_i with at least one endpoint in $A_i \setminus A_{i-1}$. We first bound the contribution of the E_i with $i \geq 2$.

Lemma 10.3.4. *With probability $1 - \exp(-\Omega(n^{1/2-\varepsilon}))$ for any constant $\varepsilon > 0$, we have $\sum_{i=2}^{\infty} L(E_i) = \mathcal{O}(\sigma n^{1-1/d})$. Additionally, $\mathrm{E}[\sum_{i=2}^{\infty} L(E_i)] = \mathcal{O}(\sigma n^{1-1/d})$.*

Proof. The proof is analogous to the proof of Lemma 10.2.2. By linearity of expectation, Lemma 10.2.1 and Jensen's inequality, we have

$$\begin{aligned}
\sum_{i=2}^{\infty} \mathrm{E}[L(E_i)] &\leq \sum_{i=2}^{\infty} c_d \cdot (3\,\mathrm{E}[n_i])^{1-1/d}(1+2a_i) \\
&\leq \sum_{i=2}^{\infty} 3c_d \cdot n^{1-1/d}\left(\frac{\sigma}{3}\right)^{2.9(d-1)(i-1)}\left(1 + 6\sigma\sqrt{i\ln(3/\sigma)}\right) \\
&\leq 3c_d \cdot n^{1-1/d}\left(\frac{\sigma}{3}\right)^{2.9(d-1)}(1+6\sigma\sqrt{\ln(3/\sigma)})\left(\sum_{i=0}^{\infty}\sqrt{i+2}\left(\frac{\sigma}{3}\right)^{2.9(d-1)i}\right).
\end{aligned}$$

By noting that $\sum_{i=0}^{\infty} \sqrt{i+2}(\sigma/3)^{2.9(d-1)i}$ is bounded by a constant, we conclude that $\sum_{i=2}^{\infty} \mathrm{E}[L(E_i)]$ is bounded by $\mathcal{O}(\sigma n^{1-1/d})$.

Let $N_k := n(\sigma/3)^{2.9d(k-1)}$ be the upper bound on $\mathrm{E}[n_k]$ derived above. By the Chernoff bounds of Lemma 10.1.2, we have

$$\Pr[n_k \geq 2N_k] \leq \exp(-N_k/3).$$

Choose k_1 such that $(\sigma/3)^{2.9d}\sigma\sqrt{n} \leq N_{k_1} \leq \sigma\sqrt{n}$. Thus, $k_1 = \mathcal{O}(\log n)$. Assume that $n_k \leq 2N_k$ for all $1 \leq k \leq k_1$. Then, analogously to the above calculation, the contribution of $E_2, \ldots, E_{k_1}$ is bounded by

$$\begin{aligned}\sum_{k=2}^{k_1} L(E_k) &\leq \sum_{k=2}^{k_1} c_d \cdot (1+2a_k)(6N_k)^{1-1/d} \\ &\leq c_d \cdot \left(1 + 6\sigma\sqrt{d\ln(3/\sigma)}\right) \cdot (6n)^{1-1/d}(\sigma/3)^{2.9(d-1)} \\ &\quad \cdot \left(\sum_{k=0}^{\infty} \sqrt{k+2}(\sigma/3)^{2.9(d-1)k}\right) = \mathcal{O}(\sigma n^{1-1/d}).\end{aligned}$$

Note that the probability that some $1 \leq k \leq k_1$ fails to satisfy $n_k \leq 2N_k$ is bounded by

$$\sum_{k=1}^{k_1} \Pr[n_k > 2N_k] \leq k_1 \exp(-N_{k_1}/3) = \exp(-\Omega(n^{1/2-\varepsilon})),$$

for any constant $\varepsilon > 0$. Since $n_{k_1} \leq 2N_{k_1}$, at most $n_{k_1} \leq 2\sigma\sqrt{n}$ vertices remain outside A_{k_1-1}. Let $k_2 := \lceil \sigma\sqrt{n} \rceil$. By a union bound, for any constant $\varepsilon > 0$,

$$\Pr[\exists j : X_j \notin A_{k_2}] \leq n(\sigma/3)^{2.9d(k_2-1)} = \exp(-\Omega(n^{1/2-\varepsilon})).$$

Assume that we have the – very likely – event that all points are in A_{k_2}, then the remaining points outside A_{k_1-1} are contained in A_{k_2}. We conclude that

$$\begin{aligned}\sum_{k=k_1}^{\infty} L(E_k) &= \sum_{k=k_1}^{k_2} L(E_k) \\ &\leq c_d \cdot (1+2a_{k_2})(6N_{k_1})^{1-1/d} \\ &= \mathcal{O}(\sqrt{k_2}(N_{k_1})^{1-1/d}) \\ &= \mathcal{O}(\sigma^{3/2-1/d} n^{\frac{1}{4}+\frac{1}{2}(1-\frac{1}{d})}) = \mathcal{O}(\sigma n^{1-1/d}).\end{aligned}$$

□

In the remainder of the proof, we bound the total length of edges inside A_1. Define $C := A_1$ and note that all edges in C have bounded length $\sqrt{d}(1+2a_1) = \mathcal{O}(1)$. We let T_i contain the set of all those edges within C (in the longest 2-optimal tour T) whose lengths are in $[\mathrm{OPT}/2^i, \mathrm{OPT}/2^{i-1}]$. Let k_1 be such that $\sqrt{d}(1+2a_1) \in [\mathrm{OPT}/2^{k_1}, \mathrm{OPT}/2^{k_1-1}]$. Then $L(T_k) = 0$ for all $k < k_1$, since no longer edges exist. Let k_2 be such that $\sigma \in [\mathrm{OPT}/2^{k_2}, \mathrm{OPT}/2^{k_2-1}]$. Then $\sum_{k=k_1}^{k_2} L(T_k) = \mathcal{O}((k_2-k_1)\cdot\mathrm{OPT}) = \mathcal{O}(\log(1/\sigma)\mathrm{OPT})$ by Lemma 10.3.1. This argument bounds the contribution of *long edges*, i.e., edges longer than σ, in the worst case, after observing the perturbation of the input points. It remains to bound the length of short edges in C.

10.3.2 Short Edges

To account for the length of the remaining edges, we take a different route than for the long edges: Call an edge that is shorter than σ a *short edge* and partition the bounding box $C = [-a_1, 1 + a_1]^d$ into a grid of $(\sigma \times \cdots \times \sigma)$-cubes $C_1, \dots, C_M$ with $M = \Theta((\sigma/(1 + a_1))^{-d}) = \Theta(\sigma^{-d})$, which we call *cells*. All edges in T_k for $k \geq k_2$, i.e., short edges, are completely contained in a single cell or run from some cell C_i to one of its $3^d - 1$ neighboring cells. For a given tour T, let $E_{C_i}(T)$ denote the short edges of T for which at least one of the endpoints lies in C_i.

We aim to relate the length of the edges $E_{C_i}(T)$ for the longest 2-optimal tour T to the length of the edges $E_{C_i}(T_{\mathrm{OPT}})$ of the optimal tour T_{OPT}. This local approach is justified by the following property.

Lemma 10.3.5. *For any tour T', the contribution $L(E_{C_i}(T'))$ of cell C_i is lower bounded by $\mathsf{TSP}(X \cap C_i) - \mathcal{O}(\sigma |X \cap C_i|^{\frac{d-2}{d-1}})$.*

Proof. Consider all edges S in T' that have at least one endpoint in C_i. Replacing those edges $(u, v) \in S$ with $u \in C_i$ and $v \notin C_i$ by the shortest edge connecting u to the boundary of C_i does not increase the total edge length by triangle inequality. If C_i were the unit cube, $L(E_{C_i}(T'))$ would thus be lower bounded by the boundary functional $\mathsf{TSP}^B(X \cap C_i)$. Instead, we scale the instance $X \cap C_i$ by $1/\sigma$ to obtain an instance X' in the unit cube, satisfying $\mathsf{TSP}(X \cap C_i) = \sigma \mathsf{TSP}(X')$ and, as argued above, $L(E_{C_i}(T')) \geq \sigma \mathsf{TSP}^B(X')$. Thus an application of Lemma 10.1.3 yields

$$L(E_{C_i}(T')) \geq \sigma \left(\mathsf{TSP}(X') - \mathcal{O}\big(|X'|^{\frac{d-2}{d-1}}\big) \right) = \mathsf{TSP}(X \cap C_i) - \mathcal{O}\big(\sigma \cdot |X \cap C_i|^{\frac{d-2}{d-1}}\big). \quad \square$$

Intuitively, a cell C_i is of one of two kinds: either few points are expected to be perturbed into it and hence it cannot contribute much to the length of any 2-optimal tour (a *sparse cell*), or many unperturbed origins are close to the cell (a *heavy cell*). In the latter case, either the conditional densities of points perturbed into C_i are small, hence any optimal tour inside C_i has a large value by Lemma 10.2.3, or we find another cell close to C_i that has a very large contribution to the length of any tour.

To formalize this intuition, fix a cell C_i and let n_i be the expected number of points X_j with $X_j \in C_i$. Assume for convenience that a_1/σ and $(1 + a_1)/\sigma$ are integer. We describe the position of a cube C_i canonically by indices $\mathrm{pos}(C_i) \in \{-\frac{a_i}{\sigma}, \dots, \frac{1+a_i}{\sigma}\}^d$. For two cells C_i and C_j, we define their distance as $\mathrm{dist}(C_i, C_j) = \|\mathrm{pos}(C_i) - \mathrm{pos}(C_j)\|_1$. For $k \geq 0$, let D_k denote all cells of distance k to C_i and let $n(D_k)$ denote the cardinality of unperturbed origins located in a cell in D_k. We call a perturbed point $X_\ell \in C_i$ with unperturbed origin $x_\ell \in C_j$, for some $C_j \in D_k$, a *k-successful point*. Let S_k denote the set of all k-successful points. Then $n_i = \sum_{k=0}^{\infty} \mathrm{E}[|S_k|]$.

Our first technical lemma shows that any cell C_i, having (in expectation) a large number μ of points perturbed into it from cells of distance at most K, contributes at least $\sigma \mu^{1-1/d} \exp(-\mathcal{O}(K + 1))$ to the length of the optimal tour.

Lemma 10.3.6. *Let $K \geq 0$ and define $S_{\leq K} := S_0 \cup \cdots \cup S_K$ as the set of k-successful points for $k \leq K$. Let $\mu := \mathrm{E}[|S_{\leq K}|]$. If $K = o(\log \mu)$, then with probability $1 - \exp(-\Omega(\mu))$, we have*

$$L(E_{C_i}(T_{\mathrm{OPT}})) \geq \frac{\sigma \mu^{1-1/d}}{\exp(\mathcal{O}(K + 1))}.$$

Proof. Note that by Lemma 10.3.5, $L(E_{C_i}(T_{\mathrm{OPT}})) \geq \mathsf{TSP}(S_{\leq K}) - \mathcal{O}(\sigma \cdot |S_{\leq K}|^{\frac{d-2}{d-1}})$. Fix any realization of $S_{\leq K}$, i.e., choice of unperturbed origins inside some cell in $D_0, \dots, D_K$

whose perturbed points fall into C_i. We can simulate the distribution of $\mathsf{TSP}(S_{\leq K})$ (under this realization of $S_{\leq K}$) by appealing to the one-step model. Note that each point in $S_{\leq K}$ is distributed as a Gaussian conditioned on containment in cell C_j. By rotational invariance of the Gaussian distribution, Lemma 10.3.2 is applicable and bounds the conditional density function of each point in $S_{\leq K}$ by $\exp(K + (3/2)d)\sigma^{-d}$. By scaling, we obtain an instance in the unit cube with $N := |S_{\leq K}|$ points distributed according to density functions of maximum density $\exp(K + (3/2)d)$. Hence, by Lemma 10.2.3 we obtain that any tour has length $\Omega(N^{1-1/d}/\exp(K/d + 3/2))$ on the scaled instance with probability $1-\exp(-\Omega(N))$. Scaling back to C_i, we obtain $\mathsf{TSP}(S_{\leq K}) \geq \Omega(\sigma N^{1-1/d}/\exp(K/d+3/2))$. Since by Chernoff bounds (Lemma 10.1.2), $|S_{\leq K}| = \Omega(\mu)$ with probability $1-\exp(-\Omega(\mu))$, we finally obtain, using Lemma 10.3.5,

$$L(E_{C_i}(T_{\mathrm{OPT}})) \geq \Omega\left(\frac{\sigma\mu^{1-\frac{1}{d}}}{\exp(\frac{K}{d}+\frac{3}{2})}\right) - \mathcal{O}(\sigma\cdot\mu^{\frac{d-2}{d-1}}) \geq \frac{\sigma\mu^{1-1/d}}{\exp(\mathcal{O}(K+1))},$$

with probability $1-\exp(-\Omega(\mu))$, where we used that $K = o(\log\mu)$. □

The following simple technical lemma shows that with constant probability, a point is perturbed into the cell it originates in.

Lemma 10.3.7. *Let $c \in Q_0 := [0,\sigma]^d$ and $Z \sim \mathcal{N}(c,\sigma^2)$. Then $\Pr[Z \in Q_0] \geq \frac{1}{(2\pi)^{d/2}}\exp(-\frac{d}{2})$.*

Proof. Let $f(x) = \frac{1}{(2\pi)^{d/2}\sigma^d}\cdot\exp(-\frac{\|x-c\|^2}{2\sigma^2})$ be the probability density function of Z. For all $x \in Q_0$, we have $\|x-c\| \leq \sqrt{d}\sigma$ and hence $f(x) \geq \frac{1}{(2\pi)^{d/2}\sigma^d}\cdot\exp(-\frac{d}{2}) =: f_{\min}$. This yields

$$\Pr[Z\in Q_0] = \int_{Q_0} f(x)dx \geq \sigma^d f_{\min} = \frac{1}{(2\pi)^{d/2}}\exp(-d/2).$$ □

We are set-up to formally show the classification of heavy cells. Recall that $M = \Theta(\sigma^{-d})$ denotes the number of cells C_i.

Lemma 10.3.8. *Let $\alpha := M^{\frac{d}{d-1}}$, $k_1 := \gamma\log\log(1/\sigma)$ and $k_2 := (1/\gamma')\sqrt{\log 1/\sigma}$ for sufficiently small constants γ,γ'. Then we can classify each cell C_i with $n_i \geq \frac{n}{\alpha}$ into one of the following two types.*

(T1) With probability $1-\exp(-\Omega(n^{1-\varepsilon}))$ for any constant $\varepsilon > 0$, we have

$$L(E_{C_i}(T)) \leq \mathcal{O}(\log 1/\sigma)L(E_{C_i}(T_{\mathrm{OPT}})).$$

(T2) There is some $C_j \in D_{k_1}\cup\cdots\cup D_{k_2}$ such that for any $f(1/\sigma) = \mathrm{polylog}(1/\sigma)$, we have

$$L(E_{C_i}(T)) \leq \frac{L(E_{C_j}(T_{\mathrm{OPT}}))}{f(1/\sigma)},$$

with probability $1-\exp(-\Omega(n^{1-\varepsilon}))$ for any constant $\varepsilon > 0$.

Proof. We start with some intuition. By Lemma 10.2.1, we can bound $L(E_{C_i}(T)) = \mathcal{O}(\sigma n_i^{1-1/d})$. If we have $\mathrm{E}[|S_{\leq k_1}|] = \Omega(n_i)$, then Lemma 10.3.6 already proves C_i to have type (T1). Otherwise, by tail bounds for the Gaussian distribution, we argue that some cell C_j in distance at most k_2 contains at least $n_i\exp(\Omega((\log\log 1/\sigma)^2))$

unperturbed origins. These are sufficiently many to let C_j contribute $f(1/\sigma)\sigma n_i^{1-1/d}$, for any $f(1/\sigma) = \text{polylog}(1/\sigma)$, to the optimal tour length.

To make the intuition formal, note that all edges in $E_{C_i}(T)$ are contained in a cube of side length 3σ around C_i. By Chernoff bounds (Lemma 10.1.2), at most $2n_i$ points are contained in C_i with probability $1 - \exp(-\Omega(n_i))$. Hence, Lemma 10.2.1 bounds

$$L(E_{C_i}(T)) \leq 3\sigma c_d (6n_i)^{1-1/d}, \tag{10.1}$$

with probability $1 - \exp(-\Omega(n_i))$.

Case 1: $\mathrm{E}[S_{\leq k_1}] > n_i/2$**.** In this case, we may appeal to Lemma 10.3.6 (since $k_1 = o(\log n_i)$) and obtain

$$L(E_{C_i}(T_{\text{OPT}})) \geq \frac{\sigma(|S_{\leq k_1}|)^{1-1/d}}{\exp(\mathcal{O}(k_1))} = \Omega\left(\frac{\sigma n_i^{1-1/d}}{\log(1/\sigma)}\right), \tag{10.2}$$

with probability $1-\exp(-\Omega(n_i))$, since $k_1 = \gamma \log\log 1/\sigma$ and γ can be chosen sufficiently small. By union bound, (10.1) and (10.2) hold with probability $1 - \exp(-\Omega(n_i)) = 1 - \exp(-\Omega(n^{1-\varepsilon}))$ for any constant $\varepsilon > 0$, proving that C_i has type (T1).

Case 2: $\mathrm{E}[S_{\leq k_1}] \leq n_i/2$**.** Since every point in C_i has an ℓ_1-distance of at least $\sigma(\text{dist}(C_i, C_j) - d)$ to every point in C_j, we have, by Lemma 10.1.1, that

$$\mathrm{E}[|S_k|] \leq n(D_k) \Pr\left[\|Z\| \geq \frac{k-d}{\sqrt{d}}\sigma\right] \leq n(D_k) \exp\left(-0.32\frac{(k-d)^2}{d}\right), \tag{10.3}$$

for sufficiently large k.

Since $\alpha = \text{poly}(1/\sigma)$, we can choose a sufficiently small constant γ' such that $k_2 = (1/\gamma')\sqrt{\log 1/\sigma}$ satisfies $\exp(-0.32(k_2-d)^2/d) \leq 1/(4\alpha)$. From $\sum_{k=0}^{\infty} n(D_k) = n$, we conclude

$$\sum_{k=k_2+1}^{\infty} \mathrm{E}[|S_k|] \leq \sum_{k=k_2+1}^{\infty} n(D_k) \exp(-0.32(k-d)^2/d) \leq \frac{n}{4\alpha} \leq \frac{n_i}{4}.$$

Hence, we have

$$\sum_{k=k_1+1}^{k_2} \mathrm{E}[|S_k|] = n_i - \mathrm{E}[|S_{\leq k_1}|] - \sum_{k=k_2+1}^{\infty} \mathrm{E}[|S_k|] \geq \frac{n_i}{4}.$$

By (10.3), it follows that

$$N := \sum_{k=k_1+1}^{k_2} n(D_k) \geq \exp\left(0.32\frac{(k_1-d)^2}{d}\right) \sum_{k=k_1+1}^{k_2} \mathrm{E}[|S_k|] = n_i \exp(\Omega((\log\log 1/\sigma)^2))$$

unperturbed origins are situated in cells in distance $k_1 < k \leq k_2$ from C_i. Note that there are at most $\sum_{k=k_1+1}^{k_2} |D_k| = \mathcal{O}(k_2^d) = \text{polylog}(1/\sigma)$ such cells and $\exp(\Omega((\log\log 1/\sigma)^2) = \omega(\log^c(1/\sigma))$ for any $c \in \mathbb{N}$. By pigeon hole principle, there is a cell $C_j \in D_{k_1} \cup \cdots \cup D_{k_2}$ with $\Omega(N/k_2^d) = n_i \exp(\Omega((\log\log 1/\sigma)^2))$ many unperturbed origins.

Let S_0' be the 0-successful points for cell C_j, i.e., the points with origin in C_j that are perturbed into C_j. By Lemma 10.3.7, each unperturbed origin $x_\ell \in C_j$

has constant probability to be perturbed into C_j, i.e., $\Pr[X_\ell \in C_j] = \Omega(1)$. Hence, $\mathrm{E}[|S'_0|] = n_i \exp(\Omega((\log\log 1/\sigma)^2))$. Thus, Lemma 10.3.6 bounds

$$L(E_{C_j}(T_{\mathrm{OPT}})) \geq \frac{\sigma(\mathrm{E}[|S'_0|])^{1-\frac{1}{d}}}{\exp(\mathcal{O}(1))} = \sigma n_i^{1-\frac{1}{d}} \exp(\Omega((\log\log 1/\sigma)^2)), \tag{10.4}$$

with probability $1 - \exp(-\Omega(\mathrm{E}[|S'_0|])) = 1 - \exp(-\Omega(n_i))$. Since (10.1) and (10.4) hold simultaneously with probability $1 - \exp(-\Omega(n_i)) = 1 - \exp(-\Omega(n^{1-\varepsilon}))$ for any constant $\varepsilon > 0$, this proves that C_i has type (T2).

□

10.3.3 Total Length of 2-Optimal Tours

With the analyses of the previous subsections, we can finally bound the total length of 2-optimal tours. To bound the total length of short edges, consider first sparse cells C_i, i.e., cells containing $n_i \leq n/\alpha$ perturbed points in expectation (recall that $\alpha = M^{\frac{d}{d-1}}$, where $M = \Theta(\sigma^{-d})$ is the number of cells). For each such cell, the Chernoff bound of Lemma 10.1.2 yields that with probability $1 - \exp(-\Omega(n/\alpha))$, at most $2n/\alpha$ points are contained in C_i, since each point is perturbed independently. By union bound, no sparse cell contains more than $2n/\alpha$ points with probability at least $1 - M\exp(-\Omega(n/\alpha)) = 1 - \exp(-\Omega(n^{1-\varepsilon}))$ for any constant $\varepsilon > 0$. In this event, Lemma 10.2.1 allows for bounding the contribution of sparse cells by

$$\sum_{i:n_i \leq n/\alpha} L(E_{C_i}(T)) \leq M(3\sigma)c_d \left(\frac{6n}{\alpha}\right)^{1-\frac{1}{d}} = \mathcal{O}\left(\frac{M\sigma n^{1-\frac{1}{d}}}{\alpha^{1-\frac{1}{d}}}\right) = \mathcal{O}(\sigma n^{1-\frac{1}{d}}). \tag{10.5}$$

For bounding the length in the remaining cells, the heavy cells, let $\mathcal{T}_1 := \{i \mid C_i \text{ has type (T1)}\}$ and $\mathcal{T}_2 := \{i \mid C_i \text{ has type (T2)}\}$. We observe that with probability at least $1 - M\exp(-\Omega(n^{1-\varepsilon})) = 1 - \exp(-\Omega(n^{1-\varepsilon}))$, all type-(T1) cells C_i satisfy $L(E_{C_i}(T)) = \mathcal{O}(\log 1/\sigma) L(E_{C_i}(T_{\mathrm{OPT}}))$. Thus,

$$\sum_{i\in\mathcal{T}_1} L(E_{C_i}(T)) \leq \mathcal{O}(\log 1/\sigma) \cdot \left(\sum_{i\in\mathcal{T}_1} L(E_{C_i}(T_{\mathrm{OPT}}))\right) \leq \mathcal{O}(\log 1/\sigma)\mathrm{OPT}, \tag{10.6}$$

where the last inequality follows from $\sum_{i=1}^{M} L_{C_i}(T_{\mathrm{OPT}}) \leq 2 \cdot \mathrm{OPT}$, which holds since every edge in OPT (inside C) is counted at most twice on the left-hand side.

Let $A : \mathcal{T}_2 \to \{1, \ldots, M\}$ be any function that assigns to each type-(T2) cell C_i a corresponding cell $C_{A(i)} \in D_{k_1} \cup \cdots \cup D_{k_2}$ satisfying the condition (T2). We say that C_i *charges* $C_{A(i)}$. We can choose any $f(1/\sigma) = \mathrm{polylog}(1/\sigma)$ and have with probability at least $1 - M\exp(-\Omega(n^{1-\varepsilon})) = 1 - \exp(-\Omega(n^{1-\varepsilon}))$ that $L(E_{C_i}(T)) \leq \frac{L(E_{C_{A(i)}}(T_{\mathrm{OPT}}))}{f(1/\sigma)}$ for all $i \in \mathcal{T}_2$. Assume that this event occurs. Since every cell C_i can only be charged by cells in distance $k_1 < k \leq k_2$, each cell can only be charged $\sum_{k=k_1+1}^{k_2} |D_k| = \mathcal{O}(k_2^d)$ times. Hence,

$$\sum_{i\in\mathcal{T}_2} L(E_{C_{A(i)}}(T_{\mathrm{OPT}})) \leq \mathcal{O}(k_2^d) \sum_{i=1}^{M} L(E_{C_i}(T_{\mathrm{OPT}})) = \mathcal{O}(k_2^d)\mathrm{OPT}.$$

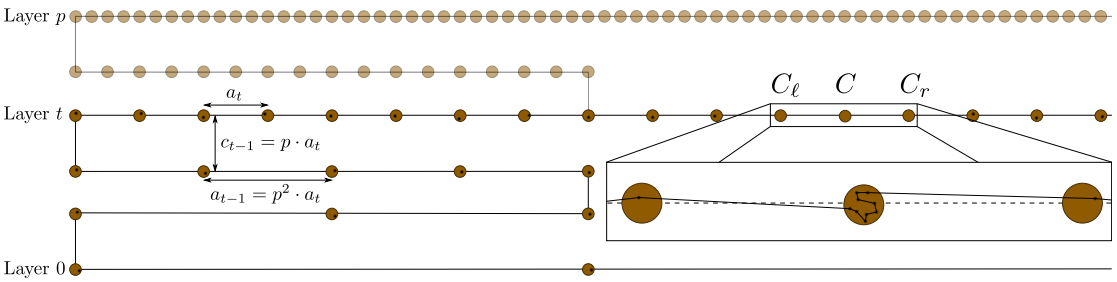


FIGURE 10.1: Parts V_1 and V_3 of the lower bound instance. Each point is contained in a corresponding small container (depicted as brown circle) with high probability. The black lines indicate the constructed 2-optimal tour, which on V_2 runs analogously.

Since $k_2^d = \text{polylog}(1/\sigma)$, choosing $f(1/\sigma) = \text{polylog}(1/\sigma)$ sufficiently large yields

$$\sum_{i \in \mathcal{T}_2} L(E_{C_i}(T)) \leq \sum_{i \in \mathcal{T}_2} \frac{L(E_{C_{A(i)}}(T_{\text{OPT}}))}{f(1/\sigma)} \leq \frac{\mathcal{O}(k_2^d)\text{OPT}}{f(1/\sigma)} = \mathcal{O}(\text{OPT}). \tag{10.7}$$

Proof of Theorem 10.3.3. By a union bound, we can bound by $1 - \exp(-\Omega(n^{1/2-\varepsilon}))$, for any constant $\varepsilon > 0$, the probability that (i) $\text{OPT} = \Omega(\sigma n^{1-1/d})$ (by Lemma 10.2.4), (ii) all edges outside C contribute $\mathcal{O}(\sigma n^{1-1/d}) = \mathcal{O}(\text{OPT})$ (by Lemma 10.3.4), (iii) all sparse cells contribute $\mathcal{O}(\sigma n^{1-1/d}) = \mathcal{O}(\text{OPT})$ (by (10.5)), (iv) the type-(T1) cells C_i induce a cost of $\mathcal{O}(\log 1/\sigma)\text{OPT}$ (by (10.6)), and (v) the type-(T2) cells induce a cost of $\mathcal{O}(\text{OPT})$ (by (10.7)). Since the remaining edges are long edges and contribute only $\mathcal{O}(\log(1/\sigma) \cdot \text{OPT})$, we obtain that every 2-optimal tour has a length of at most $\mathcal{O}(\log 1/\sigma)\text{OPT}$ with probability $1 - \exp(-\Omega(n^{1/2-\varepsilon}))$.

Since a 2-optimal tour always constitutes a $\mathcal{O}(\log n)$-approximation to the optimal tour length by Lemma 10.3.1, we also obtain that the expected cost of the worst 2-optimal tour is bounded by

$$\mathcal{O}(\log 1/\sigma) \cdot \text{OPT} + \exp(-\Omega(n^{1/2-\varepsilon})) \cdot \mathcal{O}(\log n) \cdot \text{OPT} = \mathcal{O}(\log 1/\sigma) \cdot \text{OPT}. \qquad \square$$

10.4 Lower Bound on the Approximation Ratio

We complement our upper bound on the approximation performance by the following lower bound: for $\sigma = \mathcal{O}(1/\sqrt{n})$, the worst-case lower bound is robust against perturbations. For this, we face the technical difficulty that in general, a single outlier might destroy the 2-optimality of a desired long tour, potentially cascading into a series of 2-Opt iterations that result in a substantially different or even optimal tour.

Theorem 10.4.1. *Let $\sigma = \mathcal{O}(1/\sqrt{n})$. For infinitely many n, there is an instance X of points in $\mathbb{R}^2$ perturbed by normally distributed noise of standard deviation σ such that with probability $1 - \mathcal{O}(n^{-s})$ for any constant $s > 0$, we have $\mathsf{2OPT}(X) = \Omega(\log n / \log\log n) \cdot \mathsf{TSP}(X)$. This also yields*

$$\mathrm{E}\left[\frac{\mathsf{2OPT}(X)}{\mathsf{TSP}(X)}\right] = \Omega\left(\frac{\log n}{\log\log n}\right).$$

We remark that our result transfers naturally to the one-step model with $\phi = \Omega(n)$ and interestingly, holds *with probability 1* over such random perturbations.

Furthermore, even when we initialize the tour using the *nearest neighbor heuristic*, 2-Opt might, with probability $\mathcal{O}(1)$, return a 2-optimal tour of length $\Omega(\log n/\log\log n)\cdot \mathsf{TSP}(X)$ on perturbed inputs. We omit the necessary modifications to our construction here; they will be subject of future work.

Proof of Theorem 10.4.1. We alter the construction of Chandra et al. [CKT99] to strengthen it against Gaussian perturbations with standard deviation $\sigma = \mathcal{O}(1/\sqrt{n})$ (see Figure 10.1). Let $p \geq 3$ be an odd integer and $P := 3p^{2p}$. The original instance of [CKT99] is a subset of the $(P \times P)$-grid, which we embed into $[0,1]^2$ by scaling by $1/P$, and consists of three parts V_1, V_2 and V_3. The vertices in V_1 are partitioned into the layers $L_0, \ldots, L_p$. Layer i consists of $p^{2i}+1$ equidistant vertices, each of which has a vertical distance of $c_i = p^{2p-2i-1}/P$ to the point above it in Layer $i+1$ and a horizontal distance of $a_i = p^{2p-2i}/P$ to the nearest neighbor(s) in the same layer. The set V_2 is a copy of V_1 shifted to the right by a distance of $2/3$. The remaining part V_3 consists of a copy of Layer p of V_1 shifted to the right by $1/3$ to connect V_1 and V_2 by a path of points. We regard L_i as the set of Layer-i points in $V_1 \cup V_2 \cup V_3$.

As in the original construction, we will construct an instance of $n = \Theta(p^{2p})$ points, which implies $p = \Theta(\log n/\log\log n)$. Let $0 \leq t \leq p$ be the largest odd integer such that $p^{2t+1} \leq (3\sigma)^{-1}$. In our construction, we drop all Layers $t+1, \ldots, p$ in both V_1 and V_2, as well as Layer p in V_3. Instead, we connect V_1 and V_2 already in Layer t by an altered copy of Layer t of V_1 shifted to the right by $1/3$. Let C be an arbitrary point of our construction, for convenience we will use the central point of Layer t in V_3. We introduce $p^{2p}-1$ additional copies of this point C. These surplus points serve as a "padding" of the instance to ensure $n = \Theta(p^{2p})$. Note that the resulting instance has $t+1$ layers $L_0, \ldots, L_t$. We chose t such that the magnitude of perturbation is negligible compared to the pairwise distances of all non-padding points. Furthermore, the restriction on σ ensures that incorporating the padding points increases the optimal tour length only by a constant.

Lemma 10.4.2. *With probability $1-\mathcal{O}(n^{-s})$ for any constant $s>0$, the optimal tour has length $\mathcal{O}(1)$.*

Proof. Let n be the number of points in the constructed instance. Note that $\overline{X} = (x_1, \ldots, x_n)$ consists of (i) a subset $\overline{X}^{\text{orig}}$ of the instance of Chandra et al. [CKT99], plus (ii) an additional copy $\overline{X}^t$ of Layer t and (iii) the padding points $\overline{X}^{\text{pad}}$ in V_3. Denote the number of points in $\overline{X}^{\text{orig}} \cup \overline{X}^t$ by n'. We have

$$n' = p^{2t} + 2\left(\sum_{i=0}^{t} p^{2i} + 1\right) \leq p^{2t} + 2(1-p^{-2})^{-1}p^{2t} + 2t = \mathcal{O}(\sigma^{-1}/p),$$

by choice of t. Hence $n = (p^{2p}-1)+n' = \Theta(p^{2p})$. It is easy to see [CKT99] that the original instance of Chandra et al. has a minimum spanning tree of length $\mathsf{MST}(\overline{X}^{\text{orig}}) \leq 9p^{2p}/P$. (This is achieved by the spanning tree that includes, for each Layer-i vertex with $0 \leq i < p$, the vertical edge to the point above it, and each edge between consecutive points on Layer p.) Clearly,

$$\mathsf{MST}(\overline{X}^{\text{orig}} \cup \overline{X}^t) \leq \mathsf{MST}(\overline{X}^{\text{orig}}) + \mathsf{MST}(\overline{X}^t) + 2a_t \leq 9p^{2p}/P + p^{2p}/P + 2a_t = \mathcal{O}(1).$$

Consider the perturbed instance $X \leftarrow \text{pert}_\sigma(\overline{X})$. Note that for every constant $s>0$, we have $p\sigma \geq 3\sigma\sqrt{d(s+1)\ln n}$ for sufficiently large n. Thus for each $1 \leq i \leq n$, the

Gaussian noise $Z_i \sim \mathcal{N}(0, \sigma)$ satisfies $\|Z_i\| \leq p\sigma$ with probability at least $1 - \mathcal{O}(n^{-s+1})$ by Lemma 10.1.1. By a union bound, we have $\sum_{i=1}^{n'} \|Z_i\| \leq \mathcal{O}(n'p\sigma) = \mathcal{O}(1)$ with probability at least $1 - \mathcal{O}(n^{-s})$. In this case, by the triangle inequality, the fact that $\mathsf{TSP}(Y) \leq 2 \cdot \mathsf{MST}(Y)$ for all point sets Y and since only a constant number of edges connects the three parts, we obtain

$$\begin{aligned}\mathsf{TSP}(X_1, \ldots, X_n) &\leq 2 \cdot \mathsf{MST}(\overline{X}^{\text{orig}} \cup \overline{X}^t) + 2\left(\sum_{i=1}^{n'} \|Z_i\|\right) + \mathsf{TSP}(X^{\text{pad}}) + \mathcal{O}(1) \\ &\leq \mathsf{TSP}(X^{\text{pad}}) + \mathcal{O}(1).\end{aligned}$$

Note that we may translate and scale $\overline{X}^{\text{pad}}$ to be contained in $[0, \sigma]^d$, by which $\mathsf{TSP}(X^{\text{pad}})$ may be regarded as the optimal tour length on an instance of $p^{2p} = \Theta(n)$ points in $[0, 1]^d$ perturbed by Gaussians with standard deviation 1. By Lemma 10.2.2, any 2-optimal tour and hence also the optimal tour on the scaled instance has length $\mathcal{O}(\sqrt{n})$ with probability $1 - \exp(-\Omega(\sqrt{n}))$. Scaling back to the original instance, we obtain $\mathsf{TSP}(X^{\text{pad}}) = \mathcal{O}(\sqrt{n}\sigma) = \mathcal{O}(1)$ with probability $1 - \exp(-\Omega(\sqrt{n}))$. This yields the result by a union bound. □

We find a long 2-optimal tour on all non-padding points analogously to the original construction by taking a shortcut of the original 2-optimal tour, which connects V_1 and V_2 already in Layer t (see Figure 10.1).

Consider the padding points, which are yet to be connected. Let C_ℓ denote the nearest point in Layer t of V_3 that is to the left of C. Symmetrically, C_r is the nearest point to the right of C. Let T^p be any 2-optimal path from C_ℓ to C_r that passes through all the padding points (including C). We replace the edges (C_ℓ, C) and (C, C_r) by the path T^p, completing the construction of our tour T.

Lemma 10.4.3. *Let $s > 0$ be arbitrary. With probability $1 - \mathcal{O}(n^{-s})$, T is 2-optimal and has a length of $\Omega(\log n / \log\log n)$.*

Note that given Lemma 10.4.3, Theorem 13.3.1 follows directly using Lemma 10.4.2. The (rather technical) proof of Lemma 10.4.3 hence concludes our lower bound.

Probability of 2-optimality

To account for the perturbation in the analysis, we define a safe region for every point. More formally, let x_j be any unperturbed origin. We define its *container* B_j as the circle centered at x_j with radius $\beta := a_t/8 = p^{2p-2t}/(8P) \geq \sigma p/8$. Very likely, all perturbed points lie in their containers.

Lemma 10.4.4. *For sufficiently large p, the tour T constructed as described in Section 10.4 is 2-optimal, provided that all points X_j lie in their corresponding containers B_j.*

We first show that this lemma implies Lemma 10.4.3.

Proof of Lemma 10.4.3. Let $Z \sim \mathcal{N}(0, \sigma^2)$ and $s > 0$ be arbitrary. By $\beta \geq \sigma p/8 = \Omega(\sigma \log(n)/\log\log n) = \omega(\sigma\sqrt{\log n})$, we have $\beta \geq 3\sigma\sqrt{d(s+1)\ln n}$ for sufficiently large n. By definition of the containers, Lemma 10.1.1 yields that for any point X_j and sufficiently large n,

$$\Pr[X_j \notin B_j] \leq \Pr[\|Z\| \geq \beta] \leq \Pr[\|Z\| \geq \sigma 3\sqrt{d(s+1)\ln n}] \leq n^{-(s+1)}.$$

By union bound, we conclude that with probability $1 - n^{-s}$, all points are contained in their corresponding containers and hence, by the previous lemma, T is 2-optimal.

Recall that t is the largest odd integer satisfying $p^{2t+1} \leq (3\sigma)^{-1}$. Since $\sigma^{-1} = \Omega(\sqrt{n})$, this implies $t \geq \frac{p-1}{2} - 1$. Observe that T visits $t = \Omega(p)$ many layers and crosses a horizontal distance of $2/3$ in each of them. Hence, it has a length of at least $\Omega(p) = \Omega(\log n / \log\log n)$. □

In the remainder of this section, we prove Lemma 10.4.4, i.e., show that the constructed tour is 2-optimal, provided all points stay inside their respective containers. Clearly, it suffices to show for any pair of edges (u, v) and (w, z) in the tour, the corresponding *2-change*, i.e., replacing these edges by (u, w) and (v, z) does not reduce the tour length, i.e., $d(u, w) + d(v, z) \geq d(u, v) + d(w, z)$. We first state the technical lemmas capturing the ideas behind the construction.

The first lemma treats pairs of horizontal edges and establishes how large their vertical distance must be in order to make swapping these edges increase the length of the tour. It is a generalization of a similar lemma of Chandra et al. [CKT99] to a perturbation setting, in which points are placed arbitrarily into small containers.

Note that in what follows, for a point $p \in \mathbb{R}^2$, we let p_x denote its x-coordinate and p_y its y-coordinate. Furthermore, for any points $p, q \in \mathbb{R}^2$, we let $d_x(p, q) := |p_x - q_x|$ and $d_y(p, q) := |p_y - q_y|$ denote their horizontal and vertical distance, respectively.

Lemma 10.4.5. *Let $\overline{pq}$ and $\overline{rs}$ be horizontal line segments in the Euclidean plane with $p_x < q_x$ and $r_x < s_x$. Let B_p, B_q, B_r and B_s be circles of radius β with centers p, q, r and s, respectively. If $d(r, s) \geq d(p, q) + 4\beta$ and the vertical distance $v := d_y(p, r) = d_y(q, s)$ between $\overline{pq}$ and $\overline{rs}$ is at least*

$$\sqrt{d(p, q)d(r, s) + 4\beta d(r, s)} + 2\beta,$$

then, for all $\tilde{p} \in B_p, \tilde{q} \in B_q, \tilde{r} \in B_r, \tilde{s} \in B_s$, we have

$$d(\tilde{p}, \tilde{r}) + d(\tilde{q}, \tilde{s}) \geq d(\tilde{p}, \tilde{q}) + d(\tilde{r}, \tilde{s}).$$

Proof. Note that $d_y(\tilde{p}, \tilde{r}), d_y(\tilde{q}, \tilde{s}) \geq v - 2\beta$. Furthermore, we have that

$$\begin{aligned} d(r, s) - 2\beta \leq \tilde{s}_x - \tilde{r}_x &= (\tilde{p}_x - \tilde{r}_x) + (\tilde{q}_x - \tilde{p}_x) + (\tilde{s}_x - \tilde{q}_x) \\ &\leq (\tilde{p}_x - \tilde{r}_x) + d(p, q) + 2\beta + (\tilde{s}_x - \tilde{q}_x), \end{aligned}$$

and hence

$$(\tilde{p}_x - \tilde{r}_x) + (\tilde{s}_x - \tilde{q}_x) \geq d(r, s) - d(p, q) - 4\beta,$$

where the right-hand side expression is at least 0, since $d(r, s) \geq d(p, q) + 4\beta$ by assumption. Let $L := \tilde{p}_x - \tilde{r}_x$ and $R := \tilde{s}_x - \tilde{q}_x$, then it is straight-forward to verify that the expression

$$d(\tilde{p}, \tilde{r}) + d(\tilde{q}, \tilde{s}) \geq \sqrt{(v - 2\beta)^2 + L^2} + \sqrt{(v - 2\beta)^2 + R^2}, \tag{10.8}$$

subject to $L + R \geq d(r, s) - d(p, q) - 4\beta$ is minimized when $L = R = \frac{d(r,s) - d(p,q)}{2} - 2\beta$.

Hence, we can bound (10.8) by

$$\begin{aligned}
&d(\tilde{p},\tilde{r}) + d(\tilde{q},\tilde{s}) \\
&\geq\ 2\sqrt{(v-2\beta)^2 + \left(\frac{d(r,s)-d(p,q)}{2} - 2\beta\right)^2} \\
&\geq\ 2\sqrt{d(p,q)d(r,s) + 4\beta d(r,s) + \left(\frac{d(r,s)-d(p,q)}{2} - 2\beta\right)^2} \\
&=\ 2\sqrt{\left(\frac{d(r,s)-d(p,q)}{2}\right)^2 + d(p,q)d(r,s) + 4\beta d(r,s) - 2\beta(d(r,s)-d(p,q)) + (2\beta)^2} \\
&=\ 2\sqrt{\left(\frac{d(r,s)+d(p,q)}{2}\right)^2 + 2\beta(d(r,s)+d(p,q)) + (2\beta)^2} \\
&=\ 2\left(\frac{d(r,s)+d(p,q)}{2} + 2\beta\right) = d(r,s) + d(p,q) + 4\beta \geq d(\tilde{p},\tilde{q}) + d(\tilde{r},\tilde{s}),
\end{aligned}$$

where the third line follows from our assumption on v. □

The following very basic lemma shows that a sequence of edges that share roughly the same direction will always be 2-optimal.

Lemma 10.4.6. *Let p_1, p_2, p_3 and p_4 be a sequence of points in $[0,1]^2$ such that all connecting segments $p_{i+1} - p_i$ fulfill $|(p_{i+1}-p_i)_y| \leq (p_{i+1}-p_i)_x$. Then,*

$$d(p_1,p_3) + d(p_2,p_4) \geq d(p_1,p_2) + d(p_3,p_4).$$

Proof. For any point p, let C_p denote the cone $C_p := \{q \mid |(q-p)_y| \leq (q-p)_x\}$. Let $\Delta := p_2 - p_1$, then by assumption, we have $p_2 \in C_{p_1}$ and thus $|\Delta_y| \leq \Delta_x$. Let us assume that $0 \leq \Delta_y \leq \Delta_x$ (the other case is symmetric). Since by assumption, $p_3 \in C_{p_2}$, we have for $\Delta' := p_3 - p_1$ that $\Delta'_x = \Delta_x + \delta_x$ and $\Delta'_y = \Delta'_y + \delta_y$ for some $\delta_x > 0$ and δ_y with $|\delta_y| < \delta_x$. If $\delta_x \geq \Delta_y$, the claim is immediate from $d(p_1,p_3) \geq \Delta_x + \delta_x \geq \Delta_x + \Delta_y \geq d(p_1,p_2)$. Otherwise, for $\delta_x < \Delta_y$, we obtain

$$\begin{aligned}
d(p_1,p_3) &= \sqrt{(\Delta_x+\delta_x)^2 + (\Delta_y+\delta_y)^2} \\
&\geq \sqrt{(\Delta_x+\delta_x)^2 + (\Delta_y-\delta_x)^2} \\
&\geq \sqrt{\Delta_x^2 + \Delta_y^2 + 2\delta_x(\Delta_x - \Delta_y)} \geq \sqrt{\Delta_x^2 + \Delta_y^2} = d(p_1,p_2).
\end{aligned}$$

By an analogous computation, $d(p_2,p_4) \geq d(p_3,p_4)$ follows and hence the claim. □

We can now prove Lemma 10.4.4. Assume that all points are contained in their respective containers. We call an edge between X_i and X_j *horizontal* (or *vertical*) if the edge between x_i and x_j is horizontal (or vertical) and neither x_i nor x_j belong to the set of padding points. In what follows, we will first consider horizontal-horizontal, horizontal-vertical and vertical-vertical edge pairs and then turn to pairs of edges for which at least one edge is adjacent to some padding point. Recall that β is chosen such as to satisfy $a_t = 8\beta$.

Horizontal-horizontal edge pair. Let (X_i, X_{i+1}) and (X_j, X_{j+1}) be two horizontal edges. Horizontal edges (X_i, X_{i+1}) with $x_i, x_{i+1} \in L_k$ appear only if $k \leq t$. We distinguish the following cases.

1. $x_i, x_{i+1}, x_j, x_{j+1} \in L_k$: Both edges are in the same layer. Note that no 2-change swaps neighboring edges. Assume without loss of generality that $(x_i)_x < (x_{i+1})_x < (x_j)_x < (x_{j+1})_x$ (the other case is symmetric). Since $a_k \geq a_t = 8\beta$, we have that

$$d_y(X_i, X_{i+1}) \leq 2\beta \leq a_k - 2\beta \leq d_x(X_i, X_{i+1}).$$

   Similarly, $d_y(X_j, X_{j+1}) \leq d_x(X_j, X_{j+1})$ and $d_y(X_{i+1}, X_j) \leq d_x(X_{i+1}, X_j)$. This shows that Lemma 10.4.6 is applicable to $X_i, X_{i+1}, X_j, X_{j+1}$, which yields that no 2-change can be profitable.

2. $x_i, x_{i+1} \in L_k$, and $x_j, x_{j+1} \in L_{k+1}$. By construction of T, the edges have opposite direction. Assume that $(x_i)_x < (x_{i+1})_x$ and hence $(x_j)_x > (x_{j+1})_x$ (the other case is symmetric). By construction $(x_{i+1})_x - (x_i)_x \geq a_k$. We have that $(X_{i+1})_x - (X_i)_x \geq a_k - 2\beta \geq a_t - 2\beta = 6\beta > 0$. The same reasoning shows that $(X_j)_x > (X_{j+1})_x$. Similarly, one can show that $p_y > q_y$ for all $p \in \{X_j, X_{j+1}\}$ and $q \in \{X_i, X_{i+1}\}$. Hence the 2-change to (X_i, X_j) and (X_{i+1}, X_{j+1}) has a crossing, which by triangle inequality cannot be profitable.

3. $x_i, x_{i+1} \in L_k$, and $x_j, x_{j+1} \in L_{k+\ell}$ with $\ell \geq 2$ and $k + \ell \leq t$. Either both edges have opposite directions, then the previous argument shows that a 2-change is not profitable. Otherwise, note that the first requirement of Lemma 10.4.5, $a_k \geq a_{k+\ell} + 4\beta$, is fulfilled. Also note that $\beta = \frac{a_t}{8} \leq \frac{a_k}{8p^{2\ell}}$, since $k \leq t - \ell$. We have

$$\begin{aligned}
\sqrt{d(x_i, x_{i+1})d(x_j, x_{j+1}) + 4\beta d(x_i, x_{i+1})} + 2\beta &= \sqrt{a_k a_{k+\ell} + 4\beta a_k} + 2\beta \\
&\leq \sqrt{\frac{a_k^2}{p^{2\ell}} + \frac{a_k^2}{2p^{2\ell}}} + \frac{a_k}{4p^{2\ell}} \\
&\leq \sqrt{\frac{3}{2}} \cdot \frac{a_k}{p^\ell} + \frac{a_k}{4p^{2\ell}} \\
&\leq \left(\sqrt{\frac{3}{2}} \cdot \frac{1}{p^{\ell-1}} + \frac{1}{4p^{2\ell-1}}\right) \frac{a_k}{p} \\
&\leq c_k \leq \sum_{m=0}^{\ell-1} c_{k+m} = d_y(x_i, x_j),
\end{aligned}$$

   since for sufficiently large p, we have $\sqrt{3/2}/p^{\ell-1} + 1/(4p^{2\ell-1}) \leq 1$. Consequently, Lemma 10.4.5 applies and shows that the 2-change does not yield an improvement.

Horizontal-vertical edge pair. Let (X_i, X_{i+1}) be a vertical edge and (X_j, X_{j+1}) be a horizontal edge. We assume that the vertical edge is in V_1, since the case $x_i, x_{i+1} \in V_2$ is symmetric. Exactly one of the following cases occurs.

1. $x_i \in L_k, x_{i+1} \in L_{k+1}$ and $x_j, x_{j+1} \in L_{k'}$ with $k' \in \{k, k+1\}$. The horizontal edge is in the same layer as one of the end points of the vertical edge. Clearly, $d(X_i, X_{i+1}) \leq c_k + 2\beta$ and $d(X_j, X_{j+1}) \leq a_{k'} + 2\beta$. Since a 2-change cannot swap neighboring edges, at least one horizontal segment lies between both edges. By

construction of the tour, one of the edges $\{x_i, x_j\}$ and $\{x_{i+1}, x_{j+1}\}$ crosses a vertical distance of at least c_k and the other a horizontal distance of at least $2a_{k'}$. Hence

$$d(X_i, X_j) + d(X_{i+1}, X_{j+1}) \geq 2a_{k'} + c_k - 4\beta \geq a_{k'} + c_k + 4\beta,$$

since $a_{k'} \geq a_t = 8\beta$.

2. $x_i \in L_k, x_{i+1} \in L_{k+1}$ and $x_j, x_{j+1} \in L_{k'}$ with $k' \notin \{k, k+1\}$. As in the previous case, $d(X_i, X_{i+1}) \leq c_k + 2\beta$ and $d(X_j, X_{j+1}) \leq a_{k'} + 2\beta$. Consider first the case that $k' < k$, then by construction of the tour, one of the edges $\{x_i, x_j\}$ and $\{x_{i+1}, x_{j+1}\}$ crosses a horizontal distance of at least $a_{k'}$ and the other edge crosses a vertical distance of at least $c_{k'}$, yielding

$$d(X_i, X_j) + d(X_{i+1}, X_{j+1}) \geq a_{k'} + c_{k'} - 4\beta \geq a_{k'} + c_k + 4\beta,$$

since $c_{k'} \geq c_{k-1} \geq c_k + 8\beta$. Otherwise, if $k' > k + 1$, the edges x_i, x_j crosses a vertical distance of at least $c_{k+1} + c_k$ and hence

$$d(X_i, X_j) + d(X_{i+1}, X_{j+1}) \geq c_{k+1} + c_k - 2\beta \geq a_{k'} + c_k + 4\beta,$$

since $c_{k+1} \geq a_{k+2} + 6\beta \geq a_{k'} + 6\beta$. Thus in both cases, a 2-change is not profitable.

Vertical-vertical edge pair. Let (X_i, X_{i+1}) and (X_j, X_{j+1}) be vertical edges.

1. $x_i \in L_k, x_{i+1} \in L_{k+1}$ and $x_j \in L_{k'}, x_{j+1} \in L_{k'+1}$ with $(x_i)_x = (x_j)_x$, i.e., the vertical edges are above each other. By swapping the x- and y-axis in Lemma 10.4.6, we can show that a 2-change is not profitable, since it is easy to see that $|(p-q)_x| \leq (p-q)_y$ for all consecutive pairs (p, q) in $(X_i, X_{i+1}, X_j, X_{j+1})$.

2. $x_i \in L_k, x_{i+1} \in L_{k+1}$ and $x_j \in L_{k'}, x_{j+1} \in L_{k'+1}$ with $(x_i)_x \neq (x_{i'})_x$. Clearly, $d(X_i, X_j) \geq a_0 - 2\beta$ and $d(X_{i+1}, X_{j+1}) \geq a_0 - 2\beta$, while $d(X_i, X_{i+1}) \leq c_k + 2\beta \leq c_0 + 2\beta$ and $d(X_j, X_{j+1}) \leq c_{k'} + 2\beta \leq c_0 + 2\beta$. Hence a 2-change is not profitable, since $a_0 \geq 8\beta + c_0$.

Padding points. Since we assumed for convenience that the padding points are placed at the central vertex C of Layer t in V_3, only the edges with at least one endpoint in V_3 are relevant candidates for the treatment of padding points. This is because all other edges have both endpoints at a distance of $1/6$ to the padding points, which can never be accounted for by its edge length, since all edges except in Layer 0 are much shorter than $1/3$. Separately, the Layer-0 edges can be handled easily as well: an edge $\{X_i, X_{i'}\}$ with $x_i = x_{i'} \in \overline{X}^{\text{pad}}$ is a horizontal edge, hence the pair $(X_i, X_{i'})$ and a Layer-0 edge trigger the corresponding case of horizontal-horizontal edge pairs with even smaller edge length of the edge $(X_i, X_{i'})$ in Layer t.

It remains to handle the following cases, where we regard C as a padding point, i.e., $C \in \overline{X}^{\text{pad}}$, not as a Layer-$t$ point.

1. $x_i, x_{i'} \in \overline{X}^{\text{pad}}$, and $x_j, x_{j'} \in L_t$. Clearly, $d(X_j, X_{j'}) \leq a_t + 2\beta$ and $d(X_j, X_{j'}) \leq 2\beta$. Furthermore, at least one of $\{x_j, x_{j'}\}$ has a horizontal distance of at least $2a_t$ to $x_i = x_{i'}$. Hence,

$$d(X_i, X_j) + d(X_{i'}, X_{j'}) \geq 2a_t - 2\beta \geq a_t + 4\beta \geq d(X_j, X_{j'}) + d(X_i, X_{i'})$$

2. $x_i \in \{C_\ell, C_r\}, x_{i'} \in \overline{X}^{\text{pad}}$ and $x_j, x_{j'} \in L_t$. These edge pairs are exactly as regular pairs of Layer-t edges and the corresponding case of horizontal-horizontal edge pairs applies.

3. $x_i, x_{i'}, x_j, x_{j'} \in \overline{X}^{\text{pad}} \cup \{C_r, C_\ell\}$. All such edges are 2-optimal by construction, since a *2-optimal path* from C_ℓ to C_r passing by all padding points was used.

This concludes the case analysis and thus the proof of Lemma 10.4.4.

10.5 Discussions and Open Problems

We have proved an upper bound of $\mathcal{O}(\log 1/\sigma)$ for the smoothed approximation ratio of 2-Opt. Furthermore, we have proved that the lower bound of Chandra et al. [CKT99] remains robust even for $\sigma = \mathcal{O}(1/\sqrt{n})$. We leave as an open problem to generalize our upper bounds to the one-step model to improve the current bound of $\mathcal{O}(\sqrt[d]{\phi})$ [ERV14], but conjecture that this might be difficult.

While our bound significantly improves the previously known bound for the smoothed approximation ratio of 2-Opt, we readily admit that it still does not explain the performance observed in practice. A possible explanation is that when the initial tour is not picked by an adversary or the nearest neighbor heuristic, but using a construction heuristic such as the spanning tree heuristic or an insertion heuristic, an approximation factor of 2 is guaranteed even before 2-Opt has begun to improve the tour [RSL77]. However, a smoothed analysis of the approximation ratio of 2-Opt initialized with a good heuristic might be difficult: even in the average case, it is only known that the length of an optimal TSP is concentrated around $\gamma_d \cdot n^{\frac{d-1}{d}}$ for some constant $\gamma_d > 0$. But the precise value of γ_d is unknown [Yuk98]. Since experiments suggest that 2-Opt even with good initialization does not achieve an approximation ratio of $1 + o(1)$ [JM97; JM02], one has to deal with the precise constants, which seems challenging.

Finally, we conjecture that many examples for showing lower bounds for the approximation ratio of concrete algorithms for Euclidean optimization such as the TSP remain stable under perturbation for $\sigma = \mathcal{O}(1/\sqrt{n})$. The question remains whether such small values of σ, although they often suffice to prove polynomial smoothed running time, are essential to explain practical approximation ratios or if already slower decreasing σ provide a sufficient explanation.

Chapter 11

Approximating the Fréchet Distance on c-Packed Curves

This chapter is devoted to presenting an improved approximation algorithm for the Fréchet distance on realistic input curves. Specifically, we prove the following main result.

Theorem 9.2.1. *For any $0 < \varepsilon \leq 1$, we can compute a $(1+\varepsilon)$-approximation of the continuous and discrete Fréchet distance on c-packed curves in time $\tilde{\mathcal{O}}(cn/\sqrt{\varepsilon})$.*

In any dimension $d \geq 5$, this matches the conditional lower bound that for varying $\varepsilon > 0$, there is no $(1+\varepsilon)$-approximation in time $\mathcal{O}((cn/\sqrt{\varepsilon})^{1-\delta})$ for any $\delta > 0$, unless SETH fails [Bri14].

Our algorithm also yields improved running time guarantees for other models of realistic input curves, like κ-bounded and κ-straight curves, where we are also able to essentially replace ε by $\sqrt{\varepsilon}$ in the running time bound. However, there are no matching lower bounds known in these cases. We provide the details in Section 11.2.2.

Connection to lower bound. We obtained our new algorithm by investigating why the conditional lower bound [Bri14] cannot be improved and exploiting the discovered properties. Thus, a conditional lower bound made it possible to come up with Theorem 9.2.1. This is similar to the results in Chapter 6, which we have obtained after noting that our conditional lower bounds could not be strengthened beyond a certain bound. Here, however, the reason why the conditional lower bound could not be strengthened in some sense even suggested properties that make these cases tractable. In what follows, we briefly describe this vague connection.

In the conditional lower bound [Bri14], two curves π and σ are constructed for which it is hard to decide whether $d_{\mathrm{F}}(\pi, \sigma)$ is at most δ or at least $(1+\varepsilon)\delta$. In this construction, for some consecutive points π_i, π_{i+1} and σ_j, σ_{j+1} we want to force any algorithm reaching (π_i, σ_j) to make a simultaneous step to $(\pi_{i+1}, \sigma_{j+1})$. That is, we want that (i) $\|\pi_i - \sigma_j\|, \|\pi_{i+1} - \sigma_{j+1}\| \leq \delta$ and (ii) $\|\pi_i - \sigma_{j+1}\|, \|\pi_{i+1} - \sigma_j\| > (1+\varepsilon)\delta$. By elementary geometric arguments, (i) and (ii) imply $\|\pi_i - \pi_{i+1}\|, \|\sigma_j - \sigma_{j+1}\| \geq \sqrt{\varepsilon}\delta$. Thus, we cannot "compress" the curves π and σ too well (in terms of c-packedness), resulting in the factor $1/\sqrt{\varepsilon}$ in the lower bound of [Bri14].

This bottleneck vaguely connects to the following useful property: Consider curves π and σ with the property that every point on the curve has distance at most $\sqrt{\varepsilon}\delta$ to the starting point of the curve. Consider the line L containing the starting points of π and σ. We project π and σ onto L, thereby obtaining one-dimensional curves $\hat{\pi}, \hat{\sigma}$ (see Figure 11.2 in Section 11.2.3). By the Pythagorean theorem, this projection keeps pairwise distances between π and σ roughly fixed: If $\|\pi_i - \sigma_j\|$ is approximately δ, then $\|\hat{\pi}_i - \hat{\sigma}_j\| = \|\pi_i - \sigma_j\| + \mathcal{O}(\varepsilon\delta)$. Thus, for $(1+\varepsilon)$-approximation algorithms this projection is admissible. We next describe how we use this property in our algorithm.

Outline of the Algorithm. We give an improved algorithm that approximately decides whether the Fréchet distance of two given curves π and σ is at most δ. Using a construction of [DHPW12] to search over possible values of δ, this yields an improved approximation algorithm. We partition our curves into subcurves, each of which is either a *long segment*, i.e., a single segment of length at least $\Lambda = \Theta(\sqrt{\varepsilon}\delta)$, or a *piece*, i.e., a subcurve staying in the ball of radius Λ around its initial vertex. We then run the usual algorithm that explores the reachable free-space (see Section 11.1 for definitions), however, we treat regions spanned by a piece π' of π and a piece σ' of σ in a special way. Typically, if π' and σ' consist of n' and m' segments, respectively, then their free-space could be resolved in time $\mathcal{O}(n'm')$. Our overall speedup comes from reducing this runtime to $\tilde{\mathcal{O}}(n'+m')$, which is our first main contribution. As discussed above, we project π', σ' onto the line through their starting points, obtaining one-dimensional curves $\hat{\pi}$ and $\hat{\sigma}$ without changing pairwise distances significantly. Moreover, we show how to ensure that $\hat{\pi}$ and $\hat{\sigma}$ are *separated*, i.e., all vertices of $\hat{\pi}$ lie on the one side of some point z on the line and all vertices of $\hat{\sigma}$ lie on the other side. Hence, we reduced our problem to resolving the free-space region of one-dimensional separated curves.

It is easy to see that the Fréchet distance of one-dimensional separated curves can be computed in near-linear time, since we can walk along π and σ with (appropriately defined) *greedy steps* to either find a feasible traversal or bottleneck subcurves. However, we face the additional difficulty that we have to resolve the *free-space region* of one-dimensional separated curves, i.e., given entry points on $\hat{\pi}$ and $\hat{\sigma}$, compute all exits on $\hat{\pi}$ and $\hat{\sigma}$. Our second main contribution is to present an extension of the easy greedy algorithm to handle this more complex problem.

Interestingly, Bringmann and Mulzer [BM16] very recently showed that the assumption of *separated* curves is indeed necessary: Already approximating the discrete Fréchet distance on general one-dimensional curves in (strongly) subquadratic time is not possible unless SETH fails.

Organization. We start with basic definitions and techniques borrowed from Driemel et al. [DHPW12] in Section 11.1. In Section 11.2, we present our approximate decision procedure which reduces the problem to one-dimensional separated curves. We solve the latter in Section 11.3. Here, we focus on the continuous Fréchet distance. It is straightforward to obtain a similar algorithm for the discrete variant: in this case, Section 11.3.1 becomes obsolete, which is why we save a factor of $\log 1/\varepsilon$ in the running time.

11.1 Preliminaries

Throughout this chapter, we fix the dimension $d \geq 2$. For $z \in \mathbb{R}^d$, $r > 0$ we let $B(z, r)$ be the ball of radius r around z. For integers $i \leq j$, we let $[i..j] := \{i, i+1, \ldots, j\}$, which is not to be confused with the real interval $[i, j] = \{x \in \mathbb{R} \mid i \leq x \leq j\}$. A (polygonal) curve π is defined by its vertices $(\pi_1, \ldots, \pi_n)$ with $\pi_p \in \mathbb{R}^d$, $p \in [1..n]$. We let $|\pi| = n$ be the number of vertices of π and $\|\pi\|$ be its total length $\sum_{i=1}^{n-1} \|\pi_i - \pi_{i+1}\|$, where $\|z\|$ denotes the Euclidean norm of $z \in \mathbb{R}^d$. We write $\pi_{p..b}$ for the subcurve $(\pi_p, \pi_{p+1}, \ldots, \pi_b)$. Similarly, for an interval $I = [p..b]$ we write $\pi_I = \pi_{p..b}$. We can also view π as a continuous function $\pi\colon [1, n] \to \mathbb{R}^d$ with $\pi_{p+\lambda} = (1-\lambda)\pi_p + \lambda\pi_{p+1}$ for $p \in [1..n-1]$ and $\lambda \in [0, 1]$. For the second curve $\sigma = (\sigma_1, \ldots, \sigma_m)$ we will use indices of the form $\sigma_{q..d}$ for the reader's convenience.

Variants of the Fréchet distance. Let Φ_n be the set of all continuous and non-decreasing functions ϕ from $[0,1]$ onto $[1,n]$. The *continuous Fréchet distance* between two curves π and σ with n and m vertices, respectively, is defined as

$$d_{\mathrm{F}}(\pi,\sigma) := \inf_{\substack{\phi_1\in\Phi_n\\ \phi_2\in\Phi_m}} \max_{t\in[0,1]} \|\pi_{\phi_1(t)} - \sigma_{\phi_2(t)}\|.$$

We call $\phi := (\phi_1,\phi_2)$ a (continuous) *traversal* of (π,σ), and say that it has *width* $\max_{t\in[0,1]} \|\pi_{\phi_1(t)} - \sigma_{\phi_2(t)}\|$.

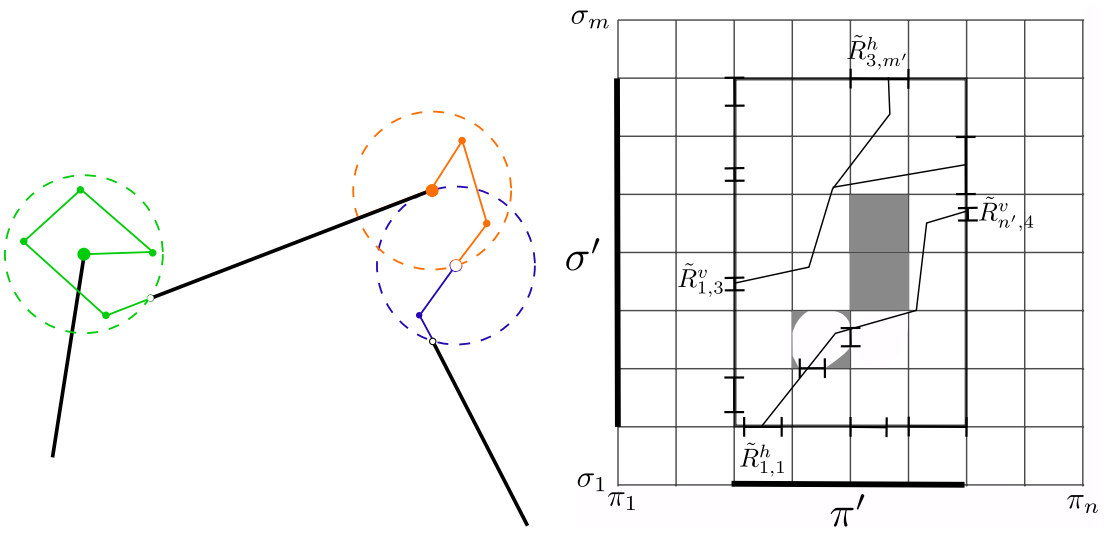
In the discrete case, we let Δ_n be the set of all non-decreasing functions ϕ from $[0,1]$ onto $[1..n]$. We obtain the *discrete Fréchet distance* $d_{\mathrm{dF}}(\pi,\sigma)$ by replacing Φ_n and Φ_m by Δ_n and Δ_m. This also yields analogous notions of a (discrete) *traversal* and its *width*. Note that any $\phi\in\Delta_n$ is a staircase function attaining all values in $[1..n]$. Hence, $(\phi_1(t),\phi_2(t))$ changes only at finitely many points in time t. At any such *time step*, we jump to the next vertex in π or σ or both.

Free-space diagram. The discrete *free-space* of curves π,σ is defined as $\mathcal{D}^d_{\le\delta}(\pi,\sigma) := \{(p,q)\in[1..n]\times[1..m] \mid \|\pi_p-\sigma_q\|\le\delta\}$. Note that any discrete traversal of π,σ of width at most δ corresponds to a monotone sequence of points in the free-space where at each point in time we increase p or q or both. Because of this property, the free-space is a standard concept used in many algorithms for the Fréchet distance.

The continuous free-space is defined as $\mathcal{D}_{\le\delta}(\pi,\sigma) := \{(p,q)\in[1,n]\times[1,m] \mid \|\pi_p - \sigma_q\|\le\delta\}$. Again, a monotone path from $(1,1)$ to (n,m) in $\mathcal{D}_{\le\delta}(\pi,\sigma)$ corresponds to a traversal of width at most δ. It is well known [AG95; God91] that each *free-space cell* $C_{i,j} := \{(p,q)\in[i,i+1]\times[j,j+1] \mid \|\pi_p-\sigma_q\|\le\delta\}$ (for $i\in[1..n-1], j\in[1..m-1]$) is convex, specifically it is the intersection of an ellipse with $[i,i+1]\times[j,j+1]$. In particular, the intersection of the free-space with any interval $[i,i+1]\times\{j\}$ (or $\{i\}\times[j,j+1]$) is an interval $I^h_{i,j}$ (or $I^v_{i,j}$), and for any such interval the subset that is reachable by a monotone path from $(1,1)$ is an interval $R^h_{i,j}$ (or $R^v_{i,j}$). Moreover, in constant time one can solve the following *free-space cell problem*: Given intervals $R^h_{i,j}\subseteq[i,i+1]\times\{j\}, R^v_{i,j}\subseteq\{i\}\times[j,j+1]$, determine the intervals $R^h_{i,j+1}\subseteq[i,i+1]\times\{j+1\}, R^v_{i+1,j}\subseteq\{i+1\}\times[j,j+1]$ consisting of all points that are reachable from a point in $R^h_{i,j}\cup R^v_{i,j}$ by a monotone path within the free-space cell $C_{i,j}$. Solving this problem for all cells from lower left to upper right we determine whether (n,m) is reachable from $(1,1)$ by a monotone path and thus decide whether the Fréchet distance is at most δ.

From approximate deciders to approximation algorithms. An *approximate decider* is an algorithm that, given curves π,σ and $\delta>0, 0<\varepsilon\le 1$, returns one of the outputs (1) $d_{\mathrm{F}}(\pi,\sigma)>\delta$ or (2) $d_{\mathrm{F}}(\pi,\sigma)\le(1+\varepsilon)\delta$. In any case, the returned answer has to be correct. In particular, if $\delta<d_{\mathrm{F}}(\pi,\sigma)\le(1+\varepsilon)\delta$ the algorithm may return either of the two outputs.

Let $D(\pi,\sigma,\delta,\varepsilon)$ be the running time of an approximate decider and set $D(\pi,\sigma,\varepsilon) := \max_{\delta>0} D(\pi,\sigma,\delta,\varepsilon)$. We assume polynomial dependence on ε, i.e., that there are constants $0<c_1<c_2<1$ such that for any $0<\varepsilon\le 1$ we have $c_1 D(\pi,\sigma,\varepsilon/2)\le D(\pi,\sigma,\varepsilon)\le c_2 D(\pi,\sigma,\varepsilon/2)$. Driemel et al. [DHPW12] gave the following construction of a $(1+\varepsilon)$-approximation for the Fréchet distance given an approximate decider. (This follows from [DHPW12, Theorem 3.15] after replacing their concrete approximate decider with running time "$\mathcal{O}(N(\varepsilon,\pi,\sigma))$" by any approximate decider with running time $D(\pi,\sigma,\varepsilon)$.)



(A) This figure illustrates our partitioning of a curve into *pieces* (contained in dashed circles) and *long segments* (bold edges).

(B) The free-space problem for pieces π' and σ' in the free-space diagram of π and σ. Given entry intervals on the lower and left boundary of the region, compute exit intervals on the upper and right boundary.

FIGURE 11.1: Definition and treatment of pieces.

Lemma 11.1.1. *Given an approximate decider with running time $D(\pi, \sigma, \varepsilon)$ we can construct a $(1+\varepsilon)$-approximation for the Fréchet distance with running time $\mathcal{O}\big(D(\pi, \sigma, \varepsilon) + D(\pi, \sigma, 1) \log n\big)$.*

11.2 The Approximate Decider

By Lemma 11.1.1 it suffices to give an improved approximate decider with running time $\mathcal{O}(\frac{cn}{\sqrt{\varepsilon}} \log^2(1/\varepsilon))$ for the Fréchet distance to prove Theorem 9.2.1.

Long segments and pieces. Let π and σ be curves for which we are to (approximately) decide whether $d_F(\pi, \sigma) > \delta$ or $d_F(\pi, \sigma) \leq (1 + \varepsilon)\delta$. We first partition π and σ into subcurves, each of which is either a *long segment*, i.e., a single segment of length at least $\Lambda = \Theta(\sqrt{\varepsilon}\delta)$, or a *piece*, i.e., a subcurve staying in the ball of radius Λ around its initial vertex (see Figure 11.1a). More formally, we modify the curve π by introducing new vertices as follows. Start with the initial vertex π_1 as current vertex. If the segment following the current vertex has length at least $\Lambda = \Lambda_{\varepsilon,\delta} := \min\{\frac{1}{2}\sqrt{\varepsilon}, \frac{1}{4}\} \cdot \delta$ then mark this segment as *long* and set the next vertex as the current vertex. Otherwise follow π from the current vertex π_x to the first point π_y such that $\|\pi_x - \pi_y\| = \Lambda$ (or until we reach the last vertex of π). If π_y is not a vertex, but lies on some segment of π, then introduce a new vertex at π_y. Mark $\pi_{x..y}$ as a *piece* of π and set π_y as current vertex. Repeat until π is completely traversed. Since this procedure introduces at most $|\pi|$ new vertices and does not change the shape of π, with slight abuse of notation we call the resulting curve again π and set $n := |\pi|$. This partitions π into subcurves $\pi^1, \ldots, \pi^k$, with $\pi^s = \pi_{p_s..b_s}$, where every part π^s is either (see also Figure 11.1a)

- a *long segment*: $b_s = p_s + 1$ and $\|\pi_{p_s} - \pi_{b_s}\| \geq \Lambda$, or
- a *piece*: $\|\pi_{p_s} - \pi_{b_s}\| = \Lambda$ and $\|\pi_{p_s} - \pi_x\| < \Lambda$ for all $x \in [p_s, b_s)$.

Note that the last piece actually might have distance $\|\pi_{p_s} - \pi_{b_s}\|$ less than Λ, however, for simplicity we assume equality for all pieces (in fact, a special handling of the last piece would only be necessary in Lemmas 11.2.6 and 11.2.9). Similarly, we introduce new vertices on σ and partition it into subcurves $\sigma^1, \ldots, \sigma^\ell$, with $\sigma^t = \sigma_{q_t..d_t}$, each of which is a long segment or a piece. Let $m := |\sigma|$.

Free-space regions. We follow the usual approach of exploring the reachable free-space (see Section 11.1). However, we treat regions spanned by a piece π' of π and a piece σ' of σ in a special way. Typically, if π', σ' consist of n', m' segments then their free-space would be resolved in time $\mathcal{O}(n'm')$, by resolving each of the $n'm'$ induced free-space cells in constant time. We show that by resorting to approximation we can reduce this running time to $\tilde{\mathcal{O}}(n' + m')$. This is made formal by the following subproblem and lemma (see also Figure 11.1b).

Problem 11.2.1 (Free-space region problem). Given $\delta > 0$, $0 < \varepsilon \leq 1$, curves π and σ with n and m vertices, respectively, and *entry intervals* $\tilde{R}^h_{i,1} \subseteq [i, i+1] \times \{1\}$ for $i \in [1..n)$ and $\tilde{R}^v_{1,j} \subseteq \{1\} \times [j, j+1]$ for $j \in [1..m)$, compute *exit intervals* $\tilde{R}^h_{i,m} \subseteq [i, i+1] \times \{m\}$ for $i \in [1..n)$ and $\tilde{R}^v_{n,j} \subseteq \{n\} \times [j, j+1]$ for $j \in [1..m)$ such that (1) the exit intervals contain all points reachable from the entry intervals by a monotone path in $\mathcal{D}_{\leq\delta}(\pi, \sigma)$ and (2) all points in the exit intervals are reachable from the entry intervals by a monotone path in $\mathcal{D}_{\leq(1+\varepsilon)\delta}(\pi, \sigma)$.

To stress that we work with approximations, we denote reachable intervals by $\tilde{R}$ instead of R in the remainder of the chapter.

Lemma 11.2.2. *If π and σ are pieces then the free-space region problem can be solved in time $\mathcal{O}((n + m) \log^2 1/\varepsilon)$.*

We will prove this lemma in Sections 11.2.3 and 11.3. Using the algorithm of the above lemma, we obtain an approximate decider for the Fréchet distance as follows.

Algorithm 11.2.3. We consider all regions $r_{s,t} = [p_s, b_s] \times [q_t, d_t]$ spanned by parts π^s and σ^t. With each region $r_{s,t}$ we are going to store the entry intervals $\tilde{R}^h_{i,q_t} \subseteq [i, i+1] \times \{q_t\}$ for $i \in [p_s..b_s)$ and $\tilde{R}^v_{p_s,j} \subseteq \{p_s\} \times [j, j+1]$ for $j \in [q_t..d_t)$. We correctly initialize the outer reachability intervals $\tilde{R}^h_{i,1}$ and $\tilde{R}^v_{1,j}$. Then we enumerate all regions sorted by increasing layer $s + t$, and among all regions with equal $s + t$ sorted by s. For each region $r_{s,t}$ we resolve its free-space region: (1) If both π^s, σ^t are long segments, we can resolve the free-space cell $r_{s,t}$ in constant time, (2) if π^s is a piece and σ^t is a long segment, we sequentially resolve the free-space cells $[i, i+1] \times [q_t, d_t]$ for $i = p_s, \ldots, b_s - 1$ (and symmetrically if π^s is a long segment and σ^t a piece), and (3) if both π^s, σ^t are pieces we solve the corresponding free-space region problem using Lemma 11.2.2. Finally, we return $d_{\mathrm{F}}(\pi, \sigma) \leq (1 + \varepsilon)\delta$ if $(n, m) \in \tilde{R}^h_{n-1,m}$ and $d_{\mathrm{F}}(\pi, \sigma) > \delta$ otherwise.

Observe that instead of enumerating *all* regions, we may enumerate only *reachable* regions, i.e., regions where some stored entry interval is non-empty. Indeed, if the reachable regions in layer L are $r_{s_1,L-s_1}, \ldots, r_{s_a,L-s_a}$, sorted by $s_1 \leq \ldots \leq s_a$, then the reachable regions in layer $L+1$ are among $\{r_{s_i+1,L-s_i}, r_{s_i,L-s_i+1} \mid 1 \leq i \leq a\}$, and we can check for each such region in constant time whether it is reachable, so we can efficiently enumerate all reachable regions in layer $L + 1$, again sorted by s. This trick was also used in [DHPW12, Lemma 3.1].

Lemma 11.2.4. *Algorithm 11.2.3 is a correct approximate decider.*

Proof. Observe that if $(n,m) \in \tilde{R}^h_{n-1,m}$ then there exists a monotone path from $(1,1)$ to (n,m) in $\mathcal{D}_{\leq(1+\varepsilon)\delta}(\pi,\sigma)$, which implies $d_{\mathrm{F}}(\pi,\sigma) \leq (1+\varepsilon)\delta$. If $d_{\mathrm{F}}(\pi,\sigma) \leq \delta$ then there is a monotone path from $(1,1)$ to (n,m) in $\mathcal{D}_{\leq\delta}(\pi,\sigma)$, implying $(n,m) \in \tilde{R}^h_{n-1,m}$. □

Let us analyze the time complexity of Algorithm 11.2.3. Let M be the set of all non-empty regions $r_{s,t}$. We define the free-space complexity

$$N(\pi,\sigma,\delta,\varepsilon) := \sum_{(s,t)\in M} (|\pi^s| + |\sigma^t|),$$

and set $N(\pi,\sigma,\varepsilon) := \max_{\delta>0} N(\pi,\sigma,\delta,\varepsilon)$. Since the algorithm considers only *reachable* regions and any reachable region is also non-empty, the running time of Algorithm 11.2.3 is $\mathcal{O}(N(\pi,\sigma,\varepsilon)\log^2 1/\varepsilon)$. Indeed, (1) if π^s, σ^t are both long segments then the running time for resolving their free-space cell is constant and they contribute a constant to $N(\pi,\sigma,\varepsilon)$, (2) if one of π^s, σ^t is a long segment and the other a piece then the running time is $\mathcal{O}(|\pi^s| + |\sigma^t|)$, which is their contribution to $N(\pi,\sigma,\varepsilon)$, and (3) if both π^s, σ^t are pieces, then solving the free-space region problem takes time $\mathcal{O}((|\pi^s| + |\sigma^t|)\log^2(1/\varepsilon))$ by Lemma 11.2.2. This proves the following lemma.

Lemma 11.2.5. *The approximate decider in Algorithm 11.2.3 runs in time* $D(\pi,\sigma,\varepsilon) = \mathcal{O}(N(\pi,\sigma,\varepsilon)\cdot\log^2 1/\varepsilon)$.

To prove Theorem 9.2.1, it remains to give a bound on $N(\pi,\sigma,\varepsilon)$ for c-packed curves, which is done in the following section. In Section 11.2.2, we give corresponding bounds for κ-bounded and κ-straight curves, yielding an improved algorithm for such curves as well.

11.2.1 Free-Space Complexity of c-Packed Curves

Recall that a curve π is c-packed if for any point $z \in \mathbb{R}^d$ and any radius $r > 0$ the total length of π inside the ball $B(z,r)$ is at most cr.

Lemma 11.2.6. *Let* π, σ *be* c*-packed curves with* n *vertices in total and* $\varepsilon > 0$. *Then* $N(\pi,\sigma,\varepsilon) = \mathcal{O}(cn/\sqrt{\varepsilon})$.

Proof. Our proof uses a similar argument as [DHPW12, Lemma 4.4]. Let $\delta > 0$ be arbitrary. We charge $N(\pi,\sigma,\varepsilon)$ to the segments of π and σ as follows. For any non-empty region $r_{s,t}$, we analyze: (1) if both π^s, σ^t are long segments, then the longer segment charges 1 to the shorter one, (2) if π^s is a piece and σ^t a long segment, then σ^t charges 1 to each of the $|\pi^s|$ segments of π^s (we proceed symmetrically if π^s is a long segment and σ^t a piece), and (3) if both π^s, σ^t are pieces then we charge 1 to each of the $|\pi^s| + |\sigma^t|$ segments of π^s and σ^t (each segment of π^s is charged by the piece σ^t, and the other way round). This accounts for $N(\pi,\sigma,\varepsilon)$, up to constant factors.

Let e be any segment of π. By construction, every part σ^t that charges 1 to e has length $\|\sigma^t\| \geq \|e\|$ (irrespective of whether σ^t is a long segment or piece): (1) If both e, σ^t are long segments then σ^t is only charging e if its length is longer, and (2) if e is contained in a piece π^s then $\|e\| \leq \Lambda \leq \|\sigma^t\|$, irrespective of σ^t being a long segment or piece.

Moreover, if σ^t is charging e, then σ^t and e have distance at most $(1+\varepsilon)\delta + 2\Lambda$. Indeed, if the ($\varepsilon$-approximate) free-space cell of segments $\pi_{i..i+1}, \sigma_{j..j+1}$ is non-empty then the segments have distance at most $(1+\varepsilon)\delta$. For pieces π^s, σ^t, for a charge to happen it suffices that *some* segments of π^s, σ^t lie in distance $(1+\varepsilon)\delta$, but then *any* pair of segments is in distance $(1+\varepsilon)\delta + 2\Lambda$.

It follows that σ^t contributes at least $\mu := \max\{\|e\|, \Lambda\}$ to the length of σ in the ball B of radius $\frac{1}{2}\|e\| + \max\{\|e\|, \Lambda\} + ((1+\varepsilon)\delta + 2\Lambda)$ around the midpoint of e. Since σ is c-packed, the number of charges to e is at most

$$\frac{\|\sigma \cap B\|}{\mu} \leq \frac{cr}{\mu} \leq \frac{c(\frac{3}{2}\|e\| + (1+\varepsilon)\delta + 3\Lambda)}{\max\{\|e\|, \Lambda\}} \leq \tfrac{9}{2}c + \frac{c(1+\varepsilon)\delta}{\min\{\frac{1}{2}\sqrt{\varepsilon}, \frac{1}{4}\} \cdot \delta} = \mathcal{O}\Big(\frac{c}{\sqrt{\varepsilon}}\Big).$$

Summing up over all segments e of π, the free-space complexity $N(\pi, \sigma, \varepsilon)$ is bounded by $\mathcal{O}(cn/\sqrt{\varepsilon})$. □

Combining Lemmas 11.2.6, 11.2.5, and 11.1.1, we obtain an approximation algorithm for the Fréchet distance with running time $\mathcal{O}(\frac{cn}{\sqrt{\varepsilon}} \log^2 1/\varepsilon + cn \log n) = \tilde{\mathcal{O}}(\frac{cn}{\sqrt{\varepsilon}})$, as desired.

11.2.2 Free-Space Complexity of κ-Bounded and κ-Straight Curves

Definition 11.2.7. *Let $\kappa \geq 1$ be a given parameter. A curve π is* κ-straight *if for any $p, b \in [1, |\pi|]$ we have $\|\pi_{p..b}\| \leq \kappa\|\pi_p - \pi_b\|$. A curve π is* κ-bounded *if for all p, b the subcurve $\pi_{p..b}$ is contained in $B(\pi_p, r) \cup B(\pi_b, r)$, where $r = \frac{\kappa}{2}\|\pi_p - \pi_b\|$.*

The following lemma from [DHPW12] allows us to transfer our speedup for c-packed curves directly to κ-straight curves. Thus, we improve the best previous algorithm for κ-straight curves with running time $\mathcal{O}(\kappa n/\varepsilon + \kappa n \log n)$ [DHPW12] to time $\tilde{\mathcal{O}}(\kappa n/\sqrt{\varepsilon})$.

Lemma 11.2.8. *A κ-straight curve is 2κ-packed.*

In the remainder of this section we consider κ-bounded curves, closely following [DHPW12, Sect. 4.2].

Lemma 11.2.9. *Let $\delta > 0$, $0 < \varepsilon \leq 1$, $\lambda > 0$, and let π be a κ-bounded curve with disjoint subcurves $\pi^1, \ldots, \pi^k$, where $\pi^s = \pi_{p_s..b_s}$ and $\|\pi_{p_s} - \pi_{b_s}\| \geq \lambda$ for all s. Then for any $z \in \mathbb{R}^d$, $r > 0$ the number of subcurves π^s intersecting $B(z, r)$ is bounded by $\mathcal{O}(\kappa^d(1 + r/\lambda)^d)$.*

Proof. Let $\pi^{s_1}, \ldots, \pi^{s_\ell}$ be the subcurves that intersect the ball $B = B(z, r)$. Let $X = \{s_1, s_3, \ldots,\}$ be the odd indices among the intersecting subcurves. For all $s \in X$ pick any point π_{x_s} in $\pi^s \cap B$. Between any points $\pi_{x_s}, \pi_{x_{s'}}$ there must lie an even subcurve $\pi^{s_{2i}}$. As the endpoints of this even subcurve have distance at least λ, we have $\|\pi_{x_s} - \pi_{x_{s'}}\| \geq \lambda/(\kappa+1)$. Otherwise, setting $r' := \frac{\kappa}{2}\|\pi_{x_s} - \pi_{x_{s'}}\|$, the even part would not fit into $B(\pi_{x_s}, r') \cup B(\pi_{x_{s'}}, r')$ which has diameter $(\kappa+1)\|\pi_{x_s} - \pi_{x_{s'}}\|$. Hence, the balls $B(\pi_{x_s}, \lambda/2(\kappa+1))$ are disjoint and contained in $B(z, r+\lambda)$. A standard packing argument now shows that $\ell \leq 2 \cdot (r+\lambda)^d/(\lambda/2(\kappa+1))^d = \mathcal{O}(\kappa^d(1 + r/\lambda)^d)$. □

Lemma 11.2.10. *For any κ-bounded curves π, σ with n vertices in total, $0 < \varepsilon \leq 1$, we have $N(\pi, \sigma, \varepsilon) = \mathcal{O}((\kappa/\sqrt{\varepsilon})^d n)$.*

Proof. Let $\delta > 0$ and consider the partitionings into long segments and pieces $\pi^1, \ldots, \pi^k$, $\sigma^1, \ldots, \sigma^\ell$ computed by our algorithm. Then $\sigma^t = \sigma_{q_t..d_t}$ satisfies $\|\sigma_{q_t} - \sigma_{d_t}\| \geq \Lambda = \min\{\frac{1}{2}\sqrt{\varepsilon}, \frac{1}{4}\} \cdot \delta$ for all t. We use the same charging scheme as in Lemma 11.2.6. Consider any segment v of a piece π^s. The segment v can be charged by a part σ^t which is either a long segment or a piece. In both cases, σ^t intersects the ball B centered at the midpoint of $\|v\|$ with radius $r := (1+\varepsilon)\delta + 2\Lambda$. By Lemma 11.2.9 with $\lambda := \Lambda$, the number of such charges is bounded by $\mathcal{O}((\kappa/\sqrt{\varepsilon})^d)$.

Now consider any long segment v of π. The segment v can be charged by segments of σ which are longer than v. Any such charging gives rise to a long segment σ^t intersecting

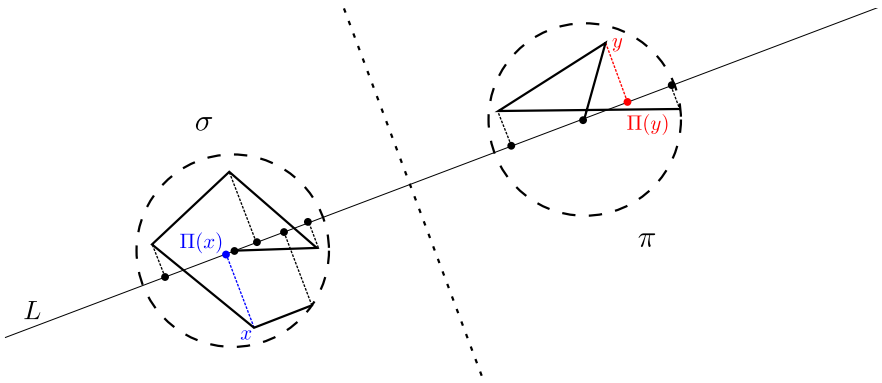


FIGURE 11.2: Projection of the pieces π, σ onto the line L through their initial vertices. This yields one-dimensional separated curves $\hat{\pi}, \hat{\sigma}$.

the ball B centered at the midpoint of v of radius $r := (1+\varepsilon)\delta + \frac{1}{2}\|v\|$. By Lemma 11.2.9 with $\lambda := \|v\|$, the number of such charges is bounded by $\mathcal{O}(\kappa^d(\frac{3}{2} + (1+\varepsilon)\delta/\|v\|)^d) = \mathcal{O}((\kappa/\sqrt{\varepsilon})^d)$, since $\|v\| \geq \Lambda = \min\{\frac{1}{2}\sqrt{\varepsilon}, \frac{1}{4}\} \cdot \delta$.

Hence, every segment of π is charged $\mathcal{O}((\kappa/\sqrt{\varepsilon})^d)$ times; a symmetric statement holds for σ. □

Together with Lemmas 11.1.1 and 11.2.5, the above lemma yields the following result. The best previously known running time was $\mathcal{O}((\kappa/\varepsilon)^d n + \kappa^d n \log n)$ [DHPW12].

Theorem 11.2.11. *For any $0 < \varepsilon \leq 1$ there is a $(1+\varepsilon)$-approximation for the continuous and discrete Fréchet distance on κ-bounded curves with n vertices in total in time $\mathcal{O}((\kappa/\sqrt{\varepsilon})^d n \log^2 1/\varepsilon + \kappa^d n \log n) = \tilde{\mathcal{O}}((\kappa/\sqrt{\varepsilon})^d n)$.*

11.2.3 Solving the Free-Space Region Problem on Pieces

It remains to prove Lemma 11.2.2. Let $(\pi, \sigma, \delta, \varepsilon)$ be an instance of the free-space region problem, where $n := |\pi|$, $m := |\sigma|$, with $\|\pi_1 - \pi_x\|, \|\sigma_1 - \sigma_y\| \leq \Lambda_{\varepsilon,\delta} = \Lambda$ for any $x \in [1, n]$, $y \in [1, m]$ (and entry intervals $\tilde{R}^h_{i,1} \subseteq [i, i+1] \times \{1\}$ for $i \in [1..n)$ and $\tilde{R}^v_{1,j} \subseteq \{1\} \times [j, j+1]$ for $j \in [1..m)$). We reduce this instance to the free-space region problem on *one-dimensional separated* curves, i.e., curves $\hat{\pi}, \hat{\sigma}$ in $\mathbb{R}$ such that all vertices of $\hat{\pi}$ lie above 0 and all vertices of $\hat{\sigma}$ lie below 0.

Since π and σ stay within distance Λ of their initial vertices, if their initial vertices are within distance $\|\pi_1 - \sigma_1\| \leq \delta - 2\Lambda$ then all pairs of points in π, σ are within distance δ. In this case, we find a translation of π making $\|\pi_1 - \sigma_1\| = \delta - 2\Lambda$ and all pairwise distances are still at most δ. This ensures that the curves π, σ are contained in disjoint balls of radius $\Lambda \leq \frac{1}{4}\delta$ centered at their initial vertices.

Consider the line L through the initial vertices π_1 and σ_1. Denote by $\Pi\colon \mathbb{R}^d \to L$ the projection onto L. Now, instead of the pieces π, σ we consider their projections $\hat{\pi} := \Pi(\pi) = (\Pi(\pi_1), \ldots, \Pi(\pi_n))$ and $\hat{\sigma} := \Pi(\sigma) = (\Pi(\sigma_1), \ldots, \Pi(\sigma_m))$, see Figure 11.2. Note that after rotation and translation we can assume that $\hat{\pi}$ and $\hat{\sigma}$ lie on $\mathbb{R} \subset \mathbb{R}^d$ and $\hat{\pi}$ and $\hat{\sigma}$ are separated by $0 \in \mathbb{R}$ (since π and σ are contained in disjoint balls centered on L). Now we solve the free-space region problem on $\hat{\pi}$, $\hat{\sigma}$, $\hat{\delta} := \delta$, and $\hat{\varepsilon} := \frac{1}{2}\varepsilon$ (with the same entry intervals $\tilde{R}^h_{i,j}, \tilde{R}^v_{i,j}$)).

Lemma 11.2.12. *Any solution to the free-space region problem on $(\hat{\pi}, \hat{\sigma}, \hat{\delta}, \hat{\varepsilon})$ solves the free-space region problem on $(\pi, \sigma, \delta, \varepsilon)$.*

Proof. Let x, y be vertices of π, σ, respectively. Clearly, $\|\Pi(x) - \Pi(y)\| \leq \|x - y\|$. Hence, any monotone path in $\mathcal{D}_{\leq\delta}(\pi, \sigma)$ yields a monotone path in $\mathcal{D}_{\leq\delta}(\hat{\pi}, \hat{\sigma}) = \mathcal{D}_{\leq\hat{\delta}}(\hat{\pi}, \hat{\sigma})$, so it will be found.

Note that x and y have distance at most Λ to L. Since $\Pi(x) - \Pi(y)$ and $x - \Pi(x) - (y - \Pi(y))$ are orthogonal, we can use the Pythagorean theorem to obtain

$$\|x - y\| = \sqrt{\|\Pi(x) - \Pi(y)\|^2 + \|x - \Pi(x) - (y - \Pi(y))\|^2} \leq \sqrt{\|\Pi(x) - \Pi(y)\|^2 + (2\Lambda)^2}.$$

Hence, any monotone path in $\mathcal{D}_{\leq(1+\hat{\varepsilon})\hat{\delta}}(\hat{\pi}, \hat{\sigma})$ yields a monotone path in $\mathcal{D}_{\leq\alpha}(\pi, \sigma)$ with $\alpha \leq \sqrt{(1+\hat{\varepsilon})^2\hat{\delta}^2 + (2\Lambda)^2}$. Plugging in $\hat{\delta} = \delta$, $\hat{\varepsilon} = \frac{1}{2}\varepsilon$, and $\Lambda = \min\{\frac{1}{2}\sqrt{\varepsilon}, \frac{1}{4}\} \cdot \delta$ we obtain $\alpha \leq \sqrt{(1+\frac{1}{2}\varepsilon)^2 + \varepsilon} \cdot \delta \leq (1+\varepsilon)\,\delta$. Thus, the desired guarantees for the free-space region problem are satisfied. □

Lemma 11.2.2 now follows from Lemma 11.3.1 below, that we prove in the next section.

11.3 On One-Dimensional Separated Curves

In this section, we show the following lemma.

Lemma 11.3.1. *The free-space region problem on one-dimensional separated curves can be solved in time* $\mathcal{O}((n+m)\log^2 1/\varepsilon)$.

First, in Section 11.3.1, we show how to reduce this problem to a *discrete* version, meaning that we can eliminate the continuous Fréchet distance and only consider the much simpler discrete Fréchet distance (for general curves such a reduction is not known to exist, but we only need it for one-dimensional separated curves). Moreover, we simplify our curves further by rounding the vertices. This yields a reduction to the following subproblem. Note that we no longer ask for an approximation algorithm.

Problem 11.3.2 (Reduced free-space problem). Given $\delta > 0$ and $0 < \varepsilon \leq 1$, given one-dimensional separated curves π, σ with n, m vertices and all vertex coordinates being multiples of $\frac{1}{4}\varepsilon\delta$, and given an entry set $E \subseteq [1..n]$, compute the exit set $F^\pi \subseteq [1..n]$ consisting of all points f such that $d_{\mathrm{dF}}(\pi_{e..f}, \sigma) \leq \delta$ for some $e \in E$ and the exit set $F^\sigma \subseteq [1..m]$ consisting of all points f such that $d_{\mathrm{dF}}(\pi_{e..n}, \sigma_{1..f}) \leq \delta$ for some $e \in E$.

Lemma 11.3.3. *The reduced free-space problem can be solved in time* $\mathcal{O}((n+m)\log 1/\varepsilon)$.

As a second step, we prove the above lemma. We first consider the special case of $E = \{1\}$ and the problem of deciding whether $n \in F^\pi$, i.e., the lower left corner $(1, 1)$ of the free-space is the only entry point and we want to determine whether the upper right corner (n, m) is an exit. This is equivalent to deciding whether the discrete Fréchet distance of π, σ is at most δ, which has an easy near-linear time algorithm as π, σ are one-dimensional and separated. We present a greedy algorithm for this special case in Section 11.3.2. To extend this to the reduced free-space problem, we prove useful structural properties of one-dimensional separated curves in Section 11.3.3. With these, we first solve the problem of determining the exit set F^π assuming $E = \{1\}$ in Section 11.3.4. Then we show for general $E \subseteq [1..n]$ how to compute F^π (Section 11.3.4) and F^σ (Section 11.3.4).

11.3.1 Reduction from the Continuous to the Discrete Case

We show that on one-dimensional separated curves, discrete and continuous Fréchet distance coincide. Moreover, subdividing one-dimensional separated curves does not change their discrete Fréchet distance.

Lemma 11.3.4. *Let π, σ be one-dimensional separated curves. Then we have $d_{\mathrm{F}}(\pi,\sigma) = d_{\mathrm{dF}}(\pi,\sigma)$. Moreover, assume that we subdivide any segments of π, σ by adding new vertices along the segments, yieldings curves π', σ'. Then we have $d_{\mathrm{dF}}(\pi',\sigma') = d_{\mathrm{dF}}(\pi,\sigma)$.*

Proof. It is known that $d_{\mathrm{F}}(\pi,\sigma) \le d_{\mathrm{dF}}(\pi,\sigma)$ holds for all curves π, σ. Thus, we only need to show that any continuous traversal $\phi = (\phi_1, \phi_2)$ of π, σ can be transformed into a discrete traversal with the same width. We adapt ϕ as follows. For any point in time $t \in [0,1]$, if $\phi_1(t)$ is at a vertex of π we set $\phi_1'(t) := \phi_1(t)$. Otherwise $\phi_1(t)$ is in the interior of a segment $\pi_{i..i+1}$ of π. Let $j \in \{i, i+1\}$ minimize π_j. We set $\phi_1'(t) := j$. Observe that ϕ_1' indeed is a non-decreasing function from $[0,1]$ onto $[1..n]$. A similar construction, where we round to the value $j \in \{i, i+1\}$ maximizing σ_j, yields ϕ_2' and we obtain a discrete traversal $\phi' = (\phi_1', \phi_2')$. The width of ϕ' is at most the width of ϕ since we rounded in the right way, i.e., we have $\pi(\phi_1'(t)) \le \pi(\phi_1(t))$ and $\sigma(\phi_2'(t)) \ge \sigma(\phi_2(t))$ so that $\|\pi(\phi_1'(t)) - \sigma(\phi_2'(t))\| \le \|\pi(\phi_1(t)) - \sigma(\phi_2(t))\|$ for all $t \in [0,1]$.

Note that the discrete Fréchet distance is in general not preserved under subdivision of segments, but the continuous Fréchet distance is. Thus, the second statement follows from the first one, $d_{\mathrm{dF}}(\pi,\sigma) = d_{\mathrm{F}}(\pi,\sigma) = d_{\mathrm{F}}(\pi',\sigma') = d_{\mathrm{dF}}(\pi',\sigma')$. □

The above lemma enables the following trick. Consider any finite sets $E \subseteq [1,n]$ and $F \subseteq [1,n]$. Add π_x as a vertex to π for any $x \in E \cup F$, with slight abuse of notation we say that π now has vertices at π_i, $i \in [1..n]$, and π_x, $x \in E \cup F$. Mark the vertices π_x, $x \in E$, as entries. Now solve the reduced free-space problem instance (π, σ, E). This yields the set F^π of all values $f \in F$ such that there is an $e \in E$ with $d_{\mathrm{dF}}(\pi_{e..f}, \sigma) \le \delta$, which by Lemma 11.3.4 is equivalent to $d_{\mathrm{F}}(\pi_{e..f}, \sigma) \le \delta$. Thus, we computed all exit points in F given entry points in E, with respect to the continuous Fréchet distance. This is already close to a solution of the free-space region problem, however, we have to cope with entry and exit *intervals*.

For the full reduction we need two more arguments. First, we can replace all non-empty input intervals $\tilde{R}^h_{i,1}$ by the leftmost point $(y_i, 1)$ in $\tilde{R}^h_{i,1} \cap \mathcal{D}_{\le\delta}(\pi,\sigma)$, specifically, we show that any traversal starting in a point in $\tilde{R}^h_{i,1}$ can be transformed into a traversal starting in $(y_i, 1)$. Thus, we add π_{y_i} as a vertex and mark it as an entry to obtain a finite and small set of entry points. Second, for any segment $\pi_{i..i+1}$ we call a point $f \in [i, i+1]$ *reachable* if there is an $e \in E$ with $d_{\mathrm{F}}(\pi_{e..f}, \sigma) \le \delta$. We show that if f is reachable then essentially all points $f' \in [i, i+1]$ with $\pi_{f'} \le \pi_f$ are also reachable. Thus, the set of reachable points is an interval with one trivial endpoint, and we only need to search for the other endpoint of the interval, which can be done by binary search. Moreover, we can parallelize all these binary searches, as solving one reduced free-space problem can answer for every segment of π whether a particular point on this segment is reachable (after adding this point as a vertex). To make these binary searches finite, we round all vertices of π and σ to multiples of $\gamma := \frac{1}{4}\varepsilon\delta$ and only search for exit points that are multiples of γ. This is allowed since the free-space region problem only asks for an approximate answer. A similar procedure yields the exits on σ reachable from entries on π, and determining the exits reachable from entries on σ is a symmetric problem. Since for the binary searches we reduce to $\mathcal{O}(\log 1/\varepsilon)$ instances of the reduced free-space problem, Lemma 11.3.1 follows from Lemma 11.3.3.

In the following we present the details of this approach. Let π, σ be one-dimensional separated curves, i.e., they are contained in $\mathbb{R}$, all vertices of π lie above 0, and all vertices of σ lie below 0. Let $n = |\pi|$, $m = |\sigma|$, $\delta > 0$ and $0 < \varepsilon \leq 1$. Consider entry intervals $\tilde{R}^h_{i,1} \subseteq [i, i+1] \times \{1\}$ for $i \in [1..n)$ and $\tilde{R}^v_{1,j} \subseteq \{1\} \times [j, j+1]$ for $j \in [1..m)$. We reduce this instance of the free-space region problem to $\mathcal{O}(\log 1/\varepsilon)$ instances of the reduced free-space problem.

First we change π, σ as follows. (1) Let $Z \subset \mathbb{R}$ be the set of all integral multiples[1] of $\gamma := \frac{1}{4}\varepsilon\delta$. We round all vertices of π, σ to values in Z, where we round down everything in π and round up in σ, yielding curves π', σ'. (2) Let $I \subseteq [1..n)$ be the set of all i with nonempty $\tilde{R}^h_{i,1} \cap \mathcal{D}_{\leq\delta}(\pi', \sigma')$. For any $i \in I$ let $(y'_i, 1)$ be the leftmost point in $\tilde{R}^h_{i,1} \cap \mathcal{D}_{\leq\delta}(\pi', \sigma')$ and let $y_i \leq y'_i$ be maximal with $\pi'_{y_i} \in Z$. Add π'_{y_i} as a vertex to π' and mark it as an entry. With slight abuse of notation, we say that π' now has its vertices at π'_i, $i \in [1..n]$ and π'_{y_i}, $i \in I$. We let $E = \{y_i \mid i \in I\}$ be the indices of the entry vertices. Note that (π', σ', E) can be computed in time $\mathcal{O}(n + m)$.

For every $i \in [1..n)$ consider the multiples of γ on $\pi'_{i..i+1}$, i.e., $S_i := \{x \in [i, i+1] \mid \pi'_x \in Z\}$. Note that S_i is either $[i, i+1]$ (if $\pi_i = \pi_{i+1}$) or it forms an arithmetic progression, specifically $S_i = \{i, i+1/t_i, i+2/t_i, \ldots, i+1\}$ for some $t_i \in \mathbb{N}$, since π'_i, π'_{i+1} are in Z and π'_x is a linear function in x. Thus, S_i and subsequences of S_i can be handled efficiently, we omit these details in the following. We want to determine the set F_i of all $f \in S_i$ such that there is an $e \in E$ with $d_{\mathrm{dF}}(\pi'_{e..f}, \sigma') \leq \delta$. We first argue that F_i is of an easy form.

Lemma 11.3.5. *If F_i is non-empty then we have $F_i = [a, b] \cap S_i$ for some $a, b \in S_i$ with $\{a, b\} \cap \{i, y_i, i+1\} \neq \emptyset$ (or $\{a, b\} \cap \{i, i+1\} \neq \emptyset$ if y_i does not exist).*

Proof. We show that if any $f \in S_i$ is reachable, i.e., there is an $e \in E$ with $d_{\mathrm{dF}}(\pi'_{e..f}, \sigma') \leq \delta$, then any $f' \in S_i$ with $\pi'_{f'} \leq \pi'_f$ and $y_i \notin (f', f]$ is also reachable. This proves the claim. Let ϕ be any traversal of $\pi'_{e..f}, \sigma'$ of width at most δ. Note that $e \leq f'$, since $y_i \notin (f', f]$ and y_i is the only entry on the segment containing f and f'. If $f' \leq f$ then we change ϕ to stop at $\pi'_{f'}$ once it arrives at this point, and we traverse the remaining part of σ staying fixed at $\pi'_{f'}$. Since $\pi'_{f'} \leq \pi'_f$ this does not increase the width of the traversal and shows that f' is also reachable. If $f' > f$ then we append a traversal to ϕ that stays fixed at σ'_m but walks in π' from π'_f to $\pi'_{f'}$. Again since $\pi'_{f'} \leq \pi'_f$ this does not increase the width of the traversal and shows that f' is also reachable. □

Note that by solving the reduced free-space problem on (π', σ', E) we decide for each $f \in [n] \cup \{y_i \mid i \in I\}$ whether there is an $e \in E$ with $d_{\mathrm{dF}}(\pi'_{e..f}, \sigma') \leq \delta$. By the above lemma, this yields one of the endpoints of the interval F_i, say a, and we only have to determine the other endpoint, say b. In the special case $\pi'_i = \pi'_{i+1}$ we even determined both endpoints already, so from now on we can assume $\pi'_i \neq \pi'_{i+1}$ so that $|S_i| < \infty$. We search for the other endpoint of F_i using a binary search over S_i. To test whether any $z \in S_i$ is in F_i, we add π'_z as a vertex of π' and solve the reduced free-space problem on (π', σ', E). If z is in the output set F^π then it is in F_i.

Note that any vertex $\pi'_x > \delta$ on π' does not have any point of σ within distance δ, which is preserved by setting $\pi'_x := 2\delta$. Thus, we can assume that π' takes values in $[0, 2\delta]$, which implies $|S_i| \leq \mathcal{O}(1/\varepsilon)$, so that our binary search needs $\mathcal{O}(\log 1/\varepsilon)$ steps. Moreover, note that we can parallelize these binary searches, since we can add a vertex z_i on every subcurve $\pi'_{i..i+1}$, so that one call to the reduced free-space problem determines for every z_i whether it is reachable. Here we use Lemma 11.3.4, since we need that further subdivision of some segments of π' does not change the discrete Fréchet distance.

[1] Without loss of generality we assume $1/\varepsilon \in \mathbb{N}$ so that $\delta \in Z$.

Note that since we add $\mathcal{O}(n)$ vertices to π' and since we need $\mathcal{O}(\log 1/\varepsilon)$ steps of binary search, Lemma 11.3.3 implies a total running time of $\mathcal{O}((n+m)\log^2 1/\varepsilon)$.

We thus computed $F_i = [a,b] \cap S_i$ with $a, b \in S_i$. We extend F_i slightly to $F'_i = [a',b'] \cap S_i$ by including the neighboring elements of a and b in S_i. Finally, we set $\tilde{R}^h_{i,m}(\pi) := [a',b'] \times \{m\}$. A similar procedure adding entries E^σ on σ' and doing a binary search over exits on π' yields an interval $\tilde{R}^h_{i,m}(\sigma)$ consisting of points $(f,m) \in [i,i{+}1]\times\{m\}$ such that there is an $e \in E^\sigma$ with $d_{\mathrm{dF}}(\pi'_{1..f}, \sigma'_{e..m}) \leq \delta$. We set $\tilde{R}^h_{i,m} := \tilde{R}^h_{i,m}(\pi) \cup \tilde{R}^h_{i,m}(\sigma)$, which will be again an interval (which follows from the proof of Lemma 11.3.5). A symmetric algorithm determines $\tilde{R}^v_{n,j}$ for $j \in [1..m)$.

We show that we correctly solve the given free-space region problem instance.

Lemma 11.3.6. *The computed intervals are a valid solution to the given free-space region instance.*

Proof. Let ϕ be any monotone path in $\mathcal{D}_{\leq\delta}(\pi,\sigma)$ that starts in a point $(p,1) \in \tilde{R}^h_{j,1}$ and ends in (b,m), witnessing that $d_{\mathrm{dF}}(\pi_{p..b}, \sigma) \leq \delta$. After rounding down π to π' and rounding up σ to σ', ϕ is still a monotone path in $\mathcal{D}_{\leq\delta}(\pi',\sigma')$. Moreover, we can prepend a path from $(y'_j,1)$ to $(p,1)$ to ϕ, since $\tilde{R}^h_{j,1} \cap \mathcal{D}_{\leq\delta}(\pi',\sigma')$ is an interval containing $(y'_j,1)$ and $(p,1)$. We can also prepend a path from $(y_j,1)$ to $(y'_j,1)$, since σ_1 is a multiple of γ and π'_{y_j} is at most $\pi'_{y'_j}$ rounded up to a multiple of γ. Let r be the value of π_b rounded down to a multiple of γ. This value r is attained at some point π'_f on the same segment $\pi'_{i..i+1}$ as π'_b. If $f \leq b$ then we change ϕ to stop at π'_f whenever it reaches this point. If $f > b$ then we change ϕ by appending a path from (b,m) to (f,m). In any case, this yields a monotone path in $\mathcal{D}_{\leq\delta}(\pi',\sigma')$ from $(y_j,1)$ to (f,m). Since such a continuous traversal is equivalent to a discrete traversal by Lemma 11.3.4, we have $f \in F_i$. By the construction of F'_i, the point (b,m) will be contained in the output $\tilde{R}^h_{i,m}(\pi)$, so we find the reachable exit (b,m) as desired. A similar argument with entries on σ shows that we satisfy property (1) of the free-space region problem.

Consider any point (f,m) in the constructed output set $\tilde{R}^h_{i,m}(\pi)$. By the construction of F'_i, there is a point b on the same segment as f with $|\pi'_b - \pi'_f| \leq \gamma$ and there is an entry $e \in E$ with $d_{\mathrm{dF}}(\pi'_{e..b}, \sigma') \leq \delta$, witnessed by a traversal ϕ. By the construction of E, there is a point $y \in \tilde{R}^h_{j,1}$, for some j, with $e \leq y$ and $|\pi'_y - \pi'_e| \leq \gamma$. First assume $y \leq f, b$. In this case, we change ϕ so that it starts at π'_y and stays there while ϕ is at any point π'_x, $x \leq y$. Moreover, if $b \leq f$ we change ϕ so that it stops at π'_b once it reaches this point, and if $b > f$ we change ϕ by appending a path from (b,m) to (f,m). This shows $d_{\mathrm{dF}}(\pi'_{y..f}, \sigma') \leq \delta + \gamma$. Since π', σ' are rounded versions of π, σ where all vertices are moved by less than γ, we obtain $d_{\mathrm{dF}}(\pi_{y..f}, \sigma) \leq \delta + 3\gamma \leq (1+\varepsilon)\delta$. In the remaining cases $e \leq f \leq y$ and $e \leq b \leq y$, we have $|\pi'_y - \pi'_x| \leq 2\gamma$ for all $x \in \{e,b,f\}$. Hence, a traversal staying fixed in π'_y does the job, i.e., $d_{\mathrm{dF}}(\pi'_y, \sigma') \leq \delta{+}2\gamma$ and $d_{\mathrm{dF}}(\pi_y, \sigma) \leq \delta{+}4\gamma = (1{+}\varepsilon)\delta$. Thus, any point (f,m) in the output set is reachable from the entry sets by a monotone path in $\mathcal{D}_{\leq(1+\varepsilon)\delta}(\pi,\sigma)$, which together with a similar argument for entries on σ proves that we satisfy property (2) of the free-space problem. □

11.3.2 Greedy Decider for One-Dimensional Separated Curves

Before solving the reduced free-space problem, let us consider the simpler problem of deciding $d_{\mathrm{dF}}(\pi,\sigma) \leq \delta$ for one-dimensional separated curves π, σ. In this section, we present a near-linear time algorithm for this problem, by walking along π and σ with *greedy steps* to either find a feasible traversal or bottleneck subcurves. We are not the

first to have this observation[2], which in any case is not the focus of this work. Instead, we are interested in the (quite complex) extension of this result to the reduced free-space problem (see Section 11.3.4), which we use as a subroutine for our main result on c-packed curves.

In the remainder of the chapter, all indices of curves will be integral. Let $\pi = (\pi_1, \ldots, \pi_n)$ and $\sigma = (\sigma_1, \ldots, \sigma_m)$ be two separated polygonal curves in $\mathbb{R}$, i.e., $\pi_i \geq 0 \geq \sigma_j$. For indices $1 \leq i \leq n$ and $1 \leq j \leq m$, define $\mathrm{vis}_\sigma(i,j) := \{k \mid k \geq j \text{ and } \sigma_k \geq \pi_i - \delta\}$ as the index set of vertices on σ that are later in sequence than σ_j and are still in distance δ to π_i (i.e, *seen* by π_i) and, likewise, $\mathrm{vis}_\pi(i,j) := \{k \mid k \geq i \text{ and } \pi_k \leq \sigma_j + \delta\}$. Hence, the set of points that we may reach on σ by starting in (π_i, σ_j) and staying in π_i can be defined as the longest contiguous subsequence $[j+1..j+k]$ such that $[j+1..j+k] \subseteq \mathrm{vis}_\sigma(i,j)$. Let $\mathrm{reach}_\sigma(i,j) := [j+1..j+k]$ denote this subsequence and let $\mathrm{reach}_\pi(i,j)$ be defined symmetrically. Note that $\pi_i \leq \pi_{i'}$ implies that $\mathrm{vis}_\sigma(i,j) \supseteq \mathrm{vis}_\sigma(i',j)$, however the converse does not necessarily hold. Also, $\mathrm{vis}_\sigma(i,j) \not\supseteq \mathrm{vis}_\sigma(i',j)$ implies that $\mathrm{vis}_\sigma(i,j) \subsetneq \mathrm{vis}_\sigma(i',j)$ and $\pi_i > \pi_{i'}$.

The visibility sets established above enable us to define a greedy algorithm for the Fréchet distance of π and σ. Let $1 \leq p \leq n$ and $1 \leq q \leq m$ be arbitrary indices on σ and π. We say that p' is a *greedy step on π from (p,q)*, written $p' \leftarrow \textsc{GreedyStep}_\pi(\pi_{p..n}, \sigma_{q..m})$, if $p' \in \mathrm{reach}_\pi(p,q)$ and $\mathrm{vis}_\sigma(i,q) \subseteq \mathrm{vis}_\sigma(p',q)$ holds for all $p \leq i \leq p'$. Symmetrically, $q' \in \mathrm{reach}_\sigma(p,q)$ is a *greedy step on σ from (p,q)*, if $\mathrm{vis}_\pi(p,i) \subseteq \mathrm{vis}_\pi(p,q')$ for all $q \leq i \leq q'$. In pseudo code, $\textsc{GreedyStep}_\pi(\pi_{p..n}, \sigma_{q..m})$ denotes a function that returns an arbitrary greedy step p' on π from (p,q) if such an index exists and returns an error otherwise (symmetrically for σ).

Consider the following greedy algorithm:

Algorithm 2 Greedy Fréchet distance decider for separated curves $\pi_{1..n}$ and $\sigma_{1..m}$ in $\mathbb{R}$

1: $p \leftarrow 1, q \leftarrow 1$
2: **repeat**
3: **if** $p' \leftarrow \textsc{GreedyStep}_\pi(\pi_{p..n}, \sigma_{q..m})$ **then** $p \leftarrow p'$
4: **if** $q' \leftarrow \textsc{GreedyStep}_\sigma(\pi_{p..n}, \sigma_{q..m})$ **then** $q \leftarrow q'$
5: **until** no greedy step was found in the last iteration
6: **if** $p = n$ and $q = m$ **then return** $d_{\mathrm{dF}}(\pi, \sigma) \leq \delta$
7: **else return** $d_{\mathrm{dF}}(\pi, \sigma) > \delta$

Theorem 11.3.7. *Let π and σ be separated curves in $\mathbb{R}$ and $\delta > 0$. Algorithm 2 decides whether $d_{\mathrm{dF}}(\pi, \sigma) \leq \delta$ in time $\mathcal{O}((n+m)\log(nm))$.*

We will first prove the correctness of the algorithm in Lemma 11.3.9 below and postpone the discussion how to implement the algorithm efficiently to Section 11.3.2.

Correctness

Note that Algorithm 2 considers potentially only very few points of the curve explicitly during its execution. Call the indices (p,q) of point pairs considered in some iteration of the algorithm (for any choice of greedy steps, if more than one exists) *greedy (point) pairs* and all points contained in some such pair *greedy points (of π and σ)*. The following useful monotonicity property holds: If some greedy point on π sees a point on σ that is yet to be traversed, all following greedy points on π will see it *until it is traversed*.

[2]We thank Wolfgang Mulzer for a pointer to this (unpublished) result by Matias Korman and Sergio Cabello (personal communication).

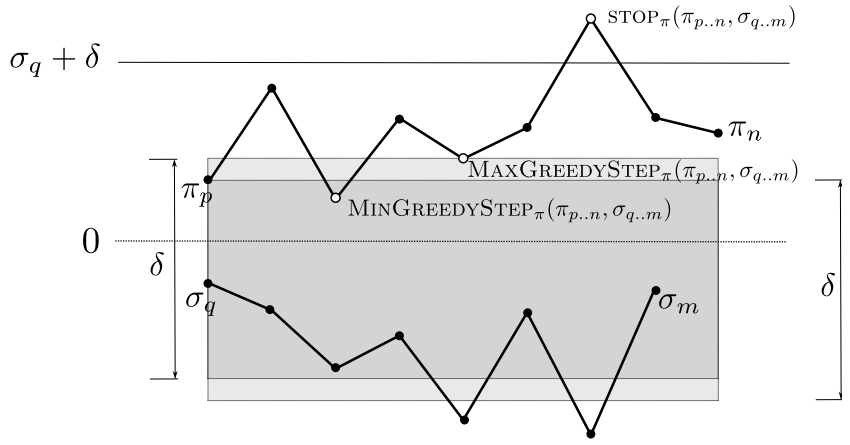


FIGURE 11.3: An illustration of greedy steps. For better visibility, the one-dimensional separated curves π, σ are drawn in the plane by mapping π_i to (i, π_i). In particular, the results of $\text{MINGREEDYSTEP}_\pi(\pi_{p..n}, \sigma_{q..m})$, $\text{MAXGREEDYSTEP}_\pi(\pi_{p..n}, \sigma_{q..m})$, and $\text{STOP}_\pi(\pi_{p..n}, \sigma_{q..m})$ are shown.

Lemma 11.3.8. *Let $(p_1, q_1), \ldots, (p_i, q_i)$ be the greedy point pairs considered in the iterations $1, \ldots, i$. It holds that*

1. $\text{vis}_\sigma(\ell, q_i) \subseteq \text{vis}_\sigma(p_i, q_i)$ *for all* $1 \leq \ell \leq p_i$, *and*
2. $\text{vis}_\pi(p_i, \ell) \subseteq \text{vis}_\pi(p_i, q_i)$ *for all* $1 \leq \ell \leq q_i$.

Proof. Let $k < i$. We first show that $\text{vis}_\sigma(\ell, q_i) \subseteq \text{vis}_\sigma(p_{k+1}, q_i)$ holds for all $p_k \leq \ell < p_{k+1}$. If $p_k = p_{k+1}$, the claim is immediate. Otherwise p_{k+1} is the result of a greedy step on π. By definition of visibility, we have $\text{vis}_\sigma(\ell, q_i) = \text{vis}_\sigma(\ell, q_k) \cap [q_i..m] \subseteq \text{vis}_\sigma(p_{k+1}, q_k) \cap [q_i..m] = \text{vis}_\sigma(p_{k+1}, q_i)$, where the inequality follows from p_{k+1} being a greedy step from (p_k, q_k).

For arbitrary $\ell \leq p_i$, let $k < i$ be such that $p_k \leq \ell < p_{k+1}$. Then $\text{vis}_\sigma(\ell, q_i) \subseteq \text{vis}_\sigma(p_{k+1}, q_i) \subseteq \text{vis}_\sigma(p_{k+2}, q_i) \subseteq \cdots \subseteq \text{vis}_\sigma(p_i, q_i)$. The second statement is symmetric. □

We will exploit this monotonicity to prove that if Algorithm 2 finds a greedy point pair that allows no further greedy steps, then no feasible traversal of π and σ exists. We derive an even stronger statement using the following notion: For a greedy point pair (p, q), define $\text{STOP}_\pi(\pi_{p..n}, \sigma_{q..m}) := \max(\text{reach}_\pi(p, q) \cup \{p\}) + 1$ as the index of the first point after π_p on π which is not seen by σ_q, or $n + 1$ if no such index exists. Let STOP_σ be defined symmetrically.

Lemma 11.3.9 (Correctness of Algorithm 2)**.** *Let (p, q) be a greedy point of π and σ, $p_{\text{stop}} := \text{STOP}_\pi(\pi_{p..n}, \sigma_{q..m})$ and $q_{\text{stop}} := \text{STOP}_\sigma(\pi_{p..n}, \sigma_{q..m})$. If on both curves no greedy step from (p, q) exists, then $d_{\text{dF}}(\pi, \sigma) > \delta$.*

In particular, if $q_{\text{stop}} \leq m$, then for all $1 \leq p' \leq n$, we have that $d_{\text{dF}}(\pi_{1..p'}, \sigma_{1..q_{\text{stop}}}) > \delta$ and if $p_{\text{stop}} \leq n$, then $d_{\text{dF}}(\pi_{1..p_{\text{stop}}}, \sigma_{1..q'}) > \delta$ for all $1 \leq q' \leq m$.

Note that the correctness of Algorithm 2 follows immediately: If the algorithm is stuck, then $d_{\text{dF}}(\pi, \sigma) > \delta$. Otherwise, it finds a feasible traversal.

Proof of Lemma 11.3.9. Consider the case that no greedy step from (p, q) exists, then the following *stuckness conditions* have to hold:

1. For all $p' \in \text{reach}_\pi(p, q)$, we have $\text{vis}_\sigma(p', q) \subsetneq \text{vis}_\sigma(p, q)$, and

2. for all $q' \in \text{reach}_\sigma(p,q)$, we have $\text{vis}_\pi(p,q') \subsetneq \text{vis}_\pi(p,q)$.

In this case, we can extend the monotonicity property of Lemma 11.3.8 to include all reachable and the first unreachable point.

Claim 11.3.10. *If the stuckness conditions hold for (p,q), then we have $\text{vis}_\sigma(i,q) \subseteq \text{vis}_\sigma(p,q)$ for all $1 \leq i \leq p_{\text{stop}}$. In particular, if π_p does not see σ_ℓ for some $\ell > q$, then no vertex π_i with $1 \leq i \leq p_{\text{stop}}$ sees σ_ℓ. The symmetric statement holds for σ.*

Proof. By Lemma 11.3.8, $\text{vis}_\sigma(i,q) \subseteq \text{vis}_\sigma(p,q)$ holds for all $i \leq p$. The first of the stuckness conditions implies $\text{vis}_\sigma(i,q) \subseteq \text{vis}_\sigma(p,q)$ for all $p < i < p_{\text{stop}}$. If $p_{\text{stop}} = n+1$, this already completes the proof of the claim. Otherwise, note that $\pi_{p_{\text{stop}}} > \pi_p$, since otherwise $p_{\text{stop}} \in \text{reach}_\pi(p,q)$. Hence $\text{vis}_\sigma(p_{\text{stop}},q) \subseteq \text{vis}_\sigma(p,q)$ holds as well. □

We distinguish the following cases that may occur under the stuckness conditions.

Case 1: $p_{\text{stop}} \leq n$ or $q_{\text{stop}} \leq m$. Without loss of generality, let $p_{\text{stop}} \leq n$ (the other case is symmetric). Assume for contradiction that a feasible traversal ϕ of $\pi_{1..p_{\text{stop}}}$ and $\sigma_{1..q'}$ exists for some $1 \leq q' \leq m$. In ϕ, at some point in time we have to move in π from $p_{\text{stop}}-1$ to p_{stop} while moving in $\sigma_{1..q'}$ from some $\sigma_{\ell'}$ to σ_ℓ where $\ell' \in \{\ell-1,\ell\}$ and σ_ℓ sees $\pi_{p_{\text{stop}}}$. Since σ_q does not see $\pi_{p_{\text{stop}}}$, the previous claim shows that $\ell > q_{\text{stop}}$. If $q_{\text{stop}} = m+1$ or $q_{\text{stop}} > q'$, this is impossible, yielding a contradiction. Otherwise, to do this transition, in some earlier step we have to move in σ from $q_{\text{stop}}-1$ to q_{stop} while moving in π from $\pi_{k'}$ to π_k for some $k < p_{\text{stop}}$ and $k' \in \{k-1,k\}$. However, by definition $q_{\text{stop}} \notin \text{vis}_\sigma(p,q)$, hence Claim 11.3.10 implies that the transition is illegal, since π_k does not see $\sigma_{q_{\text{stop}}}$. This is a contradiction. By a symmetric argument, it holds that $d_{\text{dF}}(\pi_{1..p'}, \sigma_{1..q_{\text{stop}}}) > \delta$.

Case 2: $p_{\text{stop}} = n+1$ and $q_{\text{stop}} = m+1$. In particular, this implies $p < n$ and $q < m$. In this case, $\text{reach}_\pi(p,q) = [p+1..n]$ and $\text{reach}_\sigma(p,q) = [q+1..m]$. By stuckness conditions, there exist an index $p_{\max} > p$ such that no $\sigma_{q'}$ with $q' > q$ sees $\pi_{p_{\max}}$ and an index $q_{\min}$ such that no $\pi_{p'}$ with $p' > p$ sees $\sigma_{q_{\min}}$. Assume for contradiction that a feasible traversal ϕ exists. In ϕ, at some point in time t, we have to cross either (1) from π_p to π_{p+1} while moving in σ from $\sigma_{\ell'}$ to σ_ℓ with $\ell \leq q+1 \leq q_{\min}$ and $\ell' \in \{\ell-1,\ell\}$ or (2) from σ_q to σ_{q+1} while moving from $\pi_{\ell'}$ to π_ℓ with $\ell \leq p+1 \leq p_{\max}$ and $\ell' \in \{\ell-1,\ell\}$. In the first case, $\ell < q_{\min}$ holds, since π_{p+1} does not see $\sigma_{q_{\min}}$. For all consecutive times $t' \geq t$, ϕ is in a point $\pi_{p'}$ ($p' \geq p+1$) that does not see $\sigma_{q_{\min}}$, which still has to be traversed, leading to a contradiction. Symmetrically, in the second case, for all times $t' \geq t$, ϕ is in a point $\sigma_{q'}$ ($q' \geq q+1$) that does not see $\pi_{p_{\max}}$, which still has to be traversed.

This concludes the proof of Lemma 11.3.9. □

Implementing Greedy Steps

To prove Theorem 11.3.7, it remains to show how to implement the algorithm to run in time $\mathcal{O}((n+m)\log(nm))$. We make use of geometric range search queries. The classic technique of fractional cascading [Lue78; CG86; Wil78] provides a data structure D with the following properties: (i) Given n points $\mathcal{P}$ in the plane, $D(\mathcal{P})$ can be constructed in time $\mathcal{O}(n \log n)$ and (ii) given a query rectangle $Q := I_1 \times I_2$ with intervals I_1 and I_2, find and return $q \in Q \cap \mathcal{P}$ with minimal y-coordinate, or report that no such point exists, in time $\mathcal{O}(\log n)$. Here, each interval I_i may be open, half-open or closed.

By invoking the above data structure on $\mathcal{P} := \{(i,\pi_i) \mid i \in [1 \ldots n]\}$ for a given curve $\pi = \pi_{1..n}$ (as well as all three rotations of $\mathcal{P}$ by multiples of 90°), we obtain a data structure D^π such that:

1. D^π can be constructed in time $\mathcal{O}(n \log n)$,
2. the query $\mathrm{D}^\pi.\textsc{minIndex}([x_1, x_2], [p, b])$ ($\mathrm{D}^\pi.\textsc{maxIndex}([x_1, x_2], [p, b])$) returns the minimum (maximum) index $p \leq i \leq b$ such that $x_1 \leq \pi_i \leq x_2$ in time $\mathcal{O}(\log n)$, and
3. the query $\mathrm{D}^\pi.\textsc{minHeight}([x_1, x_2], [p, b])$ ($\mathrm{D}^\pi.\textsc{maxHeight}([x_1, x_2], [p, b])$) returns the minimum (maximum) height $x_1 \leq \pi_i \leq x_2$ such that $p \leq i \leq b$ in time $\mathcal{O}(\log n)$.

The queries extend naturally to open and half-open intervals. If no index exists in the queried range, all of these operations return the index ∞, in case of minimization, or $-\infty$, in case of maximization. We will use the corresponding data structure D^σ for σ as well.

With these tools, we implement the following basic operations for arbitrary subcurves $\pi' := \pi_{p..b}$ and $\sigma' := \sigma_{q..d}$ of π and σ. See also Figure 11.3.

1. **Stopping point** $\textsc{stop}_\pi(\pi', \sigma')$**.** For points p, q, the primitive $\textsc{stop}_\pi(\pi', \sigma') := \max(\mathrm{reach}_{\pi'}(p, q) \cup \{p\}) + 1$ returns the index of the first point after π_p on π' which is not seen by σ_q, or $b + 1$ if no such index exists.

Algorithm 3 Finding the stopping point

```
1: function STOP_π(π_{p..b}, σ_{q..d})
2:     p_stop ← D^π.minIndex((σ_q + δ, ∞), [p, b])      ▷ First non-visible point on π
3:     if p_stop < ∞ then return p_stop
4:     else return b + 1
```

2. **Minimal greedy step** $\textsc{MinGreedyStep}_\pi(\pi', \sigma')$**.** This function returns the smallest index $p' \in \mathrm{reach}_{\pi'}(p, q)$ such that $\mathrm{vis}_{\sigma'}(p', q) \supseteq \mathrm{vis}_{\sigma'}(p, q)$ or reports that no such index exists.

Algorithm 4 Minimal greedy step

```
1: function MinGreedyStep_π(π_{p..b}, σ_{q..d})
2:     q_min ← D^σ.minHeight([π_p − δ, ∞), [q, d])      ▷ Lowest still visible point on σ
3:     p_cand ← D^π.minIndex((−∞, σ_{q_min} + δ], [p + 1, d])   ▷ If p' exists, it is p_cand
4:     p_stop ← STOP_π(π_{p..b}, σ_{q..d})              ▷ First non-visible point on π
5:     if p_cand < p_stop then return p_cand
6:     else return "No greedy step possible."  ▷ π_{p_cand} not reachable from π_p while at σ_q
```

3. **Maximal greedy step** $\textsc{MaxGreedyStep}_\pi(\pi', \sigma')$**.** Let $p' \in \mathrm{reach}_{\pi'}(p, q)$ be such that (i) p' is the largest index in $\mathrm{reach}_{\pi'}(p, q)$ with $|\mathrm{vis}_{\sigma'}(p', q)| = \max\{|\mathrm{vis}_{\sigma'}(z, q)| \mid z \in \mathrm{reach}_{\pi'}(p, q)\}$ and (ii) $\mathrm{vis}_{\sigma'}(p', q) \supseteq \mathrm{vis}_{\sigma'}(p, q)$. If p' exists, $\textsc{MaxGreedyStep}_\pi$ returns this value, otherwise it reports that no such index exists. Note that if p' exists, then by definition there is no greedy step on π starting from (p', q), i.e., this step is a maximal greedy step.

   While maximal greedy steps can be implemented by repeatedly computing minimal greedy steps until no further greedy step can be found, this does not suffice for our purposes. Such an approach introduces a dependence on the number of minimal greedy steps encountered, which might scale with the length of the curve. To achieve a polylogarithmic dependence on the number of vertices of π and σ, we instead separately implement $\textsc{MaxGreedyStep}_\pi$ as described in Algorithm 5.

Algorithm 5 Maximal greedy step

```
1: function MaxGreedyStep_π(π_{p..b}, σ_{q..d})
2:     q_min ← D^σ.minHeight([π_p − δ, ∞), [q, d])      ▷ Lowest still visible point on σ
3:     p_stop ← stop_π(π_{p..b}, σ_{q..d})              ▷ First non-visible point on π
4:     p_min ← D^π.minHeight((−∞, σ_{q_min} + δ], [p + 1, p_stop − 1])
                                  ▷ Maximizes visibility among reachable points
5:     if p_min = ∞ then          ▷ No reachable point has better visibility than π_p
6:         return "No greedy step possible."
7:     else
8:         q_min ← D^σ.minHeight([π_{p_min} − δ, ∞), [q, d])
                                  ▷ Lowest point on σ still seen by p_min
9:         return D^π.maxIndex((−∞, σ_{q_min} + δ], [p_min, p_stop − 1])
```

4. **Arbitrary greedy step** $\textsc{GreedyStep}_\pi(\pi', \sigma')$**.** If, in some situation, it is only required to find an arbitrary index $p' \in \text{reach}_{\pi'}(p, q)$ such that all $p \le i \le p'$ satisfy $\text{vis}_{\sigma'}(i, q) \subseteq \text{vis}_{\sigma'}(p', q)$ or report that no such index exists, we use the function $\textsc{GreedyStep}_\pi(\pi', \sigma')$ to denote that any such function suffices; in particular, $\textsc{MinGreedyStep}_\pi$ or $\textsc{MaxGreedyStep}_\pi$ can be used.

For σ, we define the obvious symmetric operations. Note that in these operations, it is not feasible to traverse all directly feasible points and check whether the visibility criterion is satisfied, since this would not necessarily yield a polylogarithmic running time.

Lemma 11.3.11. *Using $\mathcal{O}((n+m)\log nm)$ preprocessing time,* $\textsc{MaxGreedyStep}_\pi$*,* $\textsc{MinGreedyStep}_\pi$ *and* $\textsc{stop}_\pi$ *can be implemented to run in time $\mathcal{O}(\log nm)$.*

Proof. In time $\mathcal{O}((n+m)\log nm)$, we can build the data structure D^π for π and symmetrically D^σ for σ. Algorithms 3, 4 and 5 implement the greedy steps and $\textsc{stop}_\pi$ using only a constant number of queries to D^π and D^σ, each with running time $\mathcal{O}(\log n)$ or $\mathcal{O}(\log m)$. □

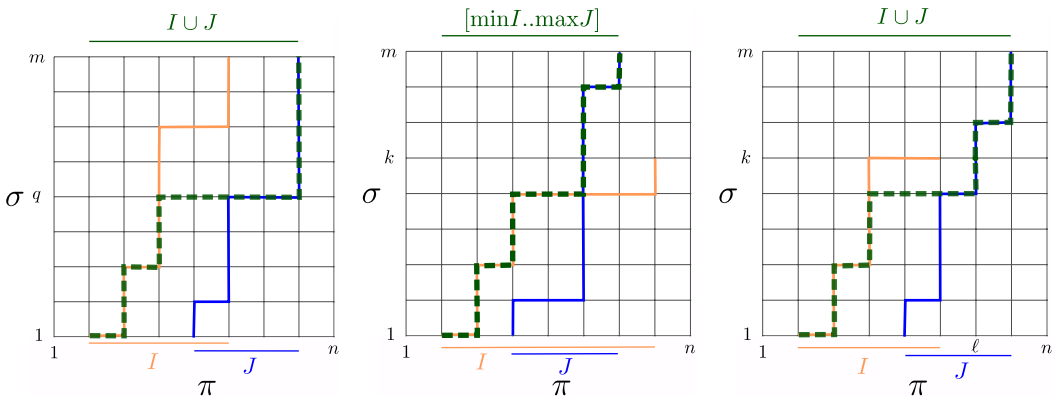
For the reduced free-space problem, these operations can be implemented even faster.

Lemma 11.3.12. *Let $\pi = \pi_{1..n}$ and $\sigma = \sigma_{1..m}$ be input curves of the reduced free-space problem. Using $\mathcal{O}((n+m)\log 1/\varepsilon)$ preprocessing time, the primitives* $\textsc{MaxGreedyStep}_\pi$*,* $\textsc{MinGreedyStep}_\pi$ *and* $\textsc{stop}_\pi$ *can be implemented to run in time $\mathcal{O}(\log 1/\varepsilon)$.*

Proof. We argue that range searching can be implemented with $\mathcal{O}(\log 1/\varepsilon)$ query time and $\mathcal{O}(n \log 1/\varepsilon)$ preprocessing time. This holds since for the point set $\mathcal{P} = \{(i, \pi_i) \mid i \in [1 \dots n]\}$ (1) the x-values are $1, \dots, n$, so that we can determine the relevant pointers in the first level of the fractional-cascading tree in constant time instead of $\mathcal{O}(\log n)$ and (2) all y-values are multiples of $\gamma = \frac{1}{4}\varepsilon\delta$ and in $[-2\delta, 2\delta]$, i.e., there are only $\mathcal{O}(1/\varepsilon)$ different y-values. For the latter, note that any point $\pi_p > \delta$ sees no point in σ, and this is preserved by setting π_p to 2δ (and similarly for σ). Using these properties it is straightforward to adapt the fractional-cascading data structure, we omit the details. □

11.3.3 Composition of One-Dimensional Curves

In this subsection, we collect essential composition properties of feasible traversals of one-dimensional curves that enable us to tackle the reduced free-space problem (see



(A) Lemma 11.3.13. (B) Lemma 11.3.14. (C) Lemma 11.3.15.

FIGURE 11.4: Composition properties of feasible traversals of one-dimensional separated curves.

Figure 11.4 for an illustration of these results). The first tool is a union lemma that states that two intersecting intervals I, J of π that each have a feasible traversal together with σ prove that also $\pi_{I\cup J}$ can be traversed together with σ.

Lemma 11.3.13. *Let $\pi = \pi_{1..n}$ and $\sigma = \sigma_{1..m}$ be one-dimensional separated curves and let $I, J \subseteq [1..n]$ be intervals with $I \cap J \neq \emptyset$. If $d_{\mathrm{dF}}(\pi_I, \sigma) \leq \delta$ and $d_{\mathrm{dF}}(\pi_J, \sigma) \leq \delta$, then $d_{\mathrm{dF}}(\pi_{I\cup J}, \sigma) \leq \delta$.*

Proof. If $I \subseteq J$, the claim is trivial. Without loss of generality, let $I = [a_I..b_I]$ and $J = [a_J..b_J]$, where $a_I \leq a_J \leq b_I \leq b_J$. Let ϕ_I (and ϕ_J) be a feasible traversal of (π_I, σ) (and (π_J, σ), respectively). By reparameterization, we can assume that $\phi_I(t) = (\psi_I(t), f(t))$ and $\phi_J(t) = (\psi_J(t), f(t))$ for suitable (non-decreasing onto) functions $\psi_I, \psi_J : [0,1] \to [1..n]$ and $f : [0,1] \to [1..m]$. One of the following cases occurs.

Case 1: There is some $0 \leq t \leq 1$ with $\psi_I(t) = \phi_J(t)$. Then we can concatenate $\phi_I(0,t)$ and $\phi_J(t,1)$ to obtain a feasible traversal of $\phi_{I\cup J}$.

Case 2: For all $0 \leq t \leq 1$, we have $\psi_I(t) < \psi_J(t)$. Let σ_q be the highest point on σ. By $d_{\mathrm{dF}}(\pi_I, \sigma) \leq \delta$ and $d_{\mathrm{dF}}(\pi_J, \sigma) \leq \delta$, the point σ_q sees all points on $\pi_{I\cup J}$. There is some $0 \leq t^* \leq 1$ with $f(t^*) = q$. We can concatenate $\phi_I(0, t^*)$ and the traversal of $\pi_{\psi_I(t^*)..\psi_J(t^*)}$ and σ_q to obtain a feasible traversal of $\pi_{a_I..\psi_I(t^*)}$ and $\sigma_{1..f(t^*)}$. Appending $\phi_J(t^*, 1)$ to this traversal yields $d_{\mathrm{dF}}(\pi_{a_I..b_J}, \sigma) \leq \delta$. □

The second result formalizes situations in which a traversal ϕ of subcurves has to cross a traversal ψ of other subcurves, yielding the possibility to follow ϕ up to the crossing point and to follow ψ from there on.

Lemma 11.3.14. *Let $\pi = \pi_{1..n}$ and $\sigma = \sigma_{1..m}$ be one-dimensional curves and consider intervals $I = [a_I..b_I]$ and $J = [a_J..b_J]$ with $J \subseteq I \subseteq [1..n]$, and $K = [1..k] \subseteq [1..m]$. If $d_{\mathrm{dF}}(\pi_I, \sigma_K) \leq \delta$ and $d_{\mathrm{dF}}(\pi_J, \sigma) \leq \delta$, then $d_{\mathrm{dF}}(\pi_{a_I..b_J}, \sigma) \leq \delta$.*

Proof. Let ϕ be a feasible traversal of π_I and σ_K and ψ a feasible traversal of π_J and σ. We first show that ϕ and ψ cross, i.e., there are $0 \leq t, t' \leq 1$ such that $\phi(t) = \psi(t')$. For all $k \in [1..K]$, let $[s_k^\phi..e_k^\phi]$ denote the interval of points that ϕ traverses on π while staying in σ_k. Similarly, $[s_k^\psi..e_k^\psi]$ denotes the interval of points ψ traverses on π while staying in σ_k. Assume for contradiction that $[s_k^\phi..e_k^\phi]$ and $[s_k^\psi..e_k^\psi]$ are disjoint for all $1 \leq k \leq K$. Then initially, we have $s_1^\phi = a_I \leq a_J = s_1^\psi$ and hence $e_1^\phi < s_1^\psi$. This implies

$s_2^\phi \leq e_1^\phi + 1 \leq s_1^\psi \leq s_2^\psi$ and inductively we obtain $e_k^\phi < s_k^\psi \leq e_k^\psi$ for all $k \in [1..K]$. This contradicts $e_K^\phi = b_I \geq b_J \geq e_K^\psi$. Hence, for some $1 \leq k \leq K$, $[s_k^\phi..e_k^\phi]$ and $[s_k^\psi..e_k^\psi]$ intersect, which gives $\phi(t) = (p,k) = \psi(t')$ for any $p \in [s_k^\phi, e_k^\phi] \cap [s_k^\psi, e_k^\psi]$ and the corresponding $0 \leq t, t' \leq 1$. By concatenating $\phi(0,t)$ with $\psi(t',1)$, we obtain a feasible traversal of $\pi_{a_I..b_J}$ and σ. □

The last result in our composition toolbox strengthens Lemma 11.3.13 to the case that the traversal of π_I uses only an initial subcurve $\sigma_{1..k}$ of σ and not the complete curve.

Lemma 11.3.15. *Let $\pi = \pi_{1..n}$ and $\sigma = \sigma_{1..m}$ be one-dimensional separated curves and consider intervals $I = [a_I..b_I]$ and $J = [a_J..b_J]$ with $1 \leq a_I \leq a_J \leq b_I \leq b_J \leq n$, and $K = [1..k] \subseteq [1..m]$. If $d_{\mathrm{dF}}(\pi_I, \sigma_K) \leq \delta$ and $d_{\mathrm{dF}}(\pi_J, \sigma) \leq \delta$, then $d_F(\pi_{I \cup J}, \sigma) \leq \delta$.*

Proof. Let ϕ be any feasible traversal of π_J and σ. There exists $a_J \leq \ell \leq b_J$ with $\phi(t) = (\ell, k)$ for some $0 \leq t \leq 1$. Hence ϕ restricted to $[0,t]$ yields a feasible traversal of $\pi_{a_J..\ell}$ and σ_K, i.e., $d_{\mathrm{dF}}(\pi_{a_J..\ell}, \sigma_K) \leq \delta$. Since I and $[a_J..\ell]$ are intersecting, Lemma 11.3.13 yields that $d_{\mathrm{dF}}(\pi_{a_I..\ell}, \sigma_K) \leq \delta$. Let ψ be such a feasible traversal of $\pi_{a_I..\ell}$ and σ_K. Concatenating ψ at $\psi(1) = (\ell, k) = \phi(t)$ with $\phi(t,1)$, we construct a feasible traversal of $\pi_{a_I..b_J}$ and σ, proving the claim. □

11.3.4 Solving the Reduced Free-Space Problem

In this section, we solve the reduced free-space problems, using the structural properties derived in the previous section and the principles underlying the greedy algorithm of Section 11.3.2. Recall that the greedy steps implemented as discussed in Section 11.3.2 run in time $\mathcal{O}(\log 1/\varepsilon)$ on the input curves of the reduced free-space problem.

Single Entry

Given the one-dimensional separated curves $\pi = (\pi_1, \ldots, \pi_n)$ and $\sigma = (\sigma_1, \ldots, \sigma_m)$ and entry set $E = \{1\}$, we show how to compute F^σ. We present the following recursive algorithm.

Algorithm 6 Special Case: Single entry

```
1:  function FIND-σ-EXITS(π_{p..b}, σ_{q..d})
2:      if q = d then
3:          if STOP_π(π_{p..b}, σ_q) = b + 1 then
4:              return {q}                ▷ The end of π is reachable while staying in σ_q
5:          else return ∅

6:      if p' ← MAXGREEDYSTEP_π(π_{p..b}, σ_{q..d}) then
7:          return FIND-σ-EXITS(π_{p'..b}, σ_{q..d})
8:      else if q' ← GREEDYSTEP_σ(π_{p..b}, σ_{q..d}) then
9:          return FIND-σ-EXITS(π_{p..b}, σ_{q..q'-1}) ∪ FIND-σ-EXITS(π_{p..b}, σ_{q'..d})
10:     else
11:         return FIND-σ-EXITS(π_{p..b}, σ_{q..d-1})        ▷ No greedy step possible
```

The following property establishes that a greedy step on a long curve is also a greedy step on a shorter curve. Clearly, the converse does not necessarily hold.

Proposition 11.3.16. *Let $1 \leq p \leq P \leq n$ and $1 \leq q \leq Q \leq m$. Any greedy step on π from (p,q) to (p',q) with $p' \leq P$ is also a greedy step with respect to $\tilde{\pi} := \pi_{p..P}$ and $\tilde{\sigma} := \sigma_{q..Q}$, i.e., if there is some $p' \leq P$ with $\mathrm{vis}_\sigma(i,q) \subseteq \mathrm{vis}_\sigma(p',q)$ for all $p \leq i \leq p'$, then also $\mathrm{vis}_{\tilde{\sigma}}(i,q) \subseteq \mathrm{vis}_{\tilde{\sigma}}(p',q)$.*

Proof. From the definition of vis_σ, we immediately derive $\mathrm{vis}_{\tilde{\sigma}}(i,q) = \mathrm{vis}_\sigma(i,q) \cap [q..Q] \subseteq \mathrm{vis}_\sigma(p',q) \cap [q..Q] = \mathrm{vis}_{\tilde{\sigma}}(p',q)$ for all $p \leq i \leq p'$. Restricting the length of π also has no influence on the greedy property, except for the trivial requirement that p' still has to be contained in the restricted curve. □

Lemma 11.3.17. *Algorithm 6 correctly identifies F^σ given the single entry $E = \{1\}$.*

Proof. Clearly, if FIND-σ-EXITS(π,σ) finds and returns an exit e on σ, then it is contained in F^σ, since the algorithm uses only feasible (greedy) steps. Conversely, we show that for all $I = [p..b]$ and $J = [q..d]$, where (p,q) is a greedy point pair of π and σ, and all $e \in J$ with $d_{\mathrm{dF}}(\pi_I, \sigma_{J\cap[1..e]}) \leq \delta$, we have $e \in$ FIND-σ-EXITS(π_I, σ_J), i.e. we find all exits.

Consider some call of FIND-σ-EXITS(π_I, σ_J) for which the precondition is fulfilled. If J consists only of a single point, then $J = \{e\}$, and a feasible traversal of π_I and σ_J exists if and only if σ_e sees all points on π_I. Let $I = [p..b]$, then this happens if and only if $\mathrm{STOP}_\pi(\pi_I, \sigma_e) = b + 1$, hence the base case is treated correctly.

Assume that $I = [p..b]$ and a maximal greedy step p' on π exists. By Property 11.3.16, this step is a greedy step also with respect to $\sigma_{J\cap[1..e]}$. Hence by Lemma 11.3.9, if there is a traversal of $\pi_{p..b}$ and $\sigma_{J\cap[1..e]}$, then a traversal of $\pi_{[p'..b]}$ and $\sigma_{J\cap[1..e]}$ also exists.

Consider the case in which $J = [q..d]$ and a greedy step q' in σ exists. If $e < q'$, then $e \in [q..q'-1]$ and $J \cap [1..e] = [q..q'-1] \cap [1..e]$. Hence, e is found in the recursive call with $J' = [q..q'-1]$. If $e \geq q'$, then by Property 11.3.16, this step is a greedy step with respect to the curves π_I and $\sigma_{J\cap[1..e]}$. Again, by Lemma 11.3.9, the existence of a feasible traversal of π_I and σ_J implies that also a feasible traversal of π_I and $\sigma_{J\cap[q'..e]}$ exists.

It remains to regard the case in which no greedy step exists. By Lemma 11.3.9, there is no feasible traversal of $\pi_{1..n}$ and $\sigma_{1..d}$. This implies $e \neq d$ and all exits are found in the recursive call with $J' = [q, d-1]$. □

Lemma 11.3.18. FIND-σ-EXITS$(\pi_{p..b}, \sigma_{q..d})$ *runs in time $\mathcal{O}((d-q+1)\cdot \log 1/\varepsilon)$.*

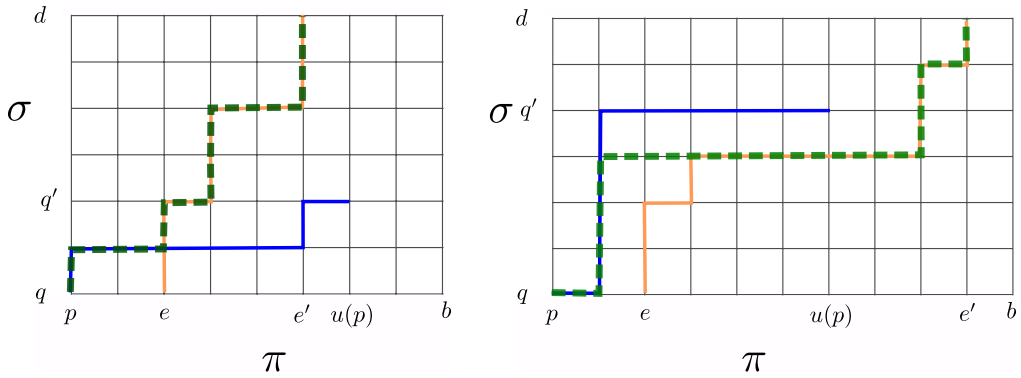
Proof. Since the algorithm's greedy steps on π are maximal, after each greedy step on π, we split σ (by a greedy step on σ) or shorten σ (if no greedy step on σ is found). Thus, it takes at most $\mathcal{O}(\log 1/\varepsilon)$ time until σ is split or shortened. The base case is also handled in time $\mathcal{O}(\log 1/\varepsilon)$. In total, this yields a running time of $\mathcal{O}((d-q+1)\log 1/\varepsilon)$. □

Note that by swapping the roles of π and σ, FIND-σ-EXITS can be used to determine F^π given the single entry σ_1 on σ. This is equivalent to having the single entry $E = \{1\}$ on π. Thus, we can also implement the function FIND-π-EXITS$(\pi_{1..n}, \sigma_{1..m})$ that returns F^π given the single entry $E = \{1\}$ on π in time $\mathcal{O}(n \log 1/\varepsilon)$.

Entries on π, Exits on π

In this section, we tackle the task of determining F^π given a set of entries E on π. It is essential to avoid computing the exits by iterating over every single entry. We show how to divide π into disjoint subcurves that can be solved by a single call to FIND-π-EXITS each.

Assume we want to traverse $\pi_{p..b}$ and $\sigma_{q..d}$ starting in π_p and σ_q. Let $u(p) := \max\{p' \in [p,b] \mid \exists q \leq q' \leq d : d_{\mathrm{dF}}(\pi_{p..p'}, \sigma_{q..q'}) \leq \delta\}$ be the last point on π that is reachable while



(A) Statement 1: A traversal of $\pi_{e..e'}$ and $\sigma_{q..d}$ has to cross a traversal of $\pi_{p..u(p)}$ and $\sigma_{q..q'}$, which we can combine by Lemma 11.3.14.

(B) Statement 2: If there was a traversal of $\pi_{e..e'}$ and $\sigma_{q..d}$, then by Lemma 11.3.15, a traversal of $\pi_{p..e'}$ and $\sigma_{q..d}$ exists, contradicting the definition of $u(p)$.

FIGURE 11.5: An illustration of the proof of Lemma 11.3.19.

traversing any subcurve of $\sigma_{q..d}$ starting in σ_q. This point fulfills the following properties (whose proof ideas are depicted in Figure 11.5).

Lemma 11.3.19. *It holds that*

1. *If there are $p \leq e \leq e' \leq u(p)$ with $d_{\mathrm{dF}}(\pi_{e..e'}, \sigma_{q..d}) \leq \delta$, then $d_{\mathrm{dF}}(\pi_{p..e'}, \sigma_{q..d}) \leq \delta$.*
2. *For all $p \leq e \leq u(p) < e'$, we have that $d_{\mathrm{dF}}(\pi_{e..e'}, \sigma_{q..d}) > \delta$.*

Proof. By definition of $u(p)$, there is a $q \leq q' \leq d$ with $d_{\mathrm{dF}}(\pi_{p..u(p)}, \sigma_{q..q'}) \leq \delta$. Since $[e, e'] \subseteq [p, u(p)]$, Lemma 11.3.14 proves the first statement. For the second statement, assume for contradiction that $d_{\mathrm{dF}}(\pi_{e..e'}, \sigma_{q..d}) \leq \delta$. Then, Lemma 11.3.15 yields that $d_{\mathrm{dF}}(\pi_{p..e'}, \sigma_{q..d}) \leq \delta$. This is a contradiction to the choice of $u(p)$, since $e' > u(p)$. □

The above lemma implies that we can ignore all entries in $[p..u(p)]$ except for p. Moreover, all exits reachable from p are contained in the interval $[p..u(p)]$. This gives rise to the following algorithm.

Algorithm 7 Given entry points E on π, compute all exits on π.

```
1:  function π-EXITS-FROM-π(π, σ, E)
2:      S ← ∅
3:      while E ≠ ∅ do
4:          p̂ ← pop minimal index from E
5:          p ← p̂, q ← 1
6:          repeat
7:              if q′ ← MAXGREEDYSTEP_σ(π_{p..n}, σ_{q..m}) then
8:                  q ← q′
9:              if p′ ← GREEDYSTEP_π(π_{p..n}, σ_{q..m}) then
10:                 p ← p′
11:         until no greedy step was found in the last iteration
12:         p̄ ← STOP_π(π_{p..n}, σ_{q..m}) − 1   ▷ determines the maximal reachable point u(p̂)
13:         S ← S ∪ FIND-π-EXITS(π_{p̂..p̄}, σ)
14:         E ← E ∩ [p̄ + 1, n]                  ▷ drops all entries in [p̂, u(p̂)]
15:     return S
```

Lemma 11.3.20. *Algorithm 7 correctly computes F^π.*

Proof. We first argue that for each considered entry $\hat{p}$, the algorithm computes $\bar{p} = u(\hat{p})$. Clearly, $\bar{p} \leq u(\hat{p})$, since only feasible steps are used to reach $\bar{p}$. If $\bar{p} = m$, this already implies that also $u(\hat{p}) = m$. Otherwise, let (p, q) be the greedy point pair on the curves $\pi_{\hat{p}..n}$ and σ for which no greedy step has been found. Then by Lemma 11.3.9, for $p_{\text{stop}} := \text{STOP}_\pi(\pi_{p..n}, \sigma_{q..m})$ and all $1 \leq q' \leq m$, we have that $d_{\text{dF}}(\pi_{\hat{p}..p_{\text{stop}}}, \sigma_{1..q'}) > \delta$. Hence, $u(\hat{p}) < p_{\text{stop}}$. Finally, note that Algorithm 7 computes $\bar{p} = p_{\text{stop}} - 1$, which proves $\bar{p} = u(\hat{p})$.

It is clear that every found exit is included in F^π. Conversely, let $e' \in F^\pi$ and $1 \leq e \leq n$ be such that $d_{\text{dF}}(\pi_{e..e'}, \sigma) \leq \delta$. For some $\hat{p}$ with $\hat{p} \leq e \leq u(\hat{p}) = \bar{p}$, we run FIND-$\pi$-EXITS$(\pi_{\hat{p}..\bar{p}}, \sigma)$. Hence by Lemma 11.3.19 (2), $e' \leq u(\hat{p})$ and by Lemma 11.3.19 (1), $d_{\text{dF}}(\pi_{\hat{p}..e'}, \sigma) \leq \delta$. Hence, the corresponding call FIND-π-EXITS$(\pi_{\hat{p}..\bar{p}}, \sigma)$ will find e'. □

Lemma 11.3.21. *Using preprocessing time $\mathcal{O}((n + m) \log 1/\varepsilon)$, Algorithm 7 runs in time $\mathcal{O}(n \log 1/\varepsilon)$.*

Proof. We first bound the cost of all calls FIND-π-EXITS(π_{I_i}, σ). Clearly, all intervals I_i are disjoint with $\bigcup I_i \subseteq [1..n]$. Hence, by Lemma 11.3.18, the total time spent in these calls is bounded by $\mathcal{O}(\sum_i |I_i| \log(1/\varepsilon)) = \mathcal{O}(n \log 1/\varepsilon)$. To bound the number greedy steps, let $p_1, \ldots, p_k$ be the distinct indices considered as values of p during the execution of π-EXITS-FROM-$\pi(\pi, \sigma)$. Between changing p from each p_i to p_{i+1}, we will make, by maximality, at most one call to MAXGREEDYSTEP$_\sigma$ and at most one call to GREEDYSTEP$_\pi$. Since $k \leq n$, the total cost of greedy calls is bounded by $\mathcal{O}(n \log 1/\varepsilon)$ as well. The total time spent in all other operations is bounded by $\mathcal{O}(n \log 1/\varepsilon)$. □

Entries on π, Exits on σ

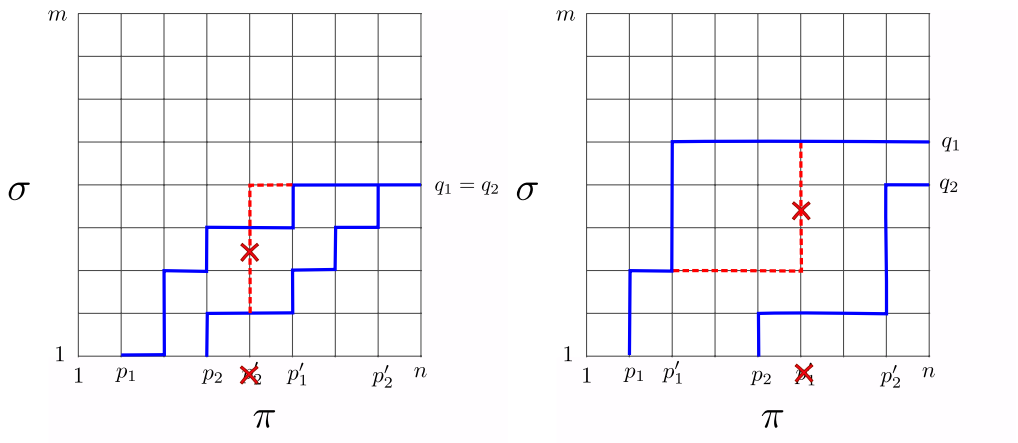
Similar to the previous section, we show how to compute the exits F^σ given entries E on π, by reducing the problem to calls of FIND-σ-EXITS on subcurves of π and σ. This time, however, the task is more intricate. For any index p on π, let $Q(p) := \min\{q \mid d_{\text{dF}}(\pi_{p..n}, \sigma_{1..q}) \leq \delta\}$ be the endpoint of the shortest initial fragment of σ such that the remaining part of π can be traversed together with this fragment[3]. Let $P(p) := \min\{p' \mid d_{\text{dF}}(\pi_{p..p'}, \sigma_{1..Q(p)}) \leq \delta\}$ be the endpoint of the shortest initial fragment of π, such that $\sigma_{Q(p)}$ can be reached by a feasible traversal.

Note that by definition, entries p with $Q(p) = \infty$ are irrelevant for determining the exits on σ. In fact, if an entry p is *relevant*, i.e., $Q(p) < \infty$, it is easy to compute $Q(p)$ due to the following lemma.

Lemma 11.3.22. *Let $Q'(p) := \min\{q \mid \sigma_q \geq \max_{i \in [p..n]} \pi_i - \delta\}$. If $Q(p) < \infty$, then $Q(p) = Q'(p)$. Similarly, $Q(p) < \infty$ implies that $P(p) = \min\{p' \mid \pi_{p'} \leq \min_{i \in [q..Q(p)]} \sigma_i + \delta\} < \infty$.*

Proof. Assume that $Q(p) < Q'(p)$ holds, then no point in $\sigma_{1..Q(p)}$ sees the highest point in $\pi_{p..n}$. Hence no feasible traversal of these curves can exist, yielding a contradiction. Assume that $Q(p) > Q'(p)$ holds instead and consider the feasible traversal ϕ of the shortest initial fragment of σ that passes through all points in $\pi_{p..n}$. At some point ϕ visits $(\pi_{p'}, \sigma_{Q'(p)})$ for some $p \leq p' \leq n$. We can alter this traversal to pass through the remaining curve $\pi_{p'..n}$ while staying in $\sigma_{Q'(p)}$, since $\sigma_{Q'(p)}$ sees all points on $\pi_{p'..n}$. This

[3] As a convention, we use $\min \emptyset = \infty$.



(A) Case $q_1 = q_2$. (B) Case $q_1 > q_2$.

FIGURE 11.6: Illustration of Lemma 11.3.23. For both p_i, $i \in \{1, 2\}$, a feasible traversal of the curves $\pi_{p_i..p'_i}$ and $\sigma_{1..q_i}$ is depicted as monotone paths in the free-space.

gives a feasible traversal of $\pi_{p..n}$ and $\sigma_{1..Q'(p)}$, which is a contradiction to the choice of ϕ and $Q(p) > Q'(p)$.

The statement for $P(p)$ follows analogously by regarding the curves $\pi_{p..n}$ and $\sigma_{1..Q(p)}$ and switching their roles. □

Note that the previous lemma shows that for relevant entries $p_1 < p_2$, we have $Q(p_1) \geq Q(p_2)$, since for relevant entries, $Q(p_1) = Q'(p_1) \geq Q'(p_2) = Q(p_2)$. We will use the following lemma to argue that (i) if $Q(p_1) = Q(p_2)$, entry p_1 dominates p_2, and (2) if $Q(p_1) > Q(p_2)$, we have $p_2 \notin [p_1..P(p_1)]$. Hence, we can ignore all entries in $[p_1..P(p_1)]$ except for p_1 itself.

Lemma 11.3.23. *Let $p_1 < p_2$ be indices on π with $q_1 := Q(p_1) < \infty$ and $q_2 := Q(p_2) < \infty$. Let $p'_1 := P(p_1)$ and $p'_2 := P(p_2)$. If $q_1 = q_2$, then $p'_1 \leq p'_2$. Otherwise, i.e., if $q_1 > q_2$, we even have $p'_1 < p_2$.*

Proof. See Figure 11.6 for illustrations. Let $q_1 = q_2$. Assume for contradiction that $p'_1 > p'_2$, then we have $d_{\mathrm{dF}}(\pi_{p_1..p'_1}, \sigma_{1..q_1}) \leq \delta$ and $d_{\mathrm{dF}}(\pi_{p_2..p'_2}, \sigma_{1..q_1}) \leq \delta$, where $[p_2..p'_2] \subseteq [p_1..p'_1]$. Hence by Lemma 11.3.14, $d_{\mathrm{dF}}(\pi_{p_1..p'_2}, \sigma_{1..q_1}) \leq \delta$ and thus $p'_1 \leq p'_2$, which is a contradiction to the assumption.

For the second statement, let p be maximal such that $\pi_p > \sigma_{q_2} + \delta$. If p does not exist or $p < p_1$, we have that $Q'(p_1) = Q'(p_2)$ and hence by Lemma 11.3.22, $q_1 = q_2$. Note that additionally $p < p_2$, since otherwise $\sigma_{q_2} < \pi_p - \delta$ with $p \geq p_2$ shows that $q_2 \neq Q'(p_2)$ contradicting Lemma 11.3.22. Thus, in what follows, we can assume that $p_1 < p < p_2$.

Assume for contradiction that $q_1 > q_2$ and $p'_1 \geq p_2$. Then a feasible traversal ϕ of $\pi_{p_1..p'_1}$ and $\sigma_{1..q_1}$ visits (π_p, σ_q) for some $1 \leq q \leq q_1$. It even holds that $q < q_1$, since otherwise there is a feasible traversal of $\sigma_{1..q_1}$ and $\pi_{p_1..p}$ with $p < p'_1$, contradicting the choice of p'_1. Clearly, $\sigma_q > \sigma_{q_2}$, since π_p sees σ_q, while it does not see σ_{q_2}. Since by choice of p, σ_{q_2} sees all of $\pi_{p+1..n}$ and σ_q sees only more (including π_p), we conclude that we can traverse all points of $\pi_{p..n}$ while staying in σ_q. Concatenating this traversal to the feasible traversal ϕ yields $d_{\mathrm{dF}}(\pi_{p_1..n}, \sigma_{1..q}) \leq \delta$ and thus $Q(p_1) \leq q < q_1$, which is a contradiction to Lemma 11.3.22. This proves that $q_1 > q_2$ implies $p'_1 < p_2$. □

Lemma 11.3.24. *Algorithm 8 fulfills the following properties.*

Algorithm 8 Given entry points E on π, compute all exits on σ.

```
1: function σ-EXITS-FROM-π(π, σ, E)
2:     F ← ∅, q̄ ← m
3:     repeat
4:         p̂ ← pop minimal index from E
5:         p ← p̂, q ← 1
6:         Q′ ← Q′(p)
7:         repeat
8:             if q′ ← MaxGreedyStep_σ(π_{p..n}, σ_{q..Q′}) then
9:                 q ← q′
10:            if q ≠ Q′ and p′ ← MinGreedyStep_π(π_{p..n}, σ_{q..Q′}) then
11:                p ← p′
12:        until q = Q′ or no greedy step was found in the last iteration
13:        if q = Q′ then
14:            F ← F ∪ Find-σ-Exits(π_{p..n}, σ_{Q′..q̄})
15:            q̄ ← Q′ − 1
16:        E ← E ∩ [p + 1, n]
17:    until E = ∅
18:    return F
```

1. *Let (p, q) with $q < Q'(\hat{p})$ be a greedy point pair of $\pi_{\hat{p}..n}$ and $\sigma_{1..Q'(\hat{p})}$ for which no greedy step exists. Then for all $e \in [\hat{p}, p]$, we have $Q(e) = \infty$.*

2. *For each considered $\hat{p}$, if we have $Q(\hat{p}) < \infty$, then the algorithm makes a recursive call to* Find-σ-Exits$(\pi_{P(\hat{p})..n}, \sigma_{Q(\hat{p})..\bar{q}})$. *In this case, the point $(P(\hat{p}), Q(\hat{p}))$ is a greedy pair of $\pi_{\hat{p}..n}$ and σ.*

Proof. For the first statement, assume for contradiction $Q(e) < \infty$. By Lemma 11.3.22, $Q(e) = Q'(e)$, which implies that for all $q' < Q'(e) \leq Q'(\hat{p})$, we have $\sigma_{q'} < \sigma_{Q'(e)}$ and hence $\mathrm{vis}_\pi(p, q') \subseteq \mathrm{vis}_\pi(p, Q'(e))$. Assume first that $q < Q'(e)$. In this case, $\mathrm{stop}_\sigma(\pi_{p..n}, \sigma_{q..Q'(\hat{p})}) \leq Q'(e)$, since otherwise $Q'(e) \leftarrow \mathrm{GreedyStep}_\sigma(\pi_{p..n}, \sigma_{q..Q'(\hat{p})})$. By Lemma 11.3.9, this proves that $d_{\mathrm{dF}}(\pi_{\hat{p}..n}, \sigma_{1..Q'(e)}) > \delta$. Since $d_{\mathrm{dF}}(\pi_{\hat{p}..e}, \sigma_{1..q'}) \leq \delta$ for some $q' < Q'(e)$, Lemma 11.3.15 yields $d_{\mathrm{dF}}(\pi_{e..n}, \sigma_{1..Q'(e)}) > \delta$. This is a contradiction to $Q(e) = Q'(e)$.

Otherwise, if $q \geq Q'(e)$, then consider $1 \leq q^* \leq q$ maximizing σ_{q^*}. In particular, we have $\sigma_{q^*} \geq \pi_{p'} - \delta$ for any $\hat{p} \leq p' \leq p$, since we constructed a traversal of $\pi_{\hat{p}..p}, \sigma_{1..q}$. Moreover, $\sigma_{q^*} \geq \sigma_{Q'(e)}$, since $1 \leq Q'(e) \leq q$. Since $\sigma_{Q'(e)} \geq \pi_{p'} - \delta$ for all $e \leq p' \leq n$ and $e \leq p$, we have $\sigma_{q^*} \geq \pi_{p'} - \delta$ for all $\hat{p} \leq p' \leq n$. This contradicts $q^* \leq q < Q'(\hat{p})$.

For the second statement, note that if $Q(\hat{p}) < \infty$, then by Lemma 11.3.22, $Q(\hat{p}) = Q'(\hat{p})$. Hence Lemma 11.3.9 yields that the algorithm finds a feasible traversal of $\pi_{\hat{p}..p}$ and $\sigma_{1..Q'(\hat{p})}$ for some $\hat{p} \leq p \leq n$. This shows that $P(\hat{p}) \leq p < \infty$. Let $\sigma' := \sigma_{1..Q(\hat{p})}$ and assume that there is a $p' < p$ with $d_{\mathrm{dF}}(\pi_{\hat{p}..p'}, \sigma') \leq \delta$ and let $(\tilde{p}, \tilde{q})$ be the greedy point of $\pi_{\hat{p}..n}$ and σ' right before the algorithm made a greedy step on π to some index in $(p', p]$. By maximality of the greedy steps on σ, there exists $\tilde{q} < q_{\min} < Q(\hat{p})$ such that $\pi_{\tilde{p}}$ does not see $\sigma_{q_{\min}}$, since otherwise $Q(\hat{p}) \in \mathrm{reach}_{\sigma'}(\tilde{p}, \tilde{q})$ with $\mathrm{vis}_\pi(\tilde{p}, \tilde{q}) \subsetneq \mathrm{vis}_\pi(\tilde{p}, Q(\hat{p}))$, i.e., $Q(\hat{p})$ would be a greedy step on σ'. By minimality of greedy steps on π, $\mathrm{vis}_{\sigma'}(\tilde{p}, \tilde{q}) \supsetneq \mathrm{vis}_{\sigma'}(i, \tilde{q})$ for all $\tilde{p} \leq i \leq p'$. Hence, no vertex on $\pi_{\tilde{p}..p'}$ sees $\sigma_{q_{\min}}$, which proves $d_{\mathrm{dF}}(\pi_{\tilde{p}..p'}, \sigma'_{\tilde{q}..Q(\hat{p})}) > \delta$. Since $(\tilde{p}, \tilde{q})$ is a greedy pair of $\pi_{\hat{p}..p'}$ and σ', this yields that $d_{\mathrm{dF}}(\pi_{\hat{p}..p'}, \sigma') > \delta$ by

Lemma 11.3.9, which is a contradiction to the assumption. Hence, the algorithm calls FIND-σ-EXITS$(\pi_{p..n}, \sigma_{Q'(\hat{p})..\bar{q}})$, where $p = P(\hat{p})$ and $Q'(\hat{p}) = Q(\hat{p})$.

It remains to show that $(P(\hat{p}), Q(\hat{p}))$ is also a greedy pair of $\pi_{\hat{p}..n}$ and the *complete* curve σ. By Lemma 11.3.22, every $\hat{p} \leq p < P(\hat{p})$ satisfies $\pi_p > \pi_{P(\hat{p})}$ and hence $\mathrm{vis}_\sigma(p, q) \subseteq \mathrm{vis}_\sigma(P(\hat{p}), q)$ for all $1 \leq q \leq m$. Hence, if at some greedy pair (p, q), $q \leq Q(\hat{p})$, a greedy step $p' \leftarrow \text{GREEDYSTEP}_\pi(\pi_{p..n}, \sigma)$ with $p' \geq P(\hat{p})$ exists, then also $P(\hat{p}) \leftarrow \text{GREEDYSTEP}_\pi(\pi_{p..n}, \sigma)$, which shows that $(P(\hat{p}), q)$ is a greedy point of $\pi_{\hat{p}..n}$ and σ. If $q = Q(\hat{p})$, then $(P(\hat{p}), Q(\hat{p}))$ is a greedy point pair. Otherwise, by Lemma 11.3.22, $P(\hat{p})$ sees all of $\sigma_{q..Q(\hat{p})}$ and $\sigma_q < \sigma_{Q(\hat{p})}$, hence $Q(\hat{p}) \in \text{GREEDYSTEP}_\sigma(\pi_{P(p)..n}, \sigma)$ and $(P(\hat{p}), Q(\hat{p}))$ is a greedy step of $\pi_{\hat{p}..n}$ and σ.

It is left to consider the case that for all greedy pairs (p, q), $q \leq Q(\hat{p})$, of $\pi_{\hat{p}..n}$ and σ, no greedy step to some $p' \geq P(\hat{p})$ exists. Then there is some (p, q) with $p < P(\hat{p})$ and $q \leq Q(\hat{p})$ for which no greedy step exists at all. We have $p_{\text{stop}} := \text{STOP}_\pi(\pi_{p..n}, \sigma_{q..m}) \leq P(\hat{p})$, since otherwise $P(\hat{p})$ would be a greedy step. Since Lemma 11.3.9 shows that $d_{\mathrm{dF}}(\pi_{\hat{p}..p_{\text{stop}}}, \sigma_{1..q'}) > \delta$ for any q', this contradicts $d_{\mathrm{dF}}(\pi_{\hat{p}..P(\hat{p})}, \sigma_{1..Q(\hat{p})}) \leq \delta$. □

Lemma 11.3.25. *Algorithm 8 correctly computes F^σ.*

Proof. Clearly, any exit found is contained in F^σ, since the methods σ-EXITS-FROM-π and FIND-σ-EXITS only use feasible steps. For the converse, let $e \in E$ be an arbitrary entry and consider the set $F^\sigma_e = \{q \mid d_{\mathrm{dF}}(\pi_{e..n}, \sigma_{1..q}) \leq \delta\}$ of σ-exits corresponding to the entry e.

As a first step, we show that if $F^\sigma_e \neq \emptyset$ and hence $Q(e) < \infty$, then we have that $F^\sigma_e = \text{FIND-}\sigma\text{-EXITS}(\pi_{P(e)..n}, \sigma_{Q(e)..m})$. Let $\bar{e} \in F^\sigma_e$. By Lemma 11.3.24, $(P(e), Q(e))$ is a greedy pair of $\pi_{e..n}$ and σ and hence also of $\pi_{e..n}$ and $\sigma_{1..\bar{e}}$. Lemma 11.3.9 thus implies $d_{\mathrm{dF}}(\pi_{P(e)..n}, \sigma_{Q(e)..\bar{e}}) \leq \delta$ and consequently $\bar{e} \in \text{FIND-}\sigma\text{-EXITS}(\pi_{P(e)..n}, \sigma_{Q(e)..m})$. The converse clearly holds as well.

Note that e is not considered as $\hat{p}$ in any iteration of the algorithm if and only if the algorithm considers some $\hat{p}$ with $e \in [\hat{p} + 1..p]$, where either (i) the algorithm finds a greedy pair (p, q) of $\pi_{\hat{p}..n}$ and $\sigma_{1..Q'(\hat{p})}$ that allows no further greedy steps, or (ii) the algorithm calls FIND-σ-EXITS$(\pi_{p..n}, \sigma_{Q'(\hat{p})..\bar{q}})$, where $p = P(\hat{p})$ by Lemma 11.3.24. In the first case, $F^\sigma_e = \emptyset$ since Lemma 11.3.24 proves $Q(e) = \infty$. In the second case, if $F^\sigma_e \neq \emptyset$, we have $Q(e) < \infty$, and hence by Lemma 11.3.23, $Q(e) = Q(\hat{p})$ and $P(\hat{p}) \leq P(e)$. Since $\sigma_{Q(\hat{p})}$ sees all of $\pi_{P(\hat{p})..n}$, any exit reachable from $(P(e), Q(e))$ is reachable from $(P(\hat{p}), Q(\hat{p}))$ as well. Hence $F^\sigma_e \subseteq F^\sigma_{\hat{p}}$.

Let $\hat{p}_1 \leq .. \leq \hat{p}_k$ be the entries considered as $\hat{p}$ by the algorithm. It remains to show that the algorithm finds all exits $\bigcup_{i=1}^k F^\sigma_{\hat{p}_i}$. We inductively show that the algorithm computes $F^\sigma_{\hat{p}_i} \setminus \bigcup_{j<i} F^\sigma_{\hat{p}_j}$ in the loop corresponding to $\hat{p} = \hat{p}_i$. For any i, let $k := \max\{j < i \mid Q(\hat{p}_j) < \infty\}$. The base case is when k is undefined, then the claim is immediate. Otherwise, note that the corresponding loop computes

$$\text{FIND-}\sigma\text{-EXITS}(\pi_{P(\hat{p}_i)..n}, \sigma_{Q(\hat{p}_i)..Q(\hat{p}_k)-1}) = F^\sigma_{\hat{p}_i} \cap [Q(\hat{p}_i)..Q(\hat{p}_k) - 1].$$

The claim follows if we can show $F^\sigma_{\hat{p}_i} \cap [Q(\hat{p}_k)..m] \subseteq F^\sigma_{\hat{p}_k}$. Let $\bar{e} \in F^\sigma_{\hat{p}_i}$ with $\bar{e} \geq Q(\hat{p}_k)$. Then $d_{\mathrm{dF}}(\pi_{\hat{p}_i..n}, \sigma_{1..\bar{e}}) \leq \delta$. Together with $d_{\mathrm{dF}}(\pi_{\hat{p}_k..n}, \sigma_{1..Q(\hat{p}_k)}) \leq \delta$, Lemma 11.3.14 shows that $d_{\mathrm{dF}}(\pi_{\hat{p}_k..n}, \sigma_{1..\bar{e}}) \leq \delta$ and hence $e \in F^\sigma_{\hat{p}_k}$.

□

Lemma 11.3.26. *Algorithm 8 runs in time $\mathcal{O}((n + m) \log 1/\varepsilon)$.*

Proof. Consider the total cost of the calls FIND-σ-EXITS$(\pi_{I_i}, \sigma_{J_i})$. Since all J_i are disjoint and $\bigcup_i J_i \subseteq [1..m]$, Lemma 11.3.18 bounds the total cost of such calls by

$\mathcal{O}(\sum_i |J_i| \log(1/\varepsilon)) = \mathcal{O}(m \log(1/\varepsilon))$. Let $p_1, \ldots, p_k$ denote the distinct indices considered as p during the execution of the algorithm. Between changing p_i to p_{i+1}, we will make at most one call to MaxGreedyStep$_\sigma$ (by maximality) and at most once call to MinGreedyStep$_\pi$. Hence $k \leq n$ bounds the number of calls to greedy steps by $\mathcal{O}(n \log(1/\varepsilon))$. □

11.4 Conclusion

We presented an improved $(1+\varepsilon)$-approximation algorithm for the Fréchet distance on c-packed curves running in time $\tilde{\mathcal{O}}(cn/\sqrt{\varepsilon})$. While our running time improves the state of the art for $\varepsilon \ll 1/\log n$, we suspect that our algorithm is too complex to speed up Fréchet distance computation in practical situations, unless ε is very small. Our running time matches a conditional lower bound, so that it is asymptotically optimal, up to lower order factors of the form $n^{o(1)}$, in dimension $d \geq 5$, and unless the Strong Exponential Time Hypothesis fails. We leave it as open problems to (1) find simpler and more practical algorithms with the same asymptotic guarantees as ours, (2) improve our $\log n$ and $\log 1/\varepsilon$-factors, and (3) determine the correct asymptotic behaviour in dimension $d = 2, 3, 4$.

Part III

Broadcasting

Chapter 12

Introduction to Part III

In the final part of this thesis, we consider two quite different, yet fundamental aspects of (randomized) broadcasting in communication networks. Our setting is that of a single person, Alice, who owns some piece of information and wishes to disseminate this piece of information to all of her friends (or even the public). Chapter 13 is concerned with how *quickly* Alice can spread the information to her friends when only direct contacts among all friends are allowed, while the aim in Chapter 14 is to broadcast the information *anonymously*, i.e., without revealing the identity of Alice.

For the first task, called *rumor spreading*, a wealth of communication protocols has been suggested and analyzed, starting with a simple communication protocol that proceeds in a round-based fashion as follows: in each round, each participant knowing the rumor randomly contacts a neighbor and informs him/her of the rumor if it was unknown to him/her. This *(synchronized) push protocol* is probably the most basic and classic randomized rumor spreading protocol, with a wide range of applications in algorithms, distributed systems, simulation and modeling of epidemic processes, etc. As such, we feel that it deserves a detailed look at its performance. In Chapter 13, we give a precise analysis of the distribution of the time it takes this protocol to inform all n nodes in a complete graph, improving upon classical results [FG85; Pit87].

The second task, i.e., anonymous information disclosure in a public network, has recently been formalized as the *cryptogenography* problem [Bro+14]: Here, the owner of the piece of information (this time called *secret*), Alice, is chosen uniformly at random from a set of n cooperative players. The players' aim is to communicate publicly (without use of any privately shared information) such that the *success probability* SUCC is maximized, i.e., the probability that (i) the correct secret is derivable from all communication and (ii) an outside observer trying to identify Alice fails in guessing her identity. In Chapter 14, we consider the simplest possible special case, in which two players cooperate to disclose a single secret bit. In this case, Brody et al. [Bro+14], using deep structural insights, improved the trivial bounds of $\text{SUCC} \in [1/4, 1/2]$ to $\text{SUCC} \in [1/3, 3/8]$. Since their protocol obtaining a success probability of $1/3$ is surprising at first sight and – when viewed in a suitable formulation of the problem – natural and elegant at a closer look, one might suspect it to be optimal. We disprove this hope by presenting protocols with success probability strictly larger than $1/3$ and additionally give a stronger hardness proof, yielding the tighter bounds $\text{SUCC} \in [0.3384, 0.3672]$.

We give more detailed introductions to Chapters 13 and 14 below.

12.1 Rumor Spreading in Complete Graphs

Randomized rumor spreading is a class of randomized processes with ample applications in algorithmics, but also in modeling natural or technical spreading processes (e.g., epidemics and computer viruses). In Chapter 13, we shall give a very precise analysis of

the most basic rumor spreading process in which a single piece of information is spread in a group of n people by, in a round-based fashion, each informed person calling a random one and gossiping the rumor to him/her.

Randomized Rumor Spreading Processes. A randomized rumor spreading process is characterized by the fact that a rumor is spread in a network by nodes of the network exchanging information with randomly chosen neighbors. Such processes and similar ones have been studied in mathematical epidemiology and stochastic particle systems, see, e.g., [Lig99]. In computer science, besides modeling epidemic processes with relevance to computer science (e.g., spread of information in social networks [DFF12], spread of computer viruses [Ber+05] and forming of opinions in social networks [Kle08]), rumor spreading is an important algorithmic paradigm.

While its very first occurrence [FG85] was only as an analysis tool for algorithmic problems with no immediate connection to rumor spreading, it quickly was noted that randomized rumor spreading can be used as a highly scalable and robust mechanism to distribute updates in replicated database applications [Dem+88]. This scalability today is mostly exploited in data-intensive applications, e.g., for media content or news feeds [Mat+12]. The second main application area of rumor spreading (here often called gossip-based algorithms) are wireless sensor networks and mobile ad-hoc networks. Here, the simple paradigm of contacting random neighbors is used to overcome the difficulties imposed by the changing and unreliable network topology, see, e.g., [IS10].

Basic Rumor Spreading. Possibly the most basic rumor spreading process regarded already in the paper by Frieze and Grimmett [FG85], aiming to spread a rumor in a network of n nodes who can all communicate with each other, proceeds as follows. We start with a single node possessing a piece of information ("rumor"). The process works in discrete time steps ("rounds"). In each round, each informed node calls a node chosen uniformly at random (including itself) and gossips the rumor to it, making the node informed if it was not before.

This process is sometimes called *synchronized rumor spreading in the push model in a complete graph*. Note that this basic process also models the spread of a single piece of information in a system where several rumors are disseminated. Note also that the assumption that all nodes can communicate with each other makes sense even in networks where there is no physical connection between any two nodes. In such networks, gossip-based algorithms usually are implemented building upon a peer-sampling service [Jel+07], which enables the individual node to connect to a random other one.

Previous Results. For the basic rumor spreading process, two classical analyses exist. For simplicity, denote by S_n the number of rounds performed until (for the first time) all nodes are informed. This random variable is also called the *broadcast time* of the rumor spreading process.

Already Frieze and Grimmett [FG85] give the fairly precise result that $S_n = (1 \pm o(1))(\log_2 n + \ln n)$ with probability $1 - o(1)$. They also prove that for all $\varepsilon, \gamma > 0$, with probability at least $1 - o(n^{-\gamma})$, the broadcast time does not exceed $(1+\varepsilon)(\log_2 n + \gamma \ln n)$. The first bound was sharpened by Pittel [Pit87], who proved that for any $h = \omega(1)$, we have $\Pr(|S_n - \log_2 n - \ln n| \geq h(n)) \to 0$.

Our Results. Despite the strong results of Frieze and Grimmett [FG85] and Pittel [Pit87], it is still surprising that a simple process like basic randomized rumor

spreading is not even better understood. Recall, for example, that the coupon collector process is much better analyzed. We can precisely describe the time needed to collect all coupons as the sum $C_n = X_1 + \ldots + X_n$ of n independent geometrically distributed random variables X_i with success rates $p_i = i/n$. Consequently, the expected time the process takes is $nH_n = n\ln(n) + \gamma n + 1/2 - O(1/n)$. For the basic rumor spreading process, we are not aware of any proof for a $\log_2 n + \ln n + \Theta(1)$ bound for the expected runtime, let alone a precise description of the distribution. It might be possible to derive results in this direction from a careful analysis of the proof in [Pit87], but since the 8-page proof analyzing the process in 7 different phases is quite technical, we preferred to use an alternative route.

We prove that the expected time needed to inform all n nodes via the basic rumor spreading process is at most

$$\mathrm{E}[S_n] \leq \lceil \log_2 n \rceil + \ln n + 2.765 + o(1).$$

In addition to the expectation, we show that the random variable describing the rumor spreading time is dominated by

$$S_n \preceq \lceil \log_2 n \rceil + 2.562 + o(1) + \frac{1 + O(n^{-\frac{1}{2}+\varepsilon})}{n} \cdot C_n + \mathrm{Geom}(1 - O(n^{-1+\varepsilon})),$$

where C_n is the time needed to collect n coupons, $\mathrm{Geom}(p)$ is an independent geometric random variable with success probability p, and ε is an arbitrarily small constant. Furthermore, from the stochastic dominance result, the following tail bound is immediately derived. With probability at least $1 - 2e^{-r}$, we have

$$S_n \leq \lceil \log_2(n) \rceil + \ln(n) + 2.188 + r,$$

for sufficiently large n. These results are proven in Section 13.2 as Theorem 13.2.6 as well as Corollaries 13.2.7 and 13.2.8.

Our upper bounds are quite sharp. For the expectation, a lower bound of $\lfloor \log_2 n \rfloor + \ln n - 1.116$ is not difficult to prove, see Corollary 13.3.2 in Section 13.3. Also, for the distribution, we show that S_n is subdominated by $\lfloor \log_2 n \rfloor - 1 + (1/n) \sum_{i=1}^{n/2} X_i$ (Theorem 13.3.1), where as above the X_i are independent geometric random variables with success probability $p_i = i/n$.

Technical Contributions. In addition to proving the relatively precise results stated above, we also feel that our analysis method is even a little simpler than the previous works. We use the following two elementary, but powerful arguments.

(i) Imagine that at some time we have at least n_1 nodes informed. Instead of trying to estimate the result of a single round starting with n_1 informed nodes, we fix a *target number* n_2 of nodes such that with high probability $1 - q$, at least n_2 nodes are informed after one round. In case this fails in a single round, we simply try again. Consequently, we have that in a *subphase* of length $1 + \mathrm{Geom}(1 - q)$, we surely go from at least n_1 informed nodes to at least n_2 informed nodes. By this, we avoid dealing with the distribution of the number of informed nodes at particular times—of course at the price of dealing with random completion times, but this is less critical since their distributions are simply a fixed number plus some geometrically distributed random variables with success probabilities very close to one. What remains is to cleverly choose the target numbers. On the one hand, they should be large to ensure that the number of subphases

(and consequently the number of rounds needed) is small, on the other, they have to be small enough such that the failure probabilities q are very small.

(ii) For the part of the process leading from cn informed nodes (for some constant $0 < c < 1$) to all nodes informed, we use an elegant reduction to the coupon collector process. We give (not too sharp) lower bounds for the number of informed nodes in these rounds, building on the elementary observation that the number of uninformed nodes typically shrinks by a constant factor. This factor is known to approach $1/e$, but we shall not exploit this and use a weaker factor. This weaker factor is enough to see that in the following κ rounds, a total of $\kappa n - O(n)$ random contacts are made. From the coupon collector process, it is well known how many random calls are needed to ensure that each of the missing $(1 - c)n$ nodes receives a call.

We did not try to optimize the additive constants in our bounds, though we imagine that our proof method allows making them precise with moderate additional effort. We also note that our results confirms, and strengthens, previous observations that the first part of the process up to a constant fraction of informed nodes shows very little variation in the runtime. For example, we shall prove that with probability $1 - n^{-1+\varepsilon}$, after $\lceil \log_2 n \rceil - 1$ rounds, at least $(5/16)n - o(n)$ nodes are informed. Note that within $\log_2 n - 2$ rounds, no rumor spreading process (where each informed node does one call per round) can inform more than $n/4$ nodes.

12.2 The 2-Player Cryptogenography Problem

Motivated by a number of recent influential cases of whistle-blowing, Brody, Jakobsen, Scheder and Winkler [Bro+14] proposed the following *cryptogenography problem* as model for anonymous information disclosure in public networks, which is the subject of Chapter 14. We have k players (potential information leakers). A random one of them holds a secret, namely a random bit. All other players only know that they are not the secret holder. Now without any non-public communication, the players aim at both making the secret public and hiding the identity of the secret holder. More precisely, we are looking for a (fully public) communication protocol in which the players – as only form of communication – broadcast bits, which may depend on public information (including all previous communication), private knowledge (with respect to the secret), and a private source of randomness. After this phase of communication, the protocol outputs a single bit depending solely on all data sent in the communication phase. The complete protocol (regulating the communication and the output function) and all communication is public, and is monitored by an eavesdropper who aims at identifying the secret owner. We say that a run of the protocol is a success for the players, if the protocol output is the secret bit and the eavesdropper fails to identify the secret owner; otherwise it is a success for the eavesdropper. Since everything is public, optimal strategies for the eavesdropper are easy to find (see below). We shall therefore always assume that the eavesdropper plays an optimal strategy. The players' success probability (for a given protocol) then is the probability (taken over the random decisions of the players and the random initial secret distribution) that simultaneously (i) the protocol outputs the true secret and (ii) an optimal eavesdropper does not blame the secret holder.

It is immediately clear that some positive (players') success probability is easy to obtain. A protocol without any communication and outputting a random bit achieves a success probability of $\frac{1}{2} - \frac{1}{2k}$ (the eavesdropper has no strictly better alternative than guessing a random player). Surprisingly, Brody et al. could show that the players, despite the complete absence of private communication, can do better. For two players, they

present a protocol having a success probability of $\frac{1}{3}$ (instead of the trivial $\frac{1}{4}$). For k sufficiently large, they present a protocol with success probability 0.5644. They also show two hardness results, namely that a success probability of more than $\frac{3}{4}$ cannot be obtained, regardless of the number of players, and that $\frac{3}{8}$ is an upper bound for the two-player case. While all these results are easy to state, they build on deep analyses of the cryptogenography problem, in particular, on clever reformulations of the problem in terms of certain convex combinations of secret distributions and functions that are concave on a certain infinite set of two-dimensional subspaces ("allowed planes") of the set of secret distributions.

The starting point for our work is the incomplete understanding of the two-player case. While the gap between upper and lower bound of $\frac{3}{8} - \frac{1}{3} \approx 0.04167$ is small, our impression is that the current-best protocol achieving success probability $\frac{1}{3}$ in two rounds together with the abstract hardness result do not give us much understanding of the structure of the cryptogenography problem. We therefore imagine that a better understanding of this smallest possible problem of leaking one bit from two players, ideally by determining an optimal protocol (that is, matching a hardness result), could greatly improve the situation.

Our Results. We shall be partially successful in achieving the above stated goals. On the positive side, we find protocols with strictly larger success probability than $\frac{1}{3}$ (namely 0.3384) and we prove a stricter hardness result of 0.3672. Our new protocols look very different from the 2-round protocol given by Brody et al., in particular, they use infinite protocol trees (but have an expected finite number of communication rounds). These findings motivate and give new starting points for further research on the cryptogenography problem.

On the not so positive side, our work on better protocols indicates that good cryptogenographic protocols can be very complicated. The simplest protocol we found that beats the $\frac{1}{3}$ barrier already has a protocol tree of depth 16, that is, the two players need to transmit 16 bits in total in the worst case. While we still manage to give a human-readable description and performance proof for this protocol, it is not surprising that previous works not incorporating a computer-assisted search did not find such a protocol. Our best protocol, giving a success probability of 0.3384, already uses 18248 non-equivalent states.

Technical Contributions. To find the improved protocols, we use a number of theoretical and experimental tools. We first reformulate the cryptogenography problem as a solitaire vector splitting game over vectors in $\mathbb{R}_{\geq 0}^{2\times k}$. Both for human researchers and for automated protocol searches, this reformulation seems to be easier to work with than the previous reformulation via convex combinations of distributions lying in a common allowed plane [Bro+14]. It also proved to be beneficial for improving upon the hardness result.

Restrictions of the vector splitting game to a finite subset of $\mathbb{R}_{\geq 0}^{2\times k}$, e.g., $\{0, \dots, T\}^{2\times k}$, can easily be solved via dynamic programming, yielding (due to the restriction possibly sub-optimal) cryptogenographic protocols. Unfortunately, for $k = 2$ even discretizations as fine as $T = 40$ are not sufficient to find protocols beating the 1/3 barrier and memory usage quickly becomes a bottleneck issue. However, exploiting the simple fact that the game values are homogeneous (that is, multiplying a game position by a non-negative scalar changes the game value by this factor), we can (partially) simulate a much finer discretization in a coarse one. This *extended dynamic programming approach* easily gives

cryptogenographic success probabilities larger than 1/3. Reading off the corresponding protocols, due to the reuse of the same position in different contexts, needs more care, but in the end gives without greater difficulties also the improved protocols.

When a cryptogenographic protocol reuses a state a second time (with a non-trivial split in between), then there is no reason to re-iterate this part of the protocol whenever this position occurs. Such a protocol allows infinite paths, while still needing only an expected finite number of rounds. Since the extended dynamic programming approach in finite time cannot find such protocols, we use a linear programming based post-processing stage. We translate each splitting operation used in the extended dynamic programming search into an inequality relating game values. By exporting these into an LP-solver, we do not only obtain better game values (possibly corresponding to cryptogenographic protocols with infinite paths, for which we would get a compact representation by making the cycles explicit), but also a way to easily certify these values using an optimality check for a linear program instead of having to trust the ad-hoc dynamic programming implementation.

Related Work. Despite a visible interest of the research community in the cryptogenography problem[1], the only relevant follow-up work is Jakobsen's paper [Jak14], which analyses the cryptogenography problem for the case that several of the players know the secret. This allows to leak a much larger amount of information as made precise in [Jak14]. Due to the asymptotic nature of these results, unfortunately, they do not give new insights in the 2-player case. Other work on anonymous broadcasting typically assumes bounded computational power of the adversary (see, e.g., [JO16]); we refer to [DD08] for a survey on anonymous communication in networks.

In [Bro+14], the cryptogenography problem was reformulated to the problem of finding the point-wise minimal function f on the set of secret distributions that is point-wise not smaller than some given function g and that is concave on an infinite set of 2-dimensional planes. Such restricted notions of concavity (or, equivalently, convexity) seem to be less understood. There exists work by Matoušek [Mat01] for a similar convexity problem, however, with only a finite number of one-dimensional directions in which convexity is required. We do not see how to extend these results to our needs.

12.3 Notes

The contents of Chapter 13 have previously been published at ANALCO'14 [DK14]. The contents of Chapter 14 have been accepted for publication at ICALP'16 [DK16a] (with an extended online version accessible at [DK16b]).

Acknowledgements

The author wishes to thank the anonymous reviewers of our ANALCO'14 and ICALP'16 submissions for their helpful remarks, Benjamin Doerr for his guidance and support and Richard J. Lipton for maintaining a wonderful blog without which an interesting problem might have gone unnoticed to us.

[1] See, e.g., `https://rjlipton.wordpress.com/2013/12/13/who-knew-the-secret/` and note that Jakobsen received the ICALP 2014 Track A Best Student Paper Award for his follow-up paper [Jak14].

Chapter 13

Rumor Spreading in Complete Graphs

In this chapter, we give a precise analysis of the rumor spreading time of the classical (synchronized) push protocol for randomized rumor spreading in complete graphs (see Chapter 12 for an overview of the results). We introduce the process, corresponding notation as well as basic concepts and tools from probability theory in Section 13.1. An upper bound (in form of a stochastic dominance result) is obtained in Section 13.2 by successively analyzing three phases of the process and finally connecting these results. A complementing lower bound is given in Section 13.3, concluding the chapter.

13.1 Notation and Preliminaries

Throughout this chapter, we analyze the basic rumor spreading process introduced by Frieze and Grimmett [FG85] (see also [Pit87] for a beautiful description of the process). The process starts with one node of a complete network on the n nodes $[n] := \{1, \dots, n\}$ knowing a rumor. In each round of the process, each node that knows the rumor chooses a node uniformly at random (including possibly itself) and gossips the rumor to that node, which becomes informed in case it was not before. We denote by I_t the set of informed nodes after the completion of round t. Let I_0 consist of the single initially informed node. By $U_t := [n] \setminus I_t$, we denote the set of uninformed nodes after completion of round t. Our main concern is the time needed to inform all nodes, that is, $S_n := \min\{t \mid |I_t| = n\}$.

Before starting the analysis of this process, let us collect a few probabilistic tools needed in the proofs.

13.1.1 Domination, Negative Association and Tail Bounds

Let a pair of random variables X, Y be given. We say that X stochastically dominates Y, and write $Y \preceq X$, if $\Pr[X \geq x] \geq \Pr[Y \geq x]$ for all x.

The following lemma is a standard tool to bound the deviation of sums of independent random variables from their expectation, see, e.g., [DP09].

Lemma 13.1.1 (Chernoff-Hoeffding Bound, [DP09, Theorem 1.1]). *Let $X := \sum_{i=1}^{n} X_i$, where the X_i are independently distributed in $[0, 1]$. Then,*

(i) for $0 < \delta < 1$,

$$\Pr[X \leq (1 - \delta)\,\mathrm{E}[X]] \leq e^{-\mathrm{E}[X]\delta^2/2},$$

(ii) for all $\delta > 0$,

$$\Pr[X \geq (1 + \delta)\,\mathrm{E}[X]] \leq \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{\mathrm{E}[X]}.$$

Many of these large-deviation bounds also hold in some dependent settings. The following lemma establishes a case in which a sequence of non-independent binary random variables $X_1, \dots, X_n$ can be substituted by independent random variables whose sum dominates (or subdominates) $\sum_{i=1}^n X_i$. This makes it possible to use the previous lemma to derive tail bounds for $\sum_{i=1}^n X_i$. This fact seems to be well known, but the only published proof we are aware of is Lemma 1.18 and 1.19 in [Doe11].

Lemma 13.1.2. *Let $X_1, \dots, X_n$ be arbitrary binary random variables. Let $X_1^*, \dots, X_n^*$ be binary random variables that are mutually independent and such that for all i, X_i^* is independent of $X_1, \dots, X_{i-1}$. If for all i and all $x_1, \dots, x_{i-1} \in \{0,1\}$,*

$$\Pr[X_i = 1 \mid X_1 = x_1, \dots, X_{i-1} = x_{i-1}] \geq \Pr[X_i^* = 1],$$

then for all $k \geq 0$, we have

$$\Pr\left[\sum_{i=1}^n X_i < k\right] \leq \Pr\left[\sum_{i=1}^n X_i^* < k\right].$$

Similarly, if for all i and all $x_1, \dots, x_{i-1} \in \{0,1\}$,

$$\Pr[X_i = 1 \mid X_1 = x_1, \dots, X_{i-1} = x_{i-1}] \leq \Pr[X_i^* = 1],$$

then for all $k \geq 0$, we have

$$\Pr\left[\sum_{i=1}^n X_i > k\right] \leq \Pr\left[\sum_{i=1}^n X_i^* > k\right].$$

A more intricate dependent setting allowing for Chernoff-type bounds is the case of negatively associated random variables. We say that random variables $X_1, \dots, X_n$ are negatively associated, if for all disjoint subsets $I, J \subseteq [n]$ and all non-decreasing functions f and g, we have

$$\mathrm{E}[f(X_i, i \in I) g(X_j, j \in J)] \leq \mathrm{E}[f(X_i, i \in I)] \, \mathrm{E}[g(X_i, i \in I)].$$

This notion has been studied especially in the balls-into-bins model, which has a direct correspondence to our rumor spreading process. In particular, we will exploit the following basic fact.

Lemma 13.1.3 (Balls Into Bins, Negative Association [DP09, Example 3.2])**.** *Let m balls be thrown independently and uniformly at random into n bins. The indicator variables Z_i defined by $Z_i = 1$ if and only if bin i is empty are negatively associated.*

The above fact allows us to use the following large-deviation bound.

Lemma 13.1.4 (Chernoff-Hoeffding Bound, Negative Association [DP09, Theorem 3.1])**.** *Let $X := \sum_{i=1}^n X_i$, where the X_i are negatively associated random variables taking values in $[0,1]$. Then for all $t > 0$,*

$$\Pr[X > \mathrm{E}[X] + t] \leq e^{-2t^2/n}.$$

We say that a random variable G is geometrically distributed with success probability p, and write $G \sim \mathrm{Geom}(p)$, if $\Pr[G = i] = (1-p)^i p$ for all $i \in \mathbb{N}_0 = \{0, 1, \dots, \}$ – note that by definition the smallest possible realization of G is the value zero. Equivalently,

$G \sim \mathrm{Geom}(p)$ whenever $\Pr[G \geq i] = (1-p)^i$ holds for all $i \in \mathbb{N}_0$. In the first phases of the process, we will bound the expected number of rounds to inform a certain number of nodes by a deterministic number and a sum of geometrically distributed variables. This value is typically dominated by the single geometric random variable with the smallest success probability. To use this fact for simplifying the results, we provide the following convenient lemma.

Lemma 13.1.5. *Let $G_1, \ldots, G_n$ be independent random variables with $G_i \sim \mathrm{Geom}(1-q_i)$. Then $\sum_{i=1}^n G_i$ is stochastically dominated by a random variable G with $G \sim \mathrm{Geom}(1 - \sum_{i=1}^n q_i)$.*

Proof. Let G_1, G_2 be independent geometrically distributed random variables with success probabilities $1-q_1$ and $1-q_2$, respectively. By law of total probability, we compute for all $t \geq 0$,

$$\begin{aligned}
\Pr[G_1 + G_2 \geq t] &= \left(\sum_{k=0}^{t-1} \Pr[G_1 = k] \Pr[G_2 \geq t-k]\right) + \Pr[G_1 \geq t] \\
&= \left(\sum_{k=0}^{t-1} (1-q_1) q_1^k q_2^{t-k}\right) + q_1^t \\
&\leq \sum_{k=0}^{t} q_1^k q_2^{t-k} \leq \sum_{k=0}^{t} \binom{t}{k} q_1^k q_2^{t-k} = (q_1+q_2)^t.
\end{aligned}$$

Hence, $G_1 + G_2 \preceq \mathrm{Geom}(1-(q_1+q_2))$. By successive application of this fact, we obtain $\sum_{i=1}^n G_i \preceq \mathrm{Geom}(1 - \sum_{i=1}^n q_i)$. □

13.1.2 The Coupon Collector Process

A central approach of our analysis is a reduction from the final stages of the rumor spreading process to the coupon collector process. In this process, we independently draw coupons $c_1, c_2, \ldots$, each having a random, equiprobable *(coupon) type* out of n distinguishable types. Let an initial collection of coupons having in total $0 \leq m < n$ different types be given. We let $C_n(n-m)$ denote the number of draws until the remaining $n-m$ coupon types have been collected. We write $C_n := C_n(n)$ for short to denote the number of draws in the typical coupon collector process that starts with an empty initial collection.

It is well known that the time to collect the remaining m coupon types, when starting with $n-m$ collected types, can be described by the sum $C_n(m) = X_1 + \cdots + X_m$ of independent geometric random variables X_i with success probability $p_i = i/n$. Let $H_i := \sum_{j=1}^i \frac{1}{j}$ denote the i-th harmonic number, then we obtain $\mathrm{E}[C_n(m)] = nH_m$. Together with the fact that

$$1/(2n) - 1/(8n^2) \leq H_n - (\ln(n) + \gamma) \leq 1/(2n),$$

where $\gamma = 0.57721\ldots$ is the Euler-Mascheroni constant (see, e.g., Pt. II, Ex 18 in [PS72]), this yields a very sharp bound of $\mathrm{E}[C_n(m)/n] = \ln(m) + \gamma + \mathcal{O}(1/m)$.

To bound the deviation of the coupon collector's time, we exploit the following lemma, which is a slight generalization of the classic upper tail bound for the coupon collector process.

Lemma 13.1.6 (Coupon Collector Tail Bound)**.** *Let $1 \leq m \leq n$ and $r \geq 0$. Then,*

$$\Pr[C_n(m) \geq n\ln(m) + rn] \leq e^{-r}.$$

Proof. By union bound, the probability that in $t := n\ln(m) + rn$ rounds, one of the initially missing m coupons types is never drawn, is bounded from above by

$$m\left(1 - \frac{1}{n}\right)^t \leq me^{-(\ln(m)+r)} = e^{-r}. \qquad \square$$

In our analysis, we couple the rumor spreading process with the coupon collector by identifying coupons with message receivers. We will count the number of messages sent in a sequence of rounds that, by our analysis, satisfy a lower bound on the number of newly informed nodes. Although this conditional process is a modification of the original coupon collector process, it is easy to see that it is only pessimistic to regard the original process instead. The following lemma makes this claim precise by showing that bounding the number of collected coupons from below only decreases the number of draws remaining until all coupons are collected.

Lemma 13.1.7. *Let $C_1, C_2 \ldots$ be a sequence of coupons drawn independently and uniformly at random from $[n]$, let $X \subseteq [n]$ be a set of x distinguished coupon types and M be arbitrary. We define $S(C_1, \ldots, C_M) \subseteq X$ as the set of coupon types of X collected among $C_1, \ldots, C_M$. Then, for any $0 \leq m \leq x$ and r,*

$$\Pr\left[C_n(x) \leq r \,|\, |S(C_1, \ldots, C_M)| \geq m\right] \geq \Pr[C_n(x) \leq r],$$

i.e., $C_n(x)$ conditioned on $|S(C_1, \ldots, C_M)| \geq m$ is stochastically dominated by $C_n(x)$.

Proof. We show that

$$\Pr[C_n(x) \leq r \mid |S(C_1, \ldots, C_M)| \geq m] \geq \Pr[C_n(x) \leq r \mid |S(C_1, \ldots, C_M)| < m],$$

from which the claim follows by the law of total probability. If $r \leq M$, then the condition $|S(C_1, \ldots, C_M)| < m \leq x$ already implies $\Pr[C_n(x) \leq r \mid |S(C_1, \ldots, C_M)| < m] = 0$, thus the claim holds trivially.

For $r > M$, let $\gamma_1, \ldots, \gamma_M$ and $\bar{\gamma}_1, \ldots, \bar{\gamma}_M$ be arbitrary realizations of $C_1, \ldots, C_M$ satisfying $|S(\gamma_1, \ldots, \gamma_M)| < m$ and $|S(\bar{\gamma}_1, \ldots, \bar{\gamma}_M)| \geq m$. The smaller set might not be contained in the larger one, however, we may couple both situations by a simple renaming of the coupons, using any permutation $\pi : [n] \to [n]$ such that $\pi(S(\gamma_1, \ldots, \gamma_M)) \subseteq S(\bar{\gamma}_1, \ldots, \bar{\gamma}_M)$ and $\pi(X) = X$. Consider any realization $C_{M+1} = c_{M+1}, \ldots, C_r = c_r$ with $|S(\gamma_1, \ldots, \gamma_M, c_{M+1}, \ldots, c_r)| = x$, then $c_{M+1}, \ldots, c_r$ must contain the coupons $X \setminus S(\gamma_1, \ldots, \gamma_M)$. By definition of π, this implies that

$$\begin{aligned} S(\pi(c_{M+1}), \ldots, \pi(c_r)) &= \pi(S(c_{M+1}, \ldots, c_r)) \\ &\supseteq \pi(X \setminus S(\gamma_1, \ldots, \gamma_M)) \supseteq \pi(X) \setminus S(\bar{\gamma}_1, \ldots, \bar{\gamma}_M). \end{aligned}$$

This implies $|S(\bar{\gamma}_1, \ldots, \bar{\gamma}_M, \pi(X_{M+1}), \ldots, \pi(X_r))| = |\pi(X)| = x$. After noting that $C_{M+1}, \ldots, C_r$ is equally distributed to $\pi(C_{M+1}), \ldots, \pi(C_r)$, this immediately yields

$$\Pr\left[C_n(x) \leq r \,|\, X_1 = \gamma_1, \ldots, X_M = \gamma_M\right] \leq \Pr\left[C_n(x) \leq r \,|\, X_1 = \bar{\gamma}_1, \ldots, X_M = \bar{\gamma}_M\right],$$

proving the claim. $\square$

13.2 Upper Bounds

To prove the upper bounds, we use three different ways to prove lower bounds on the numbers of informed vertices. When there are few informed vertices, a birthday-paradox-type computation shows that very likely, all calls reach different uninformed vertices (Phase 1). When there are more, but at most a small constant fraction of informed nodes, a similar argument together with a Chernoff bound argument shows that the number of informed nodes almost doubles (Phase 2). When there are even more informed nodes, then, like previous works, switching the focus to the uninformed nodes and again using Chernoff bounds shows that the number of uninformed nodes shrinks by a constant factor (Phase 3). This shows that in the following p_3 rounds, $p_3 n - \mathcal{O}(n)$ random calls are made in total. Together with a reduction to the coupon collector process, we derive a bound on the rumor spreading time.

13.2.1 Phase 1

When only few nodes are informed, the random calls performed by these nodes have a very high chance of targeting an uninformed node, and consequently, there is a good chance that the number of informed nodes doubles. The following lemma makes this observation precise.

Lemma 13.2.1 (Phase 1)**.** *Let $n_1 < \sqrt{n}$ be a power of two and $t_1 := \min\{t \geq 0 \mid |I_t| \geq n_1\}$. Then t_1 is stochastically dominated by $\log_2(n_1) + \mathrm{Geom}(1 - n_1^2/n)$.*

Proof. Let $p_1 := \log_2 n_1$, $1 \leq j \leq p_1$ and $t \geq 0$. Assume that in the $(t+1)$-st round, the $|I_t|$ informed nodes perform their actions in some given order. Then, when the k-th node chooses its random communication partner, at most $|I_t| + k - 1$ nodes are informed. Consequently, with probability at least $1 - (|I_t| + k - 1)/n$ it calls an uninformed node and informs it. Hence

$$\begin{aligned} \Pr\left[|I_{t+1}| \geq 2^j \mid |I_t| \geq 2^{j-1}\right] &\geq \prod_{k=0}^{2^{j-1}-1} \left(1 - \frac{2^{j-1}+k}{n}\right) \\ &\geq 1 - \frac{2^{2(j-1)} + \sum_{k=0}^{2^{j-1}-1} k}{n} \geq 1 - \frac{(3/2)2^{2(j-1)}}{n}, \end{aligned}$$

where the second inequality follows from Bernoulli's inequality.

We divide the process until at least n_1 nodes are informed into p_1 subphases, where each subphase j ends as soon as at least 2^j nodes are informed. These subphases almost surely consist of a single round each. Formally, define $t^{(0)} := 0$ and introduce the random stopping times

$$t^{(j)} := \min\{t \geq 0 \mid |I_t| \geq 2^j\}, \qquad \text{for } 1 \leq j \leq p_1.$$

Clearly, at time $t^{(p_1)} = \sum_{j=1}^{p_1} t^{(j)} - t^{(j-1)}$ at least $2^{p_1} = n_1$ nodes are informed. For $1 \leq j \leq p_1$, the above calculation shows that $t^{(j)} - t^{(j-1)}$ is stochastically dominated by $1 + \mathrm{Geom}(1 - q_j)$ with $q_j := (3/2)2^{2(j-1)}/n$, since in each round t with $|I_t| \geq 2^{j-1}$, we have that $|I_{t+\ell}| < 2^j$ with probability at most q_j^ℓ for each $\ell \geq 1$.

We introduce independent random variables $G_j \sim \mathrm{Geom}(1 - q_j)$ and obtain

$$t^{(p_1)} \preceq p_1 + \sum_{j=1}^{p_1} G_j.$$

To complete the proof, note that

$$\sum_{j=1}^{p_1} q_j \leq \sum_{j=1}^{p_1} \frac{(3/2)2^{2(j-1)}}{n} \leq \frac{2^{2p_1-1}}{n},$$

and apply Lemma 13.1.5. This proves the claim. □

13.2.2 Phase 2

When more than just very few nodes are informed, the probability for the informed nodes to double in one round is smaller than what we are willing to tolerate. However, the number of informed nodes still increases, in expectation, almost by a factor of two per round, and is additionally closely concentrated around its expectation. This is what we shall exploit in this subsection.

Let $1 \leq k \leq \frac{n}{2}$ and assume that $|I_t| \geq k$. Enumerate k nodes from I_t in an arbitrary manner $u_1, \ldots, u_k$. Denote by $c(u_j)$ the random node called by u_j in the $(t+1)$-st round. Let X_j be the indicator random variable for the event $c(u_j) \notin I_t \cup \{c(u_1), \ldots, c(u_{j-1})\}$. Then $|I_{t+1}| \geq k + \sum_{j=1}^{k} X_j$. While $X_1, \ldots, X_k$ are not independent, for all $j = 1, \ldots, k$, they satisfy the property that regardless of the outcome of $X_1, \ldots, X_{j-1}$, we have $\Pr[X_j = 1] \geq 1 - (k + j - 1)/n$. This not only allows us to bound the expectation of $|I_{t+1}|$ by

$$\mathrm{E}\left[|I_{t+1}|\right] \geq 2k - \frac{k^2 + \sum_{i=0}^{k-1} i}{n} \geq 2k - \frac{3k^2}{2n} = 2k\left(1 - \frac{3k}{4n}\right), \tag{13.1}$$

but also allows to use Chernoff bounds. By Lemma 13.1.2, the above property implies that $X_1 + \ldots + X_k$ stochastically dominates $Y_1 + \ldots + Y_k$, where the Y_j are independent binary random variables with $\Pr[Y_j = 1] = 1 - (k + j - 1)/n$. For convenience, we define $Y = \sum_{j=1}^{k} Y_j$ and note that by an analogous calculation to (13.1), we obtain $\mathrm{E}[Y] \geq k(1 - \frac{3k}{2n})$.

Lemma 13.2.2. *Let $0 < s < \frac{1}{2}$ and $\frac{n^s}{2} \leq k \leq \frac{n}{2}$. Then*

$$\Pr\left[|I_{t+1}| \leq 2k\left(1 - \frac{3k}{4n} - \frac{1}{2n^{\frac{s}{2}}}\right) \,\middle|\, |I_t| \geq k\right] \leq e^{-n^{\frac{s}{2}}/24}.$$

Proof. Assume that $|I_t| \geq k$. If the event $|I_{t+1}| \leq 2k\left(1 - \frac{3k}{4n} - \frac{1}{2n^{\frac{s}{2}}}\right)$ occurs, we have

$$\begin{aligned} \sum_{j=1}^{k} X_j &\leq k\left(1 - \frac{3k}{2n} - \frac{1}{n^{\frac{s}{2}}}\right) \\ &\leq \left(1 - \frac{1}{n^{\frac{s}{2}}}\right) k\left(1 - \frac{3k}{2n}\right) \leq \left(1 - \frac{1}{n^{\frac{s}{2}}}\right) \mathrm{E}\left[Y\right], \end{aligned} \tag{13.2}$$

where the random variables $Y_1, \ldots, Y_k$ and $Y = \sum_{j=1}^{k} Y_j$ are as defined above.

If $\frac{1}{3}n^{1-\frac{s}{2}} \leq k \leq \frac{n}{2}$, we can simply apply the Chernoff bound of Lemma 13.1.1(i), and obtain

$$\begin{aligned}
&\Pr\left[|I_{t+1}| \leq 2k\left(1-\frac{3k}{4n}-\frac{1}{2n^{\frac{s}{2}}}\right) \,\middle|\, |I_t| \geq k\right] \\
&\leq \Pr\left[\sum_{j=1}^{k} Y_j \leq \left(1-\frac{1}{n^{\frac{s}{2}}}\right)\mathrm{E}\left[Y\right]\right] \\
&\leq \exp\left(-\frac{1}{2}\left(\frac{1}{n^{\frac{s}{2}}}\right)^2 k\left(1-\frac{3k}{2n}\right)\right) \\
&\leq \exp\left(-\frac{k}{8n^s}\right) \leq \exp(-n^{1-\frac{3}{2}s}/24) \leq \exp(-n^{\frac{s}{2}}/24),
\end{aligned}$$

where the second line follows from Lemma 13.1.2, the third from Lemma 13.1.1(i) as well as $\mathrm{E}[Y] \geq k(1-\frac{3k}{2n})$, and finally the last from $\frac{1}{3}n^{1-\frac{s}{2}} \leq k \leq n/2$ and $s < \frac{1}{2}$.

If $\frac{1}{2}n^s \leq k \leq \frac{1}{3}n^{1-\frac{s}{2}}$, it is useful to regard $\bar{X}_j := 1 - X_j$ and reformulate (13.2) to the equivalent inequality $\sum_{j=1}^{k} \bar{X}_j \geq \frac{3k^2}{2n} + \frac{k}{n^{\frac{s}{2}}}$. Analogously, we define $\bar{Y}_j := 1 - Y_j$ and $\bar{Y} := \sum_{j=1}^{k} \bar{Y}_j$, resulting in $\mathrm{E}[\bar{Y}] \geq \frac{3k^2}{2n}$. Setting δ such that $(1+\delta)\,\mathrm{E}[\bar{Y}] = \frac{3k^2}{2n} + \frac{k}{n^{\frac{s}{2}}}$, we see that $\delta \geq \frac{2n^{1-\frac{s}{2}}}{3k} \geq 2$. Thus similar to above, we compute

$$\begin{aligned}
&\Pr\left[|I_{t+1}| \leq 2k\left(1-\frac{3k}{4n}-\frac{1}{2n^{\frac{s}{2}}}\right) \,\middle|\, |I_t| \geq k\right] \\
&\leq \Pr\left[\sum_{j=1}^{k} \bar{Y}_j \geq (1+\delta)\,\mathrm{E}\left[\bar{Y}\right]\right] \\
&\leq \left(\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\right)^{\mathrm{E}[\bar{Y}]} \leq \left(\frac{e}{3}\right)^{(1+\delta)\,\mathrm{E}[\bar{Y}]} \\
&\leq \left(\frac{e}{3}\right)^{\frac{k}{n^{\frac{s}{2}}}} \leq \exp\left(-\frac{k}{12n^{\frac{s}{2}}}\right) \leq \exp\left(-n^{\frac{s}{2}}/24\right),
\end{aligned}$$

where the second line follows from Lemma 13.1.2, the third from Lemma 13.1.1(ii) and the observation $\delta \geq 2$, and finally the last from $(1+\delta)\,\mathrm{E}[\bar{Y}] \geq \frac{k}{n^{\frac{s}{2}}}$ and $k \geq \frac{n^s}{2}$. □

Equipped with the lemma above, we can show that for any $\ell \geq 1$, intuitively after $\lceil \log_2 n \rceil - \ell$ rounds, plus a small geometrically distributed random number of extra rounds, we expect to inform about $n(\frac{1}{2^\ell} - \frac{3}{4}\frac{1}{2^{2\ell}})$ nodes. For $\ell = 2$, this corresponds to a $(\frac{1}{4} - \frac{3}{64})$-fraction of the nodes (deviating only slightly from the best-possible fraction of $\frac{1}{4}$ whenever n is a power of two), while for $\ell = 1$, this still amounts to $\frac{5}{16}$ of the nodes.

In the proof of the following lemma, we use the analysis provided by Phase 1 to boost the number of informed nodes at the beginning of Phase 2 to n^s (for small s).

Lemma 13.2.3 (Phase 2)**.** *Let $\ell \geq 1$ be an integer and $s < 1/2$ with $n^{\frac{s}{2}} \geq (3/2)\log_2 n$. Define $n_2 := n\left(\frac{1}{2^\ell} - \frac{3}{4}\frac{1}{2^{2\ell}} - \varepsilon\right)$, where $\varepsilon := \frac{\log_2(n)}{2n^{\frac{s}{2}}}$, and correspondingly, $t_2 := \min\{t \geq 0 \mid |I_t| \geq n_2\}$. Then t_2 is stochastically dominated by*

$$\lceil \log_2(n) \rceil - \ell + \mathrm{Geom}(1 - (n^{-1+2s} + \log_2(n)e^{-n^{\frac{s}{2}}/24})).$$

Proof. We define n_1 as the largest power of two that is at most n^s, hence $n^s/2 \leq n_1 \leq n^s$. We apply the Phase 1 Lemma (Lemma 13.2.1) to see that $t_1 := \min\{t \geq 0 \mid |I_t| \geq n_1\}$ is dominated by $\log_2(n_1) + \mathrm{Geom}(1 - n_1^2/n)$. Similarly to the proof of the Phase 1 Lemma, we introduce subphases $t^{(1)}, t^{(2)}, \ldots$ with

$$t^{(j)} := \min\{t \geq 0 \mid |I_t| \geq a_j\},$$

where $a_0 := n_1$ and $a_i := 2a_{i-1}(1 - \frac{3a_{i-1}}{4n} - \frac{1}{2n^{\frac{s}{2}}})$. Trivially, $a_i \leq 2^i a_0$. Hence,

$$a_i \geq 2a_{i-1}\left(1 - \frac{3}{4}\frac{2^{i-1}a_0}{n} - \frac{1}{2n^{\frac{s}{2}}}\right),$$

and, by a simple induction,

$$\begin{aligned} a_i &\geq 2^i a_0 \prod_{j=1}^{i}\left(1 - \frac{3}{4}\frac{2^{i-j}a_0}{n} - \frac{1}{2n^{\frac{s}{2}}}\right) \\ &\geq 2^i a_0\left(1 - \frac{3}{4}\frac{2^i a_0}{n} - \frac{i}{2n^{\frac{s}{2}}}\right). \end{aligned} \tag{13.3}$$

Set $p_2 := \lceil \log_2(n) \rceil - \ell - \log_2(n_1)$ and note that for all $i < p_2$, we have that

$$a_i \leq 2^i a_0 \leq 2^{\lceil \log_2(n) \rceil - (\ell+1)} \leq \frac{n}{2^\ell} \leq \frac{n}{2},$$

since $\ell \geq 1$. Hence, for all $i < p_2$ Lemma 13.2.2 is applicable to bound the distribution of $t^{(i+1)} - t^{(i)}$ by deterministic single round plus a geometrically distributed random variable with high success probability, which we will use below. Observe that

$$a_{p_2-1} \geq \frac{n}{2^{\ell+1}}\left(1 - \frac{3}{4}\frac{1}{2^{\ell+1}} - \frac{p_2 - 1}{2n^{\frac{s}{2}}}\right) =: \bar{a}_{p_2-1}.$$

Since the lower bound of (13.3) might be decreasing for large values of $2^i a_0$, in particular when $i \geq p_2$, we now pessimistically work with the smaller lower bound of $\bar{a}_{p_2-1}$ instead of a_{p_2-1}. We set, as the target of the last subphase,

$$\begin{aligned} \bar{a}_{p_2} &:= 2\bar{a}_{p_2-1}\left(1 - \frac{3}{4}\frac{\bar{a}_{p_2-1}}{n} - \frac{1}{2n^{\frac{s}{2}}}\right) \\ &\geq \frac{n}{2^\ell}\left(1 - \frac{3}{4}\frac{1}{2^{\ell+1}} - \frac{p_2 - 1}{2n^{\frac{s}{2}}}\right) \cdot \left(1 - \frac{3}{4}\frac{1}{2^{\ell+1}} - \frac{1}{2n^{\frac{s}{2}}}\right) \\ &\geq \frac{n}{2^\ell}\left(1 - \frac{3}{4}\frac{1}{2^\ell} - \frac{p_2}{2n^{\frac{s}{2}}}\right) \geq n_2, \end{aligned}$$

and redefine $t^{(p_2)} := \min\{t \geq 0 \mid |I_{t+1}| \geq \bar{a}_{p_2}\}$. Note that Lemma 13.2.2 still applies, since $\bar{a}_{p_2}$ is a lower bound on a_{p_2}. From the above inequality, $t_2 \leq t^{(p_2)}$ follows.

Consequently, t_2 is dominated by $t_1 + \lceil \log_2(n) \rceil - \ell - \log_2(n_1) + \sum_{i=1}^{p_2} G_i$, where G_i is geometrically distributed with success probability $1 - q_i$ with $q_i \leq e^{-n^{\frac{s}{2}}/24}$ using Lemma 13.2.2. Since Lemma 13.2.1 yields $t_1 \preceq \log_2(n_1) + \mathrm{Geom}\left(1 - \frac{n_1^2}{n}\right)$, and $\sum_{i=1}^{p_2} q_i \leq$

$\log_2(n)e^{-n^{\frac{s}{2}}/24}$, we obtain, by Lemma 13.1.5,

$$\begin{aligned} t_2 &\preceq t_1 + \lceil \log_2(n) \rceil - \ell - \log_2(n_1) + \mathrm{Geom}(1 - \log_2(n)e^{-n^{\frac{s}{2}}/24}) \\ &\preceq \lceil \log_2(n) \rceil - \ell + \mathrm{Geom}(1 - (n^{-1+2s} + \log_2(n)e^{-n^{\frac{s}{2}}/24})). \end{aligned}$$ □

13.2.3 Phase 3

Once a linear number of nodes is informed, the probability for a specific uninformed node to stay uninformed is less than a constant, hence we switch our focus from the set of informed nodes to the set of uninformed nodes. Let $|U_t| \leq cn$ for some constant $0 < c < 1$. Then

$$\mathrm{E}[|U_{t+1}|] \leq |U_t| \left(1 - \frac{1}{n}\right)^{|I_t|} \leq cn \left(1 - \frac{1}{n}\right)^{(1-c)n} \leq cne^{-(1-c)}.$$

In fact, the number of uninformed nodes is concentrated in an $\mathcal{O}(n^{(1/2)+\delta})$-interval around its expectation.

Lemma 13.2.4. *For any $\delta > 0$, we have*

$$\Pr\left[|U_{t+1}| > cne^{-(1-c)} + n^{\frac{1}{2}+\delta} \,\middle|\, |U_t| \leq cn\right] \leq e^{-n^{2\delta}}.$$

Proof. Without loss of generality, assume that $U_t = \{1, \ldots, u\}$ with $u \leq cn$. We introduce $X := \sum_{i=1}^{\lfloor cn \rfloor} X_i$, where we set the indicator variables $X_i = 1$ if and only if node i is not informed by any node in I_t. Observe that X is a pessimistic estimate on $|U_{t+1}|$, i.e., $|U_{t+1}| \leq X$, and that $\mathrm{E}[X] \leq cne^{-(1-c)}$. The indicator variables are negatively associated by Lemma 13.1.3. Consequently,

$$\Pr\left[|U_{t+1}| > cne^{-(1-c)} + n^{\frac{1}{2}+\delta}\right] \leq \Pr\left[X > \mathrm{E}[X] + n^{\frac{1}{2}+\delta}\right] \leq e^{-\frac{2n^{1+2\delta}}{cn}} \leq e^{-n^{2\delta}},$$

using Lemma 13.1.4. □

We are ready to analyze Phase 3, which starts with some fraction $0 < c < 1$ of uninformed nodes and analyzes the time it takes until all these remaining nodes are informed. The lemma below shows that this time is effectively dominated by

$$\left\lceil (1 + o(1)) \frac{C_n(\lfloor cn \rfloor)}{n} + \frac{c}{1 - e^{-(1-c)}} + o(1) \right\rceil + \mathrm{Geom}(1 - o(1)),$$

and gives explicit bounds on all $o(1)$-terms.

Lemma 13.2.5 (Phase 3)**.** *Let $0 < c < 1$ and $t^{(0)} := \min\{t \geq 0 \mid |U_t| \leq cn\}$. For any $0 < \delta < 1/2$, the number of rounds until all nodes are informed is stochastically dominated by*

$$t^{(0)} + \left\lceil (1 + g(n)) \frac{C_n(\lfloor cn \rfloor)}{n} + \frac{c}{1 - e^{-(1-c)}} + h(n) \right\rceil + \mathrm{Geom}\left(1 - \frac{\ln(en)e^{-n^{2\delta}}}{1 - c}\right),$$

where for $q := e^{-(1-c)}$ *and* $Z := \left(c + \frac{1}{1-q}\right) n^{-\frac{1}{2}+\delta}$, *we define*

$$\begin{aligned} g(n) &:= \frac{Z}{1-Z} = \mathcal{O}\left(n^{-\frac{1}{2}+\delta}\right), \\ h(n) &:= \frac{n^{-\frac{1}{2}+\delta} \ln en}{(1-c)(1-q)} + g(n)\left(\frac{c}{1-q} + \frac{n^{-\frac{1}{2}+\delta} \ln en}{(1-c)(1-q)}\right) = \mathcal{O}\left(n^{-\frac{1}{2}+\delta} \ln n\right). \end{aligned}$$

Proof. As in the previous phases, for some p_3 to be chosen later, we introduce p_3 subphases almost surely consisting of single rounds each. To this end, we set

$$\varepsilon := n^{-1/2+\delta}, \qquad z_0 := c, \qquad \text{and} \qquad z_i := z_{i-1} e^{-(1-z_{i-1})} + \varepsilon.$$

Let $t^{(i)} := \min\{t > t^{(i-1)} \mid |U_t| \leq z_i n\}$ be the time that concludes the i-th subphase of Phase 3. Using the previous lemma, we immediately see that

$$\Pr[t^{(i)} - t^{(i-1)} > 1] \leq \Pr[|U_{t+1}| > z_i n \mid |U_t| \leq z_{i-1} n] \leq e^{-n^{2\delta}}.$$

Consequently, using Lemma 13.1.5, $\sum_{i=1}^{p_3}(t^{(i)} - t^{(i-1)}) \preceq p_3 + \mathrm{Geom}(1 - p_3 e^{-n^{2\delta}})$.

Observe that $z_i \leq c$ for all i and thus, with $q := e^{-(1-c)}$,

$$z_i \leq z_{i-1} q + \varepsilon \leq \cdots \leq q^i c + \sum_{j=0}^{i-1} q^j \varepsilon.$$

Clearly, the total number C_1 of messages sent in the subphases $1, \ldots, p_3$ is lower bounded by

$$\begin{aligned} C_1 \geq \sum_{i=0}^{p_3-1} n(1-z_i) &= n\left(p_3 - \sum_{i=0}^{p_3-1} z_i\right) \\ &\geq n\left(p_3 - c\sum_{j=0}^{p_3-1} q^j - \varepsilon p_3 \sum_{j=0}^{p_3-1} q^j\right) \\ &\geq n\left(p_3 - \frac{c}{1-q} - \frac{p_3\varepsilon}{1-q}\right). \end{aligned} \tag{13.4}$$

Recall that $C_n(x)$ is the random number of coupons to be drawn in the coupon collector process with n coupon types until a set of x distinguished type has been collected. Equivalently, this is the time needed to collect all coupon types given that we already start with $n - x$ distinct types. We couple the rumor spreading process and the coupon collector process by identifying informed nodes and coupons, and mimicking the choice of message receivers as coupons in the coupon collector process. We observe that if the number of messages sent after round $t^{(0)}$ is larger than $C_n(\lfloor cn \rfloor)$, the rumor spreading process is completed.

Note that when collecting all coupons (i.e., nodes receiving the rumor) in the subphases $1, 2, \ldots, p_3$, by definition it is assured that at least $|U_{t^{(0)}}| - z_{p_3} n$ of the distinguished coupon types appear among these coupons. Hence these are not independent draws from the set of all coupons. However, by Lemma 13.1.7 this condition only decreases the number of messages to be sent.

Consider the situation in round $t^{(p_3)}$ and afterwards. Setting $p_3 := \lceil \log_q \varepsilon \rceil$, the number of uninformed nodes is bounded by

$$z_{p_3} \leq cq^{\log_q \varepsilon} + \varepsilon \sum_{j=0}^{p_3-1} q^j \leq \left(c + \frac{1}{1-q}\right)\varepsilon.$$

Consequently, in each additional round $C_2 \geq (1 - z_{p_3})n = (1 - \mathcal{O}(\varepsilon))n$ messages are sent. We define the target number of coupons C^* as $C_n(\lfloor cn \rfloor)$ where we condition the underlying coupon collector process to be such that the conditions for every subphase $1 \leq i \leq p_3$, namely $|U_{t^{(i)}}| \leq z_i n$, are satisfied (see also Lemma 13.1.7). It remains to determine j such that $j \cdot C_2 + C_1 \geq C^*$, i.e., the number of messages sent in rounds $t^{(0)} + 1, \ldots, t^{(p_3)} + \lceil j \rceil$ is at least the required amount of C^* and the process is completed. By (13.4) and $C_2 \geq n(1 - z_{p_3})$, this happens if

$$jn(1 - z_{p_3}) + n\left(p_3 - \frac{c}{1-q} - \frac{p_3\varepsilon}{1-q}\right) = C^*.$$

We compute

$$\begin{aligned} j &= \frac{C^*}{n}\frac{1}{1-z_{p_3}} + \frac{1}{1-z_{p_3}}\left(\frac{c}{1-q} + \frac{\varepsilon p_3}{1-q} - p_3\right) \\ &\leq \frac{C^*}{n}\frac{1}{1-z_{p_3}} + \frac{c}{1-q} + \frac{\varepsilon p_3}{1-q} - p_3 + \frac{z_{p_3}}{1-z_{p_3}}\frac{c+\varepsilon p_3}{1-q} \\ &= \frac{C^*}{n}\frac{1}{1-z_{p_3}} + \frac{c}{1-q} - p_3 + h_c(\varepsilon), \end{aligned}$$

where $h_c(\varepsilon) := \frac{\varepsilon p_3}{1-q} + \frac{z_{p_3}}{1-z_{p_3}}\frac{c+\varepsilon p_3}{1-q} = \mathcal{O}(\varepsilon \cdot p_3)$. Note that the second line follows from $(1 - z_{p_3})^{-1} = 1 + z_{p_3}/(1 - z_{p_3})$ and $p_3 \geq 0$.

By repeated application of Lemma 13.1.7, we can replace C^* by the independent, unconditioned random variable $C_n(\lfloor cn \rfloor)$ and conclude that the total number of rounds is stochastically dominated by

$$\begin{aligned} & t^{(0)} + \sum_{i=1}^{p_3}(t^{(i)} - t^{(i-1)}) + \lceil j \rceil \\ &\preceq\ t^{(0)} + \mathrm{Geom}(1 - p_3 e^{-n^{2\delta}}) + \lceil p_3 + j \rceil \\ &\preceq\ t^{(0)} + \mathrm{Geom}(1 - p_3 e^{-n^{2\delta}}) + \left\lceil \frac{C_n(\lfloor cn \rfloor)}{n}\frac{1}{1-z_{p_3}} + \frac{c}{1-q} + h_c(\varepsilon) \right\rceil, \end{aligned}$$

where $1 + g(n) \geq \frac{1}{1-z_{p_3}}$ and $h(n) \geq h_c(\varepsilon) = h_c(n^{-1/2+\delta})$, using that $p_3 = \lceil \ln(n^{1/2+\delta})/(1-c) \rceil \leq \ln(en)/(1-c)$ and $z_{p_3} \leq (c + (1-q)^{-1})\varepsilon$. □

13.2.4 Connecting the Phases

We can finally prove the main theorem, combing the analysis of all phases. Specifically, we instantiate the Phase 2 Lemma (Lemma 13.2.3) with $\ell = 1$, i.e., we use the analysis of Phases 1 and 2 to obtain a fraction of roughly $\frac{5}{16}$ informed nodes and use the analysis of Phase 3 thereafter.

Theorem 13.2.6. *Let $s < 1/2$ with $n^{\frac{s}{2}} \geq (3/2)\log_2 n$. The number of rounds until the rumor spreading process on the complete graph with n nodes informs all nodes is*

stochastically dominated by

$$\lceil \log_2 n \rceil + \left\lceil (1+g(n)) \frac{C_n\left(\lfloor \left(\frac{11}{16}+\varepsilon\right) n \rfloor\right)}{n} + 1.562 + 10.71\varepsilon + \mathcal{O}(\varepsilon^2) + h(n) \right\rceil + \mathrm{Geom}\left(1 - \left(\frac{1}{n^{1-2s}} + \frac{\log_2(n)}{\exp(n^{\frac{s}{2}}/24)} + \frac{4\ln(en)}{\exp(n^s)}\right)\right),$$

where we choose $\delta = s/2$, $c = \frac{11}{16} + \varepsilon$ *and define* $\varepsilon := \frac{\log_2(n)}{2n^{\frac{s}{2}}}$, $g(n) = \mathcal{O}(n^{-(1+s)/2})$ *and* $h(n) = \mathcal{O}(n^{-(1+s)/2} \ln n)$ *as in Lemma 13.2.5.*

Proof. By Lemma 13.2.3 (with $\ell := 1$), we have that for $\bar{c} := \frac{5}{16} - \varepsilon$, the time until at least $\lceil \bar{c}n \rceil$ nodes are informed is stochastically dominated by $\lceil \log_2 n \rceil - 1 + \mathrm{Geom}(1 - (n^{-1+2s} + \log_2(n) e^{-n^{\frac{s}{2}}/24}))$. At this point, we have a fraction of at most $c := 1 - \bar{c} = \frac{11}{16} + \varepsilon$ uninformed nodes. Define $F(x) := \frac{x}{1-e^{-(1-x)}}$ and note that

$$F(c) \leq F\left(\frac{11}{16}\right) + F'\left(\frac{11}{16}\right)\varepsilon + \mathcal{O}(\varepsilon^2) \leq 2.562 + 10.71\varepsilon + \mathcal{O}(\varepsilon^2). \tag{13.5}$$

With $\delta = \frac{s}{2}$, Lemma 13.2.5 yields that the total number of rounds is stochastically dominated by

$$\lceil \log_2 n \rceil - 1 + \left\lceil (1+g(n)) \frac{C_n(\lfloor cn \rfloor)}{n} + F(c) + h(n) \right\rceil + \mathrm{Geom}\left(1 - \left(n^{-1+2s} + \log_2(n) e^{-n^{\frac{s}{2}}/24}\right)\right) + \mathrm{Geom}\left(1 - \frac{\ln(en)}{(1-c)} e^{-n^s}\right).$$

Plugging in the value of c and using (13.5), $\ln(en)(1-c)^{-1} e^{-n^s} \leq 4\ln(en) e^{-n^s}$, as well as Lemma 13.1.5 concludes the proof. □

Using that $\mathrm{E}[C_n(\lfloor cn \rfloor)/n] \leq H_{\lfloor cn \rfloor} \leq \ln(n) + \ln(c) + \gamma + \mathcal{O}(1/n)$, we immediately derive the following statement, since Theorem 13.2.6 bounds the expected number of total rounds by

$$\lceil \log_2 n \rceil + (\ln(n) + \ln(11/16) + \gamma + 1.562) + 1 + o(1).$$

Corollary 13.2.7. *The expected number of rounds until the rumor spreading process on the complete graph with n nodes informs all nodes is bounded by*

$$\lceil \log_2 n \rceil + \ln n + 2.765 + o(1).$$

Corollary 13.2.8. *Let $r \geq 0$ be constant and n be sufficiently large. The probability that the rumor spreading process on the complete graph with n nodes does not inform all nodes in $\lceil \log_2(n) \rceil + \ln(n) + 2.188 + r$ rounds is bounded by $2e^{-r}$.*

Proof. Theorem 13.2.6 confines deviations from the expected value to the deviations of the coupon collector process $C_n(m)$ with $m = \lfloor (\frac{11}{16} + o(1))n \rfloor$ and a geometric random variable $X \sim \mathrm{Geom}(1 - \mathcal{O}(n^{-1+\varepsilon}))$ for some arbitrarily small $\varepsilon > 0$. By Lemma 13.1.6, we have that

$$\Pr\left[\frac{C_n(m)}{n} > \ln(m) + r\right] \leq e^{-r}.$$

For the geometric random variable X, there is a constant C such that for sufficiently large n, we have, with $\alpha := 1/\ln(n^{1-\varepsilon}/C)$,

$$\Pr[X > \alpha r] \leq \left(\frac{C}{n^{1-\varepsilon}}\right)^{\alpha r} \leq e^{-r}.$$

Consequently, with probability at most $2e^{-r}$, it holds that

$$\begin{aligned} S_n &\leq \lceil \log_2(n) \rceil + \lceil (1 + g(n))(\ln(m) + r) + 1.562 + o(1) \rceil + r\alpha \\ &\leq \lceil \log_2(n) \rceil + \lceil \ln(n) + \ln(11/16) + r + 1.562 + o(1) \rceil + r\alpha \\ &\leq \lceil \log_2(n) \rceil + \ln(n) + r + 2.188, \end{aligned}$$

for sufficiently large n. □

13.3 Lower Bounds

In this section, we give two coarse lower bounds, which still show that our upper bounds derived in the previous section are relatively sharp. Stronger lower bounds, giving a better additive constant, could be proven with similar arguments as in the previous section, in particular, by giving upper bounds on the expected number of informed vertices together with large deviation bounds.

Theorem 13.3.1. *The number of rounds until the rumor spreading process on the complete graph with n nodes informs all nodes is stochastically subdominated by*

$$\lfloor \log_2 n \rfloor - 1 + \left\lceil \frac{C_n(\lceil n/2 \rceil)}{n} \right\rceil.$$

Proof. The number of informed nodes can at most double per round, hence it is guaranteed that after $\lfloor \log_2(n/2) \rfloor$ rounds at least $\lceil n/2 \rceil$ vertices remain uninformed. Coupling the rumor spreading process with a coupon collector, $C_n(\lceil n/2 \rceil)$ describes the number of coupons to be drawn until the last $\lceil n/2 \rceil$ of the remaining coupon types are collected. Since at most n coupons are collected per round, $\lceil C_n(\lceil n/2 \rceil)/n \rceil$ is a lower bound on the number of rounds required to inform the last $\lceil n/2 \rceil$ nodes. Consequently, the number of rounds required to inform all nodes is stochastically subdominated by $\lfloor \log_2 n \rfloor - 1 + \lceil \frac{C_n(\lceil n/2 \rceil)}{n} \rceil$. □

Corollary 13.3.2. *The expected number of rounds until the rumor spreading process on the complete graph with n nodes informs all nodes is at least*

$$\lfloor \log_2 n \rfloor + \ln(n) - 1.116.$$

Proof. Using $H_x \geq \ln(x) + \gamma$, we compute

$$\mathrm{E}\left[\frac{C_n(\lceil n/2 \rceil)}{n}\right] \geq \ln(n) + \gamma - \ln(2) \geq \ln(n) - 0.116,$$

and the claim follows from the previous theorem. □

Chapter 14

The 2-Player Cryptogenography Problem

In this chapter, we analyze the 2-player cryptogenography problem and improve upon previous upper and lower bounds on the optimal success probability (see Chapter 12 for an overview of the results). Section 14.1 introduces the problem, describes our approach to obtain strong cryptogenographic protocols and proves new upper bounds. Section 14.2 revisits the concavity method used in the previous work and provides a stronger lower bound. Section 14.3 concludes this chapter with a short discussion and outlook.

14.1 Finding Better Cryptogenography Protocols

This section is devoted to the design of stronger cryptogenographic protocols. In particular, we demonstrate that a success probability of more than 1/3 can be achieved. We start by making the cryptogenography problem precise (Sections 14.1.1 and 14.1.2) and introduce an equivalent formulation as solitaire vector splitting game (Section 14.1.3). We illustrate both formulations using the best known protocol for the 2-player case (Section 14.1.4). In Section 14.1.5, we state basic properties that simplify the analysis of protocols and aid our automated search for better protocols, which is detailed in Section 14.1.7. In Section 14.1.6, we give a simple, human-readable proof that 1/3 is not the optimal success probability by analyzing a protocol with success probability $\frac{449}{1334} \approx 0.3341$. We describe how to post-optimize and certify the results obtained by the automated search using linear programming in Section 14.1.8 and summarize our findings (in particular, the best lower bound we have found) in Section 14.1.9.

14.1.1 The Cryptogenography Problem

Let us fix an arbitrary number k of players called $1, \dots, k$ for simplicity. We write $[k] := \{1, \dots, k\}$ for the set of players. We assume that a random one of the them, the "secret owner" $J \in [k]$, initially has a secret, namely a random bit $X \in \{0, 1\}$. The task of the players is, using public communication only, to make this random bit public without revealing the identity of the secret owner. More precisely, we assume that the players, before looking at the secret distribution, (publicly) decide on a communication protocol π. This is again public, that is, all bits sent are broadcast to all players, and they may depend only on previous communication, the private knowledge of the sender (whether he is the secret owner or not, and if so, the secret), and private random numbers of the sender. At the end of the communication phase, the protocol specifies an output bit Y (depending on all communication).

The aspect of not disclosing the identity of the secret owner is modeled by an adversary, who knows the protocol (because it was discussed in public) and who gets

to see all communication (and consequently also knows the protocol output Y). The adversary, based on all this data, blames one player K. The players win this game if the protocol outputs the true secret (that is, $Y = X$) and the adversary does not blame the secret owner (that is, $K \neq J$), otherwise the adversary wins. It is easy to see (see Section 14.1.2) what the best strategy for the adversary is (given the protocol and the communication), so the interesting part of the cryptogenography problem is finding strategies that maximize the probability that the players win assuming that the adversary plays optimally. We call this the (players') success probability of the protocol.

While the game starts with a uniform secret distribution, it will be useful to regard arbitrary secret distributions. In general, a *secret distribution* is a distribution D over $\{0,1\} \times [k]$, where D_{ij} is the probability that player $j \in [k]$ is the secret owner and the secret is $i \in \{0,1\}$. Modulo a trivial isomorphism, D is just a vector in $\mathbb{R}^{2\times k}_{\geq 0}$ with $\|D\|_1 = 1$. We denote by $\Delta = \Delta_k$ the set of all these distributions (this was denoted by $\Delta(\{0,1\} \times [k])$ in [Bro+14]).

Brody et al. [Bro+14] observe that any cryptogenographic protocol can be viewed as successive rounds of one-bit communication, where in each step some (a priori) secret distribution probabilistically leads to one of two follow-up (a posteriori) distributions (depending on the bit transmitted) such that the a priori distribution is a convex combination of these and a certain proportionality condition is fulfilled (all three distributions lie in the same "allowed plane"). Conversely, whenever the initial distribution can be written as such a convex combination of certain distributions, then there is a round of a cryptogenographic protocol leading to these two distributions (with certain probabilities). Consequently, the problem of finding a good cryptogenographic protocol is equivalent to iteratively rewriting the initial equidistribution as certain convex combinations of other secret distributions in such a way that the success probability, which can be expressed in terms of this rewriting tree, is large. Instead of directly working with this formulation, we propose a slightly different reformulation in Section 14.1.3. To prepare readers that are unfamiliar with the work of Brody et al. [Bro+14], we give a high-level introduction in the following section.

14.1.2 The Convex Combination Formulation

For readers' convenience, we give a high-level description of the convex combination formulation of Brody et al. For proofs and a more formal treatment, we refer the reader to [Bro+14].

Optimal strategy of the adversary. Recall that X denotes the secret bit and J the identity of the secret owner. Fix a protocol π of the players, which for every state of the protocol execution, i.e., every possible history of communication, determines (1) which player's turn it is to communicate (or whether communication has ended) and (2) probability distributions over the next message this player sends (two for the case that this player is the secret owner, i.e., one for each value of the secret bit, and one for the other case). Thus, both (1) and (2) may depend on all previous communication, and (2) possibly depends on the value of the secret bit, namely if the active player is the secret owner. Additionally, π fixes a protocol output function $\textsc{Out}$ that given the transcript τ of all communication returns the players' guess $\textsc{Out}(\tau)$ on the secret bit. Without loss of generality, we may assume that the protocol proceeds in rounds, where in each round a message consisting of a single bit is sent.

Let $\textsc{Com}(\pi)$ denote the transcript of all communication of the protocol π. Note that this is a random variable, since we assume a random player to be the owner

of a random secret bit. It is not difficult to see what the optimal strategy of the adversary is, given the knowledge of the protocol π. He may assume that the players' guess is correct, i.e., $\textsc{Out}(\textsc{Com}(\pi)) = X$, as otherwise the players have already lost and therefore his guess is irrelevant. After the protocol execution has finished with a transcript τ, the adversary maximizes his winning probability by blaming the player $\operatorname{argmax}_{j\in[k]} \Pr[J = j \mid \textsc{Com}(\pi) = \tau, X = \textsc{Out}(\tau)]$ (breaking ties arbitrarily), i.e., the player who is most likely to own the secret given the communication described by τ.

From this reasoning, it becomes clear that the decisive information in the game is, for any partial transcript τ', the distribution $D = ((D_{0,1}, D_{1,1}), \ldots, (D_{0,k}, D_{1,k}))$, where

$$D_{x,j} = \Pr[J = j, X = x \mid \textsc{Com}(\pi) \text{ starts with } \tau']$$

is the probability an outside observer (knowing only public information, i.e., all previous communication) assigns to the event $(J = j, X = x)$. As a simple consequence, assume that some fixed transcript τ' transforms the initial uniform distribution into the distribution D and no further communication is allowed. Then the optimal choice for the protocol output is the guess

$$\operatorname*{argmax}_{x\in\{0,1\}} \left(\sum_{j\in[k]} D_{x,j} - \max_{j\in[k]} D_{x,j} \right),$$

as for any fixed choice $\textsc{Out}(\tau') = x$, the adversary blames the player $\bar{\jmath} := \operatorname{argmax}_{j\in[k]} D_{x,j}$ and hence the players win in all cases with $X = x$ except when $J = \bar{\jmath}$. We call the strategy applied here the *zero-bit strategy* (since no further communication is done).

Convex combination formulation. Brody et al. prove that it is not only sufficient, but in fact equivalent to represent the cryptogenography game using only the distributions D described above and how the protocol affects these distributions. More precisely, one can model the game starting from any initial distribution D on $\{0,1\} \times \{1,\ldots,k\}$. Then the first bit sent by some player j *splits* D into the distributions D^0 (for the case that the 0-bit is sent) and D^1 (for the case that the 1-bit is sent), i.e., D^i is the distribution an outside observer assigns to (J, X) after bit i has been sent. By this abstraction, one can recursively consider the distributions D^0 and D^1 (i.e., their optimal protocols and success probabilities).

To determine the properties of possible splits in a protocol, let p be the probability that player j transmits a 0-bit. By a simple calculation, we have that $D = pD^0 + (1-p)D^1$ (cf. [Bro+14, Lemma 4.1]). Additionally, since a player may only use the information whether or not he has the secret bit (and if so, the value of the secret bit), player j may never leak new information about whether another player $j' \in [k] \setminus \{j\}$ is more likely to have secret 0 or 1 (i.e., the ratio of $D_{0,j'}$ and $D_{1,j'}$ is maintained in the resulting distributions D^0 and D^1) or whether player $j' \neq j$ is more likely to have the secret than another player $j'' \in [k] \setminus \{j, j'\}$. This transfers to a proportionality condition that $(D^0)_{|\{0,1\}\times([k]\setminus\{j\})} = \lambda D_{|\{0,1\}\times([k]\setminus\{j\})}$ for some $\lambda \in [0,1]$. In fact, any split of D into D^0 and D^1 satisfying these conditions can be realized by a cryptogenographic protocol. Thus, the cryptogenography game is equivalent to, starting from the uniform distribution, recursively apply splits satisfying these conditions (i.e., *allowed splits*), using the zero-bit strategy at the leaves, in such a way that the resulting success probability is maximized. We argue that this view is equivalent to our vector splitting formulation (that will be introduced in the following section) in Lemma 14.1.6.

14.1.3 The Solitaire Vector Splitting Game

Instead of directly using the "convex combination" formulation of Brody et al., we propose a slightly different reformulation as *solitaire vector splitting game*. This formulation seems to ease finding good cryptogenographic protocols (lower bounds for the success probability), both for human researchers and via automated search (Section 14.1.6). The main advantage of our formulation is that it takes as positions all $2k$-dimensional vectors with non-negative entries, whereas the cryptogenographic protocols are only defined on distributions over $\{0,1\} \times [k]$. In this way, we avoid arguing about probabilities and convex combinations and instead simply split a vector (resembling a secret distribution) into a sum of two other vectors. Furthermore, a simple monotonicity property (Proposition 14.1.5) eases the analyses. Still, there is an easy translation between the two formulations, so that we can re-use whatever results were found in [Bro+14].

Definition 14.1.1. *Let $D \in \mathbb{R}^{2\times k}_{\geq 0}$. We say that (D_0, D_1) is a j-*allowed split of D *if $D = D_0 + D_1$ and D_0 (and thus also D_1) is proportional to D on $\{0,1\} \times ([k] \setminus \{j\})$, that is, there is a $\lambda \in [0,1]$ such that $(D_0)_{|\{0,1\}\times([k]\setminus\{j\})} = \lambda D_{|\{0,1\}\times([k]\setminus\{j\})}$. We call (D_0, D_1) an* allowed split *of D if it is a j-allowed split of D for some $j \in [k]$.*

The objective of the vector splitting game is to recursively apply allowed splits to a given vector $D \in \mathbb{R}^{2\times k}_{\geq 0}$ with the target of maximizing the sum of the

$$p(D') := \max_{x\in\{0,1\}} \left(\sum_{j\in[k]} D'_{x,j} - \max_{j\in[k]} D'_{x,j} \right)$$

values of the resulting vectors (note that when D' is a distribution, then $p(D')$ is the 0-bit success probability of D' as argued in Section 14.1.2). More precisely, an n-round play of the vector splitting game is described by a binary tree of height at most n, where the nodes are labeled with *game positions* in $\mathbb{R}^{2\times k}_{\geq 0}$. The root is labeled with the initial position D. For each non-leaf v, the labels of the two children form an allowed split of the label of v. The payoff of such a play is $\sum_{D'} p(D')$, where D' runs over all leaves of the game tree. The aim is to maximize the payoff. Right from this definition, it is clear that the maximum payoff achievable in an n-round game started in position D, the *value* of this game, is $\textsc{succ}_n(D)$ as defined below.

Definition 14.1.2. *For all $n \in \mathbb{N}$ and for all $D \in \mathbb{R}^{2\times k}_{\geq 0}$, we recursively define*

(i) $\textsc{succ}_0(D) := \max_{x\in\{0,1\}} \left(\sum_{j\in[k]} D_{x,j} - \max_{j\in[k]} D_{x,j} \right)$;

(ii) $\textsc{succ}_n(D) := \max_{(D_0,D_1)} \left(\textsc{succ}_{n-1}(D_0) + \textsc{succ}_{n-1}(D_1) \right)$, *if $n \geq 1$. Here the maximum is taken over all allowed splits (D_0, D_1) of D.*

For an example of an admissible game, we refer to Figure 14.1 in Section 14.1.4.

It is easy to see that the game values are non-decreasing in the number of rounds, but bounded. The limiting value is thus well defined.

Lemma 14.1.3. *Let $D \in \mathbb{R}^{2\times k}_{\geq 0}$ and $n \in \mathbb{N}$. Then $\textsc{succ}_n(D) \leq \|D\|_1$ and $\textsc{succ}_{n+1}(D) \geq \textsc{succ}_n(D)$. Consequently, $\textsc{succ}(D) := \lim_{n\to\infty} \textsc{succ}_n(D)$ is well defined and is equal to $\sup_{n\in\mathbb{N}} \textsc{succ}_n(D)$.*

Proof. The previous definition and an elementary induction shows $\textsc{succ}_n(D) \leq \|D\|_1$. Since $(D, 0)$ is an allowed split of D and $\textsc{succ}_n(0) = 0$ by the previous observation, we have $\textsc{succ}_{n+1}(D) \geq \textsc{succ}_n(D) + \textsc{succ}_n(0) = \textsc{succ}_n(D)$. □

Proposition 14.1.4 (scalability)**.** *Let $D \in \mathbb{R}^{2\times k}_{\geq 0}$ and $\lambda \geq 0$. Then $\textsc{succ}_n(\lambda D) = \lambda\,\textsc{succ}_n(D)$ for all $n \in \mathbb{N}$. Consequently, $\textsc{succ}(\bar{\lambda}D) = \lambda\,\textsc{succ}(D)$.*

Proof. The statements follow right from the definition of $\textsc{succ}_n$ and $\textsc{succ}$ via induction. □

Proposition 14.1.5 (monotonicity)**.** *Let $D, E \in \mathbb{R}^{2\times k}_{\geq 0}$ with $E \geq D$ (component-wise). Then $\textsc{succ}_n(E) \geq \textsc{succ}_n(D)$ for all $n \in \mathbb{N}$. Consequently, $\textsc{succ}(E) \geq \textsc{succ}(D)$.*

Proof. Clearly $\textsc{succ}_0(E) \geq \textsc{succ}_0(D)$. Hence assume that for some $n \in \mathbb{N}$, we have $\textsc{succ}_n(E) \geq \textsc{succ}_n(D)$ for all $E \geq D$. Let (D^0, D^1) with $D^0 = (d^0_{ij})$ and $D^1 = D - D^0 = (d^1_{ij})$ be an allowed split of $D = (d_{ij})$. Building on these, define $E^0 := E^0(D^0, D) = (e^0_{ij})$ with $e^0_{ij} := d^0_{ij}\frac{e_{ij}}{d_{ij}}$ and $E^1 := E^1(D^1, D) := E - E^0$, hence $e^1_{ij} = d^1_{ij}\frac{e_{ij}}{d_{ij}}$. Then (E^0, E^1) is an allowed split of E satisfying $E^0 \geq D^0$ and $E^1 \geq D^1$. Hence

$$\begin{aligned}\textsc{succ}_{n+1}(E) &\geq \max_{(D^0,D^1)} (\textsc{succ}_n(E^0) + \textsc{succ}_n(E^1)) \\ &\geq \max_{(D^0,D^1)} (\textsc{succ}_n(D^0) + \textsc{succ}_n(D^1)) = \textsc{succ}_{n+1}(D),\end{aligned}$$

where the maxima range over all allowed splits (D^0, D^1) of D. □

From the previous definitions and observations, we derive that the game values for games starting with a distribution D, that is, $\|D\|_1 = 1$, and the success probabilities of the optimal cryptogenographic protocols for D, are equal.

Lemma 14.1.6. *Let $D \in \mathbb{R}^{2\times k}_{\geq 0}$ with $\|D\|_1 = 1$. Then for all $n \in \mathbb{N}$, our definitions of $\textsc{succ}_n$ coincide with the ones of Brody et al., which are the success probabilities of the best n-round cryptogenographic protocols for the distribution D. Consequently, also the definition of $\textsc{succ}(D)$ coincides.*

Proof. Let us for the moment denote the success probabilities defined by Brody et al. by $s_n(D)$ and $s(D)$ and then show that $\textsc{succ}_n(D) = s_n(D)$ and consequently $\textsc{succ}(D) = s(D)$ for all distributions D.

By definition, we have $\textsc{succ}_0(D) = s_0(D)$ for all distributions D. Lemmas 4.1 and 4.2 of [Bro+14] establish that the first round of any cryptogenographic protocol for the distribution D with some probability λ leads to a distribution D^0 and with probability $\bar{\lambda} := 1-\lambda$ leads to a position D^1 such that $D = \lambda D^0 + \bar{\lambda} D^1$ and D^0, D^1 are proportional to D on $\{0,1\} \times ([k] \setminus \{j\})$ for some $j \in [k]$. Conversely, for any such λ, D^0, D^1 there is a one-round cryptogenographic protocol leading to the distribution D^0 with probability λ and to D^1 with probability $\bar{\lambda}$. Hence for any $n \geq 1$, the success probability $s_n(D)$ of the optimal n-round protocol for the distribution D is $s_n(D) = \max_{\lambda, D^0, D^1}(\lambda s_{n-1}(D^0) + \bar{\lambda} s_{n-1}(D^1))$, where λ, D^0, D^1 run over all values as above. Note that these are exactly those values which make $(\lambda D^0, \bar{\lambda} D^1)$ an allowed split of D. By induction and scalability, we obtain

$$\begin{aligned}s_n(D) &= \max_{\lambda, D^0, D^1} (\lambda\,\textsc{succ}_{n-1}(D^0) + \bar{\lambda}\,\textsc{succ}_{n-1}(D^1)) \\ &= \max_{\lambda, D^0, D^1} (\textsc{succ}_{n-1}(\lambda D^0) + \textsc{succ}_{n-1}(\bar{\lambda} D^1)) \\ &= \max_{(\bar{D}^0, \bar{D}^1)} (\textsc{succ}_{n-1}(\bar{D}^0) + \textsc{succ}_{n-1}(\bar{D}^1)) = \textsc{succ}_n(D),\end{aligned}$$

where the last maximum is taken over all allowed splits $(\bar{D}^0, \bar{D}^1)$ of D. □

14.1.4 Example: The Best-so-far 2-Player Protocol

We now turn to the case of two players. We use this section to describe the best known protocol for two players in the different languages. We also use this mini-example to sketch the approaches used in the following sections to design superior protocols.

For two players, we usually write a game position $D = (D_{01}, D_{11}, D_{02}, D_{12}) \in \mathbb{R}^{2\times 2}_{\geq 0}$ as $D = (a, b, c, d)$. The 0-round game value (equaling the success probability of the 0-bit protocol) then is

$$\text{SUCC}_0(D) = \max\{\min\{a, c\}, \min\{b, d\}\}.$$

As a warmup, let us describe the best known 2-player protocol TwoBit in the two languages. In the language of Brody et al., the first player can send a (randomized) bit that transforms the initial distribution $(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$ with probability $\frac{1}{2}$ each into the distributions $(\frac{1}{3}, \frac{1}{3}, \frac{1}{6}, \frac{1}{6})$ and $(\frac{1}{6}, \frac{1}{6}, \frac{1}{3}, \frac{1}{3})$. In the first case, the second player can send a bit leading to each of the distributions $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0)$ and $(\frac{1}{3}, \frac{1}{3}, 0, \frac{1}{3})$ with probability $\frac{1}{2}$, both having a 0-bit success probability of $\frac{1}{3}$. In the second possible result of the first move, the first player can lead to an analogous situation. Consequently, after two rounds of the protocol we end up with four equally likely distributions all having a 0-bit success probability of $\frac{1}{3}$. Hence the protocol TwoBit has a success probability of $\frac{1}{3}$.

In the language of the splitting games, we can forget about the probabilities and simply split up the initial distribution. Using the scaling invariance, to ease reading we scaled up all numbers by a factor of 12. Figure 14.1 shows the game tree corresponding to the TwoBit protocol. It shows that $\text{SUCC}_2(3, 3, 3, 3) \geq 4$, proving again the existence of a cryptogenographic protocol for the distribution $(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}) = \frac{1}{12}(3, 3, 3, 3)$ with success probability $\frac{4}{12} = \frac{1}{3}$.

Note that each allowed split (D^0, D^1) of D implies the corresponding inequality $\text{SUCC}(D) \geq \text{SUCC}(D^0) + \text{SUCC}(D^1)$, which follows from clause (ii) of Definition 14.1.2 and taking the limit $n \to \infty$. Hence the game tree giving the $\frac{1}{3}$ lower bound for the success probability equivalently gives the following proof via inequalities.

$$\begin{aligned}
\text{SUCC}(3, 3, 3, 3) &\geq \text{SUCC}(2, 2, 1, 1) + \text{SUCC}(1, 1, 2, 2),\\
\text{SUCC}(2, 2, 1, 1) = \text{SUCC}(1, 1, 2, 2) &\geq \text{SUCC}(1, 0, 1, 1) + \text{SUCC}(0, 1, 1, 1),\\
\text{SUCC}(1, 0, 1, 1) = \text{SUCC}(0, 1, 1, 1) &\geq \text{SUCC}_0(0, 1, 1, 1) = 1.
\end{aligned}$$

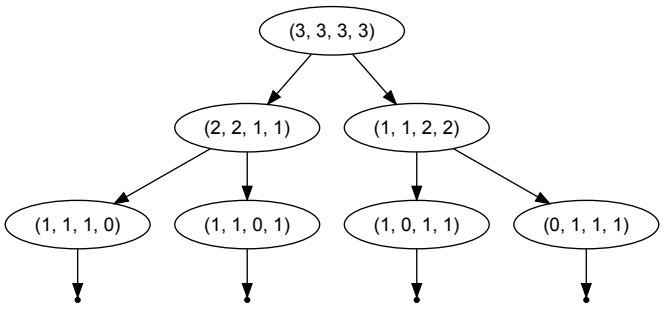


FIGURE 14.1: Game tree corresponding to TwoBit.

The splitting game and the inequality view will in the following be used to design stronger protocols (better lower bounds for the optimal success probability). We shall compute good game trees by computing lower bounds for the game values of a discrete

set of positions via repeatedly trying allowed splits. For example, the above game tree for the starting position $(3,3,3,3)$ could have easily be found by recursively computing the game values for all positions in $\{0,1,2,3\}^4$.

It turns out that such an automated search leads to better results when we also allow scaling moves (referring to Proposition 14.1.4). For example, in the above mini-example of computing optimal game values for all positions $\{0,1,2,3\}^4$, we could try to exploit the fact that $\textsc{succ}(1,1,1,1) = \frac{1}{3}\textsc{succ}(3,3,3,3)$. Such scaling moves are a cheap way of working in $\{0,1,2,3\}^4$, but trying to gain the power of working in $\{0,1,\ldots,9\}^4$, which is computationally more costly, especially with regard to memory usage. Scaling moves may lead to repeated visits of the same position, resulting in cyclic structures. Here translating the allowed splits used in the tree into the inequality formulation and then using an LP-solver is an interesting approach (detailed in Section 14.1.8). It allows to post-optimize the game trees found, in particular, by solving cyclic dependencies. This leads to slightly better game values and compacter representations of game trees.

14.1.5 Useful Facts

For some positions of the vector splitting game, the true value is easy to determine. We do this here to later ease the presentation of the protocols.

Proposition 14.1.7. *We have* $\textsc{succ}(a,b,c,d) \le \min\{a,c\} + \min\{b,d\}$.

This statement is a simple corollary of the concavity method detailed in Section 14.2.1, hence at this point, we only state the proposition and postpone the proof.

Proposition 14.1.8. *Let* $D = (a,b,c,0)$. *Then* $\textsc{succ}(D) = \textsc{succ}_0(D) = \min\{a,c\}$.

Proof. Clearly, $\textsc{succ}(D) \ge \textsc{succ}_0(D) = \min\{a,c\}$. By Proposition 14.1.7, we obtain $\textsc{succ}(D) \le \min\{a,c\}$, proving the claim. □

Proposition 14.1.9. *If* $D = (a,b,c,d)$ *is such that* $a+b \le \min\{c,d\}$, *then* $\textsc{succ}(D) = a+b$.

Proof. We have $D \ge D' := (a,b,a+b,a+b)$ and (D^0, D^1) with $D^0 = (a,0,a,a)$ and $D^1 = (0,b,b,b)$ is an allowed split of D'. Hence $\textsc{succ}(D) \ge \textsc{succ}(D') \ge \textsc{succ}(D^0) + \textsc{succ}(D^1) = a+b$. The upper bound follows immediately from Proposition 14.1.7. □

14.1.6 Small Protocols Beating the $1/3$ Barrier

We now present a sequence of protocols showing that there are cryptogenographic protocols having a success probability strictly larger than $\frac{1}{3}$. These protocols are still relatively simple, so we also obtain a human-readable proof of the following result.

Theorem 14.1.10 (Stronger Protocols I)**.** *For the 2-player cryptogenography probem, we have* $\textsc{succ}(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4}) \ge \frac{449}{1334} \approx 0.3341$.

Proof. To be able to give a readable mathematical proof, we argue via inequalities for game values $\textsc{succ}(\cdot)$. We later discuss how the corresponding protocols (game trees) look like.

We first observe the following inequalities, always stemming from allowed splits (the underlined entries are proportional). Whenever Proposition 14.1.8 or 14.1.9 determine a value, we exploit this without further notice.

$$\begin{aligned} \textsc{succ}(\underline{12,12},12,12) &\ge \textsc{succ}(\underline{7,7},6,4) + \textsc{succ}(\underline{5,5},6,8), \\ \textsc{succ}(\underline{5,5},6,8) &\ge \textsc{succ}(\underline{2,2},0,2) + \textsc{succ}(\underline{3,3},6,6) = 2+6 = 8. \end{aligned}$$

This proves $\textsc{succ}(12,12,12,12) \geq 8 + \textsc{succ}(7,7,6,4)$. To analyze $\textsc{succ}(7,7,6,4)$, we use the allowed split

$$\textsc{succ}(7,7,\underline{6,4}) \quad \geq \quad \textsc{succ}(4,5,\underline{3,2}) + \textsc{succ}(3,2,\underline{3,2}) \tag{14.1}$$

and regard the two positions $(4,5,3,2)$ and $(3,2,3,2)$ separately in some detail.

Claim 1: The value of $(4,5,3,2)$ *satisfies* $\textsc{succ}(4,5,3,2) \geq \frac{55}{12}$. By scaling, we have $\textsc{succ}(4,5,3,2) = \frac{1}{2}\,\textsc{succ}(8,10,6,4)$. We present the allowed splits

$$\begin{aligned} \textsc{succ}(\underline{8,10},6,4) &\geq \textsc{succ}(\underline{4,5},2,4) + \textsc{succ}(\underline{4,5},4,0) = \textsc{succ}(4,5,2,4) + 4, \\ \textsc{succ}(4,5,\underline{2,4}) &\geq \textsc{succ}(1,2,\underline{1,2}) + \textsc{succ}(3,3,\underline{1,2}) = \textsc{succ}(1,2,1,2) + 3, \end{aligned}$$

hence $\textsc{succ}(8,10,6,4) \geq \textsc{succ}(1,2,1,2) + 7$. To bound the latter term, we use the scaling $\textsc{succ}(1,2,1,2) = \frac{1}{6}\,\textsc{succ}(6,12,6,12)$ and consider the allowed splits

$$\begin{aligned} \textsc{succ}(\underline{6,12},6,12) &\geq \textsc{succ}(\underline{5,10},3,9) + \textsc{succ}(\underline{1,2},3,3) = \textsc{succ}(5,10,3,9) + 3, \\ \textsc{succ}(5,10,\underline{3,9}) &\geq \textsc{succ}(0,6,\underline{2,6}) + \textsc{succ}(5,4,\underline{1,3}) = 6 + 4 = 10. \end{aligned}$$

Thus $\textsc{succ}(6,12,6,12) \geq 13$ and $\textsc{succ}(1,2,1,2) \geq \frac{13}{6}$. This shows $\textsc{succ}(4,5,3,2) \geq \frac{1}{2}\,(\textsc{succ}(1,2,1,2) + 7) \geq \frac{55}{12}$.

Claim 2: We have $\textsc{succ}(3,2,3,2) \geq \frac{5}{3} + \frac{2}{9}\,\textsc{succ}(7,7,6,4)$. By scaling, we obtain $\textsc{succ}(3,2,3,2) = \frac{1}{3}\,\textsc{succ}(9,6,9,6)$ and compute

$$\begin{aligned} \textsc{succ}(9,6,\underline{9,6}) &\geq \textsc{succ}(6,3,\underline{6,4}) + \textsc{succ}(3,3,\underline{3,2}), \\ \textsc{succ}(6,3,\underline{6,4}) &\geq \textsc{succ}(3,0,\underline{3,2}) + \textsc{succ}(3,3,\underline{3,2}) = 3 + \textsc{succ}(3,3,3,2), \end{aligned}$$

and hence $\textsc{succ}(9,6,9,6) \geq 3 + 2\,\textsc{succ}(3,3,3,2)$. To bound the latter term, we scale $\textsc{succ}(3,3,3,2) = \frac{1}{3}\,\textsc{succ}(9,9,9,6)$ and present the allowed splits

$$\begin{aligned} \textsc{succ}(\underline{9,9},9,6) &\geq \textsc{succ}(\underline{7,7},6,4) + \textsc{succ}(\underline{2,2},3,2), \\ \textsc{succ}(\underline{2,2},3,2) &\geq \textsc{succ}(\underline{1,1},1,0) + \textsc{succ}(\underline{1,1},2,2) = 1 + 2 = 3. \end{aligned}$$

Thus, we obtain $\textsc{succ}(3,3,3,2) \geq 1 + \frac{1}{3}\,\textsc{succ}(7,7,6,4)$, implying that $\textsc{succ}(3,2,3,2) \geq \frac{5}{3} + \frac{2}{9}\,\textsc{succ}(7,7,6,4)$.

Putting things together. Claims 1 and 2 together with (14.1) give

$$\textsc{succ}(7,7,6,4) \geq \tfrac{75}{12} + \tfrac{2}{9}\,\textsc{succ}(7,7,6,4).$$

By solving this elementary equation, we obtain $\textsc{succ}(7,7,6,4) \geq \frac{225}{28}$, and it follows that $\textsc{succ}(12,12,12,12) \geq \frac{225}{28} + 8 = \frac{449}{28} = 16 + \frac{1}{28}$. Scaling leads to the claim of the theorem $\textsc{succ}(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4}) \geq \frac{1}{3} + \frac{1}{1344} = \frac{449}{1344} = 0.33407738\ldots$. □

When translating the inequalities into a game tree (see Figure 14.2 for the result), we first observe that in Claim 2 we obtained two different nodes labeled with the position $(3,3,3,2)$. Since there is no reason to treat them differently, we can identify these two nodes and thus obtain a more compact representation of the game tree. This is the reason why the node labeled $(3,3,3,2)$ in Figure 14.2 has two incoming edges.

Interestingly, such identifications can lead to cycles. If we translate the equations for position $(7,7,6,4)$ and its children into a graph, then we observe that the node for $(7,7,6,4)$ has a descendant also labeled $(7,7,6,4)$ (this is what led to the inequality $\textsc{succ}(7,7,6,4) \geq \frac{75}{12} + \frac{2}{9}\,\textsc{succ}(7,7,6,4)$). By transforming this inequality to

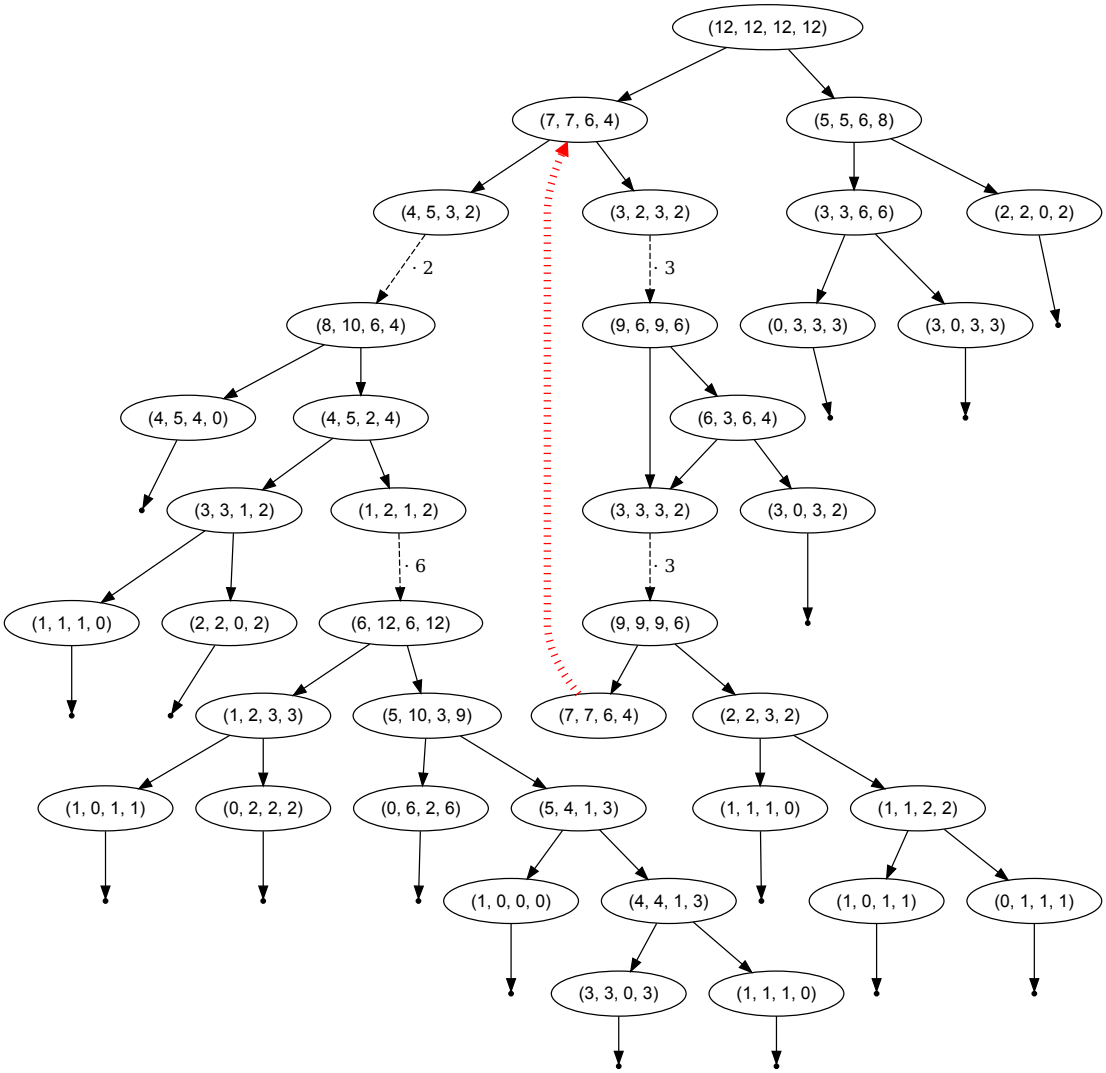


FIGURE 14.2: Game tree representation of the protocols of Theorem 14.1.10.

$\textsc{succ}(7,7,6,4) \geq \frac{225}{28}$, we obtain a statement that is true, but that does not anymore refer to an actual (finite) game tree. However, there is a sequence of game trees with values converging to the value we determined. These trees are obtained from recursively applying the above splitting procedure for $(7,7,6,4)$ a certain number ℓ of times and then using the 0-round tree for the lowest node labeled $(7,7,6,4)$. The value of this game tree is $8 + \sum_{i=0}^{\ell-1}(\frac{2}{9})^i \frac{75}{12} + (\frac{2}{9})^\ell \textsc{succ}_0(7,7,6,4) = 8 + \frac{75}{12}\frac{1-(\frac{2}{9})^\ell}{1-\frac{2}{9}} + 6 \cdot (\frac{2}{9})^\ell = \frac{449}{28} - \frac{57}{28}(\frac{2}{9})^\ell$. Hence for $\ell \geq 3$, this is more than 16 (which represents a success probability of $\frac{1}{3}$), corresponding to a game tree of height[1] $4 + 4\ell \geq 16$.

14.1.7 Automated Search

The vector splitting game formulation enables us to search for good cryptogenographic protocols as follows. We try to determine the game values of all positions from a discrete set $\mathcal{D} := \{0,\dots,T\}^{2\times k}$ by repeatedly applying allowed splits. More precisely, we store a function $s : \mathcal{D} \to \mathbb{R}$ that gives a lower bound on the game value $\textsc{succ}(D)$ of each position $D \in \mathcal{D}$. We initialize this function with $s \equiv \textsc{succ}_0$ and then in order of ascending $\|D\|_1$ try all allowed splits $D = D_0 + D_1$ and update $s(D) \leftarrow s(D_0) + s(D_1)$ in case we find that $s(D)$ was smaller.

Recall that for any secret distribution D, the game value $\textsc{succ}(D)$ is the supremum success probability of cryptogenographic protocols for D. Hence, e.g., the value $s(T,\dots,T)/(2Tk) \leq \textsc{succ}(1/(2k),\dots,1/(2k))$ is a lower bound for this success probability. As we will discuss later, by keeping track of the update operations performed, we can not only compute such a lower bound, but also concrete protocols.

Since even for $k = 2$, the size of the position space $\mathcal{D}$ and the number of allowed splits increase quickly with T, only moderate choices of T are computationally feasible, limiting the power of this approach drastically. However, using the scaling invariance $\lambda\textsc{succ}(D) = \textsc{succ}(\lambda D)$, we can introduce a scaling step: we iteratively optimize using allowed splits (or rather, a generalization called *relaxed splits* that we define below), then backpropagate the computed values (updating, e.g., $s(1,1,1,1) \leftarrow (1/T) \cdot s(T,T,T,T)$) to repeat the process. Surprisingly, this simple modification is sufficient to find protocols that are better than the previous best protocol TwoBit. Algorithm 9 outlines our basic search procedure.

Instead of restricting to optimize only over all allowed splits $(D_0, D_1) \in \mathcal{D}^2$ of $D \in \mathcal{D}$, however, we use monotonicity of the discretization to exploit even more reasonable splits. For ease of presentation, we focus here on the 2-player case (the generalization to larger values of k is straightforward).

Definition 14.1.11. *We call the distributions $D_0 = (a_0,b_0,c_0,d_0) \in \mathcal{D}$ and $D_1 = (a_1,b_1,c_1,d_1) \in \mathcal{D}$ a* relaxed split *of $D = (a,b,c,d) \in \mathcal{D}$, if $a = a_0 + a_1$, $b = b_0 + b_1$, $c = c_0 + c_1$, $d_0 = \lfloor (d/c) \cdot c_0 \rfloor$ and $d_1 = \lfloor (d/c) \cdot c_1 \rfloor$.*

Observation 14.1.12. *Any relaxed split $(D_0, D_1) \in \mathcal{D}^2$ of $D \in \mathcal{D}$ yields a feasible lower bound $\textsc{succ}(D) \geq \textsc{succ}(D_0) + \textsc{succ}(D_1)$.*

Proof. The statement follows from noting that $\bar{D_0} := (a_0,b_0,c_0,(d/c)\cdot c_0)$ and $\bar{D_1} := (a_1,b_1,c_1,(d/c)\cdot c_1)$ yield an allowed split of D. Thus $\textsc{succ}(D) \geq \textsc{succ}(\bar{D_0}) + \textsc{succ}(\bar{D_1}) \geq \textsc{succ}(D_0) + \textsc{succ}(D_1)$, where the last inequality follows from monotonicity (Proposition 14.1.5). □

[1] Note that the height of a game tree refers to the number of transmitted bits and thus does not include the number of (virtual) scaling moves.

Algorithm 9 Computing a lower bound on SUCC for the (discretized) positions $\mathcal{D} = \{0, \ldots, T\}^{2\times k}$.

Initialize $s \equiv \text{SUCC}_0$
repeat
  Splitting Step:
  for all $D \in \mathcal{D}$ in ascending order of $\|D\|_1$ **do**
    for all relaxed splits $(D_0, D_1) \in \mathcal{D}^2$ of D **do**
      if $s(D) < s(D_0) + s(D_1)$ **then**
        Update $s(D) \leftarrow s(D_0) + s(D_1)$
  Scaling Step:
  for all $D \in \mathcal{D}$ and all $\lambda \in \mathbb{N}$ with $\lambda D \in \mathcal{D}$ **do**
    if $s(D) < \frac{s(\lambda D)}{\lambda}$ **then**
      Update $s(D) \leftarrow \frac{s(\lambda D)}{\lambda}$
until termination criterion is met

Note that while the definition relaxes an allowed split only at coordinate d, by symmetry of SUCC, we obtain the same lower bound when relaxing at other coordinates. This allows us to even split distributions $D = (a, b, c, d) \in \mathcal{D}$ where neither of the ratios a/b and c/d occur "perfectly" in another distribution $D' \in \mathcal{D}$ by dismissing some "vector mass", i.e., rounding down from $d/c \cdot c_i$ to $\lfloor d/c \cdot c_i \rfloor$. Although these splits might appear inherently wasteful (as this loss can never be regained), the best protocols that we find do indeed make use of (a small number of) such relaxed splits.

Implementation Details. Since SUCC is symmetric in both the secret and the player dimension, for all distributions $D = ((D_{0,1}, D_{1,1}), \ldots, (D_{0,k}, D_{1,k})) \in \mathcal{D}$ we may assume the following standard form to speed up computation and reduce memory usage: $D_{0,1} \geq D_{1,1}$ and $D_{0,i} \geq D_{0,i+1}$ for $i = 1, \ldots, k-1$. More specifically, for $k = 2$, we can without loss of generality even assume that $D_{0,1} \geq D_{1,0}, D_{0,2}, D_{1,2}$.

In principle, an implementation should take care to avoid propagation of floating-point rounding errors, since previously computed entries are reused heavily and identical values are regularly recalculated in a number of different ways. Instead of using interval arithmetic, however, we chose to use a simple, fast implementation ignoring potential rounding errors: This is justified by (i) our LP-based post-optimization which gives a proof of the obtained lower bound (that can be checked using an exact LP solver), hence correctness of the output remains certified, and (ii) the fact that our discretization of the search space introduces an inherent imprecision that very likely dominates the floating-point rounding errors.

The probably most desirable termination criterion is to run the search until no improvements can be found. However, when the running time becomes a bottleneck issue, we can restrict the search to a small fixed number of iterations. This is especially useful in combination with the post-optimization (as the gain in the result per iteration is decreasing for later iterations of the process, which intuitively give increasingly accurate approximations of infinite "ideal" protocols – the post-optimization could potentially resolve these cyclic structures earlier in the process).

Results. The success probabilities of the protocols computed following the above approach, using different values for T, are given in the first line of Table 14.1. Further results exploiting the post-optimization are given in Table 14.2 in Section 14.1.9.

T	15	20	25	30
Automated search	0.3369432925	0.3376146092	0.3379027186	0.3381689066
Iterations	119	129	141	146
Constraints	535	1756	4217	13958
Game Positions	394	1326	2956	9646

TABLE 14.1: Lower bounds $s(T,\dots,T)/(4T)$ on $\text{SUCC}(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4})$ stemming from the automated search (line 1). Given are also the number of iterations until the automated search procedure converged, i.e., stopped finding improvements using relaxed splits or scalings, and the number of game positions and constraints that had an influence on the value of $s(T,\dots,T)$.

14.1.8 Post-Optimization via Linear Programming

When letting Algorithm 9 also keep track of at what time which update operation was performed, this data can be used to extract strategies for the splitting game (and cryptogenographic protocols). Some care has to be taken to only extract those intermediate positions that had an influence on the final game value for the position we are interested in (see below).

While this approach does deliver good cryptogenographic protocols, manually verifying the correctness of the updates or analyzing the structure of the underlying protocol quickly becomes a difficult task, as the size of the protocol grows rapidly.

Fortunately, it is possible to output a compact, machine-verifiable certificate for the lower bound obtained by the automated search that might even prove a better lower bound than computed: Each update step in the automated search corresponds to a valid inequality of the form $\text{SUCC}(D) \geq \text{SUCC}(D_0) + \text{SUCC}(D_1)$, $\text{SUCC}(D) \geq \text{SUCC}_0(D)$ or $\lambda \cdot \text{SUCC}(D) = \text{SUCC}(\lambda \cdot D)$. We can extract the (sparse) set $\text{ineq}(T,T,T,T)$ of those inequalities that lead to the computed lower bound on $\text{SUCC}(T,T,T,T)$.

Reconstructing the Strategy. Memorizing the best splits found by the dynamic programming updates, it is straightforward to reconstruct the best strategy found by the automated search. For preciseness, we define the *i-th update step* for $i = 2k-1$ as the k-th splitting step (during the execution of Algorithm 9) and for $i = 2k$ as the k-th scaling step, i.e., each update step is alternatingly a splitting and a scaling step. For every distribution D and update step i, we maintain an index $L(D,i)$ defined as the last update step before and including i in which $s(D)$ has been updated to a better value, or 0 if $s(D)$ has never been updated. Moreover, for every D and update step i in which $s(D)$ has been updated, we keep a constraint $I(D,i)$ which represents the inequality or equality used to update $s(D)$ to its best value in update step i. More specifically, $I(D,i)$ stores the inequality $\text{SUCC}(D) \geq \text{SUCC}(D_0) + \text{SUCC}(D_1)$ if the update step i represents a splitting step of D into (D_0,D_1) and $\lambda\text{SUCC}(D) = \text{SUCC}(\lambda D)$ if the update step i represents a scaling of D by λ. Furthermore, we let $I(D,0)$ represent the inequality $\text{SUCC}(D) \geq \text{SUCC}_0(D)$. This gives rise to the procedure $\text{ineq}(D,i)$ (see Algorithm 10) that returns a (sparse) list of inequalities proving the lower bound computed by the automated search (after letting it run for i update steps). When using Algorithm 9 with I iterations and discretization T, we may thus obtain a proof $\text{ineq}(T,T,T,T) = \text{ineq}((T,T,T,T),2I)$ of the lower bound computed.

Algorithm 10 Computing the proof $\text{ineq}(D, i)$ of the lower bound on $\text{SUCC}(D)$ obtained by running the automated search for i update steps.

```
1: function ineq(D, i)
2:     if L(D, i) < i then
3:         return ineq(D, L(D, i))
4:     if i is a splitting step with I(D, i) = "SUCC(D) ≥ SUCC(D_0) + SUCC(D_1)" then
5:         return I(D, i) ∪ ineq(D_0, i) ∪ ineq(D_1, i)
6:     else if i is a scaling step with I(D, i) = "λSUCC(D) = SUCC(λD)" then
7:         return I(D, i) ∪ ineq(λD, i − 1)
8:     else if i = 0 with I(D, 0) = "SUCC(D) ≥ SUCC_0(D)" then
9:         return I(D, 0)
```

The Linear Program. Consider replacing each occurrence of $\text{SUCC}(D')$ in the set of inequalities $\text{ineq}(T, T, T, T)$ found by the automated search by a variable $s_{D'}$. We obtain a system of linear inequalities S that has the feasible solution $s_{D'} = \text{SUCC}(D')$ (for every occurring vector D'). Hence in particular, the optimal solution of the linear program of *minimizing* s_D *subject to* S is a lower bound on $\text{SUCC}(D)$. It is easy to see that this solution is at least as good as the solution stemming from the automated search alone. It can, however, even be better, in particular when a game strategy yields cyclic visits to certain positions. Table 14.2 contains, for different values of T, the success probabilities found by automated search (run with a bounded iteration number of 20) and by this above linear programming approach. The table also contains the number of linear inequalities (and game positions) that were extracted from the automated search run. We observe that consistently the LP-based solution is minimally better. We also observe that the number of constraints is still moderate, posing no difficulties for ordinary LP solvers (which stands in stark contrast to feeding all relaxed splits and scalings over the complete discretization to the LP solver, which quickly becomes infeasible).

Hence the advantage of our approach of extracting the constraints from the automated search stage is that it generates a much sparser sets of constraints that still are sufficiently meaningful. After solving the LP, we can further sparsify this set of inequalities by deleting all inequalities that are not tight in the optimal solution of the LP, since these cannot correspond to the best splits found for the corresponding vector D, yielding a smaller set of relevant inequalities, which might help to analyze the structure of strong protocols.

14.1.9 Our Best Protocol

We report the best protocol we have found using the approach outlined in the previous sections.

Theorem 14.1.13 (Stronger Protocols II)**.** *For the 2-player cryptogenography problem,* $\text{SUCC}(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}) \geq 0.3384736$.

Proof. On `http://people.mpi-inf.mpg.de/~marvin/verify.html`, we provide a linear program based on feasible inequalities on the discretization $\mathcal{D}$ with $T = 50$. To verify the result, one only has to (1) check validity of each inequality, i.e., checking whether each constraints encodes a feasible scaling, relaxed split or zero-bit success probability and (2) solve the linear program. Since we represent the distributions $D = (a, b, c, d)$ using a normal form $a \geq b, c, d$ (to break symmetries), checking validity of each splitting constraint is not completely trivial, but easy. We provide a simple checker program to

T	30	35	40	45	50
Automated search	0.3381086510	0.3381937725	0.3383218072	0.3383946540	0.3384414508
LP solution	0.3381527322	0.3382301900	0.3383547901	0.3384303130	0.3384736461
Iterations	20	20	20	20	20
Constraints	5373	8882	12410	18659	24483
Game states	4126	6789	9396	13992	18248

TABLE 14.2: Lower bounds $s(T,\dots,T)/(4T)$ on $\text{SUCC}(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4})$ stemming from the automated search only (line 1) and from the LP solution of the linear system extracted from the automated search data (line 2), when the number of iterations is restricted to 20.

verify validity of the constraints. The LP is output in a format compatible with the LP solver `lp_solve`[2]. □

14.2 A Stronger Hardness Result

In this section, we prove that any 2-player cryptogenographic protocol has a success probability of at most 0.3672. This improves over the previous 0.375 bound of [Bro+14].

Theorem 14.2.1 (Stronger Hardness Result)**.** *For the 2-player cryptogenography problem, we have* $\text{SUCC}(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4}) \leq \frac{47}{128} = 0.367188$.

To prove the result, we apply the concavity method used by Brody et al. [Bro+14] which consists of finding a function s that (i) is lower bounded by SUCC_0 for all distributions and (ii) satisfies a certain concavity condition. We first relax the lower bound requirement to hold only for six particular simple distributions (namely, the distributions $(1,0,0,0),\dots,(0,0,0,1)$, $(\frac{1}{2},0,\frac{1}{2},0)$ and $(0,\frac{1}{2},0,\frac{1}{2})$) instead of all distributions. This simplifies the search for a suitable stronger candidate function satisfying (i) - it remains to verify condition (ii) for the thus found candidate function.

We first describe the previously used concavity method in Section 14.2.1 and apply it to our new upper bound function in Section 14.2.2.

14.2.1 Revisiting the Concavity Method

We revisit the original result of Brody et al. [Bro+14, Theorem 4.4] establishing the concavity method for the cryptogenography problem. Similar concavity arguments have appeared earlier in different settings in information complexity and information theory (e.g., [Bra+13; MI13; MIG12]). In what follows, we mostly use the language of convex combinations of secret distributions in allowed planes. To this aim, let $\Delta := \{D \in \mathbb{R}^{2\times 2}_{\geq 0} \mid \|D\|_1 = 1\}$ denote the set of these distributions. For a given distribution $D = (a,b,c,d) \in \Delta$, there are two *allowed planes* through D, namely $\{(a',b',\delta c,\delta d) \mid a',b',\delta \in \mathbb{R}\} \cap \Delta$ and $\{(\delta a,\delta b,c',d') \mid c',d',\delta \in \mathbb{R}\} \cap \Delta$.

Lemma 14.2.2 (Concavity Method, [Bro+14, Theorem 4.4])**.** *Let* $s : \Delta \to \mathbb{R}$ *satisfy*

(C1) $s(D) \geq \lambda s(D_0) + (1-\lambda)s(D_1)$ *for all* $\lambda \in [0,1]$ *and all* D_0, D_1 *such that* $D = \lambda D_0 + (1-\lambda)D_1$ *and* D_0, D_1 *lie in the same allowed plane through* D.

(C2) $s(D) \geq \text{SUCC}_0(D)$ *for all* $D \in \Delta$.

[2] `lpsolve.sourceforge.net`

Then $\textsc{succ}(D) \leq s(D)$ *holds for all* $D \in \Delta$.

To transfer this method to the vector splitting formulation, consider an $s' : \mathbb{R}^{2\times 2}_{\geq 0} \to \mathbb{R}$, which is scalable (i.e., $s'(\lambda D) = \lambda s'(D)$ for all $\lambda \in \mathbb{R}_{\geq 0}, D \in \mathbb{R}^{2\times 2}_{\geq 0}$) and when restricted to Δ equals s. Then condition (C1) is equivalent to $s'(D) \geq s'(D_0) + s'(D_1)$ for all allowed splits (D_0, D_1) of D, i.e., superadditivity of s' for all allowed splits. Condition (C2) remains effectively the same: $s'(D) \geq \textsc{succ}_0(D)$ for all $D \in \mathbb{R}^{2\times 2}_{\geq 0}$. If these conditions hold, we obtain $\textsc{succ}(D) \leq s'(D)$ for all $D \in \mathbb{R}^{2\times 2}_{\geq 0}$, which is equivalent to $\textsc{succ}(D) \leq s(D)$ for all $D \in \Delta$.

As a simple application of the concavity method, we can now give the simple proof of Proposition 14.1.7 stated in Section 14.1.5.

Proposition 14.1.7. *We have* $\textsc{succ}(a, b, c, d) \leq \min\{a, c\} + \min\{b, d\}$.

Proof. We use the vector splitting formulation. Define $s_{UB}(a, b, c, d) := \min\{a, c\} + \min\{b, d\}$. We have

$$\textsc{succ}_0(a, b, c, d) = \max\{\min\{a, c\}, \min\{b, d\}\} \leq \min\{a, c\} + \min\{b, d\} = s_{UB}(a, b, c, d),$$

which proves condition (C2) of Lemma 14.2.3. Note that $f : (x, y) \mapsto \min\{x, y\}$ is superadditive and hence s_{UB}, as a sum of superadditive functions, is superadditive as well. This proves $s_{UB}(D) \geq s_{UB}(D_0) + s_{UB}(D_1)$ even for *all* splits $D = D_0 + D_1$ (not only allowed splits). □

The following lemma is an extension of the concavity theorem. We relax the condition that s is lower bounded by $\textsc{succ}_0$ on *all* distributions to now on only six particular, very simple distributions.

Lemma 14.2.3 (Concavity Method, adapted)**.** *Let* $s : \Delta \to \mathbb{R}$ *satisfy*

(C1) $s(D) \geq \lambda s(D_0) + (1 - \lambda)s(D_1)$ *for all* $\lambda \in [0, 1]$ *and all* D_0, D_1 *such that* $D = \lambda D_0 + (1 - \lambda)D_1$ *and* D_0, D_1 *lie in the same allowed plane through* D.

(C2')
- $s(1/2, 0, 1/2, 0), s(0, 1/2, 0, 1/2) \geq 1/2$, *and*
- $s(1, 0, 0, 0), s(0, 1, 0, 0), s(0, 0, 1, 0), s(0, 0, 0, 1) \geq 0$.

Then $\textsc{succ}(D) \leq s(D)$ *holds for all* $D \in \Delta$.

Proof. To appeal to Lemma 14.2.2, we need to show that (C1) and (C2') imply (C2), i.e., for all $D = (a, b, c, d) \in \Delta$, we have $s(D) \geq \textsc{succ}_0(D) = \max\{\min\{a, c\}, \min\{b, d\}\}$.

To prove this statement, we again find it more convenient to use the vector splitting formulation. To this aim, we extend s to $s' : \mathbb{R}^{2\times 2}_{\geq 0} \to \mathbb{R}$ by defining $s'(D) := \|D\|_1 \cdot s(\frac{D}{\|D\|_1})$. Recall that in this view (C1) is equivalent to $s'(D) \geq s'(D_0) + s'(D_1)$ for all allowed splits (D_0, D_1) of D. Assume that $\min\{a, c\} \geq \min\{b, d\}$ (the other case is symmetric). Consider $D_0 := (a, 0, c, d)$, $D_1 := (0, b, 0, 0)$, which is a 1-allowed split of D. By scaling and (C2'), we have $s'(D_1) = b \cdot s(0, 1, 0, 0) \geq 0$. We split D_0 into $E_0 := (a, 0, c, 0)$ and $E_1 := (0, 0, 0, d)$, which is a 2-allowed split. Again, by scaling and (C2'), we obtain $s'(E_1) \geq 0$. Assuming that $a \geq b$ (since the other case is symmetric), we finally split E_0 into $F_0 := (b, 0, b, 0)$ and $F_1 := (a - b, 0, 0, 0)$, which is a 1-allowed split that satisfies, by scaling and (C2'), $s'(F_1) \geq 0$ and $s'(F_0) = 2b \cdot s(\frac{1}{2}, 0, \frac{1}{2}, 0) \geq b$. Thus, $s(D) = s'(D) \geq b = \textsc{succ}_0(D)$. □

14.2.2 The Adapted Upper Bound Function

Motivated by our relaxation of the condition of the concavity method, we adapt the upper bound function of Brody et al. [Bro+14] and set

$$\begin{aligned} s(a,b,c,d) &:= \frac{1-f(a,b,c,d)}{4}, \\ f(a,b,c,d) &:= a^2+b^2+c^2+d^2-6(ac+bd)+8abcd. \end{aligned}$$

In fact, we have changed their upper bound function by introducing an additional term of $8abcd$, which attains a value of zero on the distributions $(\frac{1}{2},0,\frac{1}{2},0)$, $(1,0,0,0)$, etc., thus not affecting the zero-bit success probability condition of the concavity method. Additionally, this function also satisfies the concavity condition, which we will prove later.

Lemma 14.2.4. *The thus defined function s satisfies the concavity condition (C1) of Lemma 14.2.3.*

As an immediate consequence of the concavity method of Lemma 14.2.3, we obtain the upper bound of $\textsc{succ}(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4}) \leq \frac{47}{128}$ of Theorem 14.2.1.

Proof of Theorem 14.2.1. Simple calculations show that s satisfies $s(1,0,0,0)=0$ and $s(1/2,0,1/2,0)=1/2$, which by symmetry implies that s satisfies condition (C2'). Since additionally condition (C1) is satisfied by Lemma 14.2.4, the concavity method (Lemma 14.2.3) is applicable and yields $\textsc{succ}(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4}) \leq s(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4}) = \frac{47}{128}$. □

In the remainder of this section, we will show that s does not violate concavity on allowed planes (condition (C1)), i.e., prove Lemma 14.2.4. To this end, we exclusively use the convex combination view, in which $\|D_0\|_1 = \|D_1\|_1 = \|D\| = 1$.

For $q \geq 0$, we define $f_q : S \to \mathbb{R}$, where $S := \{(a,b) \mid 0 \leq a,b \leq 1, a+b \leq 1\}$, by

$$f_q(a,b) := f\left(a, b, \frac{1}{1+q}(1-a-b), \frac{q}{1+q}(1-a-b)\right),$$

i.e., for fixed $q \geq 0$, this function maps each pair $(a,b) \in S$ to the value of f at the unique distribution (a',b',c',d') with $a'=a$, $b'=b$ and $d'/c'=q$.

Lemma 14.2.5. *If f_q is convex for all $q \geq 0$, then s satisfies the concavity condition (C1) of Lemma 14.2.3.*

Proof. Let $D=(a,b,c,d)$ be an arbitrary distribution and let $D_0=(a_0,b_0,c_0,d_0), D_1 = (a_1,b_1,c_1,d_1)$ be such that $D = \lambda D_0 + (1-\lambda)D_1$ for some $\lambda \in [0,1]$ and D_0, D_1 lie in the same allowed plane through D. By the symmetry $s(a,b,c,d) = s(c,d,a,b)$, we can without loss of generality assume that D_0, D_1 and D are proportional on the entries of player 2 (i.e., a 1-allowed split).

Assume first that $c=d=0$, then any D_0 and D_1 lie on the line $(t,1-t,0,0)$ with $t \in \mathbb{R}$. Define the restriction of s to this line as $\tilde{s}(t) := s(t,1-t,0,0) = \frac{1}{4}\left(1-(1-t)^2-t^2\right)$ and note that $\tilde{s}(t)$ is concave by $\frac{d^2\tilde{s}}{dt^2} = -1$. Thus (C1) is satisfied for $c=d=0$.

Hence we may assume $c>0$ or $d>0$ and more specifically, by the symmetry $s(a,b,c,d) = s(b,a,d,c)$, that $c>0$. We set $q := \frac{d}{c}$ and observe that $q = \frac{d_0}{c_0} = \frac{d_1}{c_1}$, since (D_0,D_1) is a 1-allowed split of D. Then by $\|D\|_1 = \|D_i\|_1 = 1$, we have $D = (a,b,\frac{1}{1+q}(1-a-b), \frac{q}{1+q}(1-a-b))$ and $D_i = (a_i,b_i,\frac{1}{1+q}(1-a_i-b_i), \frac{q}{1+q}(1-a_i-b_i))$ (for

$i \in \{0,1\}$). Recall that $D = \lambda D_0 + (1-\lambda)D_1$. The claim now follows from the simple calculation

$$\begin{aligned} s(D) &= \frac{1 - f(a,b,\frac{1}{1+q}(1-a-b),\frac{q}{1+q}(1-a-b))}{4} = \frac{1-f_q(a,b)}{4} \\ &\geq \lambda\frac{1-f_q(a_0,b_0)}{4} + (1-\lambda)\frac{1-f_q(a_1,b_1)}{4} \qquad \text{[by convexity of } f_q\text{]} \\ &= \lambda s(D_0) + (1-\lambda)s(D_1). \end{aligned}$$

□

It remains to analyze the concavity of f_q.

Lemma 14.2.6. *For all $q \geq 0$, f_q is concave on S.*

Proof. We use the fact that f_q has continuous second partial derivatives and thus is convex if the Hessian $H := \mathbf{H}(f_q)$ is positive semidefinite. By straight-forward calculations, we obtain

$$\mathbf{H}(f_q) = \begin{pmatrix} \frac{4\left(q^2+4\left(2b^2+(3a-2)b+1\right)q+4\right)}{(q+1)^2} & \frac{4\left(2q^2+\left(6a^2+8(2b-1)a+6b^2-8b+5\right)q+2\right)}{(q+1)^2} \\ \frac{4\left(2q^2+\left(6a^2+8(2b-1)a+6b^2-8b+5\right)q+2\right)}{(q+1)^2} & \frac{4\left(4q^2+4\left(2a^2+(3b-2)a+1\right)q+1\right)}{(q+1)^2} \end{pmatrix}.$$

To verify positive semidefiniteness, we exploit the criterion that $\mathbf{H}(f_q)$ is positive semidefinite if its principal minors are non-negative, i.e.,

$$\frac{4\left(4+4\left(1+(-2+3a)b+2b^2\right)q+q^2\right)}{(1+q)^2} \geq 0, \qquad \text{and} \tag{14.2}$$

$$\mathbf{Det}[H] \geq 0. \tag{14.3}$$

Verifying (14.2) is equivalent to showing that $4 + 4p_1(a,b)q + q^2 \geq 0$, where $p_1(a,b) := 1+(-2+3a)b+2b^2$. Note that by $0 \leq a,b \leq 1$, we have $p_1(a,b) \geq 1-2b+2b^2 \geq 1-2b \geq -1$. Thus,

$$4 + 4p_1(a,b)q + q^2 \geq 4 - 4q + q^2 = (q-2)^2 \geq 0,$$

proving (14.2).

Regarding (14.3), straight-forward calculations reveal that

$$\mathbf{Det}[H] = \frac{64q}{(1+q)^4}\left(p_2(a,b) + p_3(a,b)q + p_4(a,b)q^2\right),$$

where

$$\begin{aligned} p_2(a,b) &= 2a^2 + 6b - ab - 4b^2, \\ p_3(a,b) &= 12(a+b) - 23(a^2+b^2) - 32ab + 24(a^3(1-b) + b^3(1-a)) \\ &\quad +48(ab^2 + a^2b) - 30a^2b^2 - 9(a^4+b^4), \\ p_4(a,b) &= 6a - 4a^2 - ab + 2b^2. \end{aligned}$$

Claim 14.2.7. *For all $0 \leq a,b \leq 1$ with $a+b \leq 1$ we have*

$$p_2(a,b), p_3(a,b), p_4(a,b) \geq 0.$$

Proof. Note that by $a,b \leq 1$, we have $p_2(a,b) = 2a^2 + 6b - ab - 4b^2 \geq 2a^2 + 6b - b - 4b = 2a^2 + b \geq 0$. Since $p_4(a,b) = p_2(b,a)$, it directly follows that $p_4(a,b) \geq 0$.

To prove the remaining statement $p_3(a,b) \geq 0$, we will exploit the following basic inequalities

$$a^4 + b^4 \leq a^3(1-b) + b^3(1-a), \tag{14.4}$$
$$a^2b^2 \leq \frac{1}{2}(ab^2 + a^2b - ab^3 - a^3b), \tag{14.5}$$

which directly follow from plugging in $a \leq 1-b$ and $b \leq 1-a$ into the left-hand sides. We compute

$$\begin{aligned}
p_3(a,b) = \; & 12(a+b) - 23(a^2+b^2) - 32ab + 24(a^3(1-b) + b^3(1-a)) \\
& + 48(ab^2 + a^2b) - 30a^2b^2 - 9(a^4+b^4), \\
\geq \; & 12(a+b) - 23(a^2+b^2) - 32ab + 15(a^3(1-b) + b^3(1-a)) \\
& + 48(ab^2 + a^2b) - 30a^2b^2 && \text{[by (14.4)]} \\
\geq \; & 12(a+b) - 23(a^2+b^2) - 32ab + 15(a^3+b^3) \\
& + 33(ab^2 + a^2b) && \text{[by (14.5)]} \\
= \; & 12(a+b) - 23(a+b)^2 + 14ab + 4(a^3+b^3) \\
& + 11(a+b)^3 \\
\geq \; & 12(a+b) - 23(a+b)^2 + 11(a+b)^3,
\end{aligned}$$

Basic calculus shows that $t : [0,1] \to \mathbb{R}, s \mapsto 12s - 23s^2 + 11s^3$ has global minima $t(0) = t(1) = 0$ and thus $p_3(a,b) \geq t(a+b) \geq 0$. □

Thus $p_2(a,b) + p_3(a,b)q + p_4(a,b)q^2 \geq 0$ follows from $q \geq 0$, which implies that $\mathbf{Det}[H] \geq 0$. Hence, we have verified (14.2) and (14.3), which proves that f_q is convex for all $q \geq 0$. □

Proof of Lemma 14.2.4. Combining Lemmas 14.2.5 and 14.2.6 yields that s is concave on all allowed planes. □

14.3 Conclusion

Despite the fundamental understanding of the cryptogenography problem obtained by Brody et al. [Bro+14], determining the success probability even of the 2-player case remains an intriguing open problem. The previous best protocol with success probability 1/3, while surprising and unexpected at first, is natural and very symmetric (in particular when viewed in the convex combination or vector splitting game formulation). We disprove the hope that it is an optimal protocol by exhibiting less intuitive and less symmetric protocols having success probabilities up to 0.3384. Concerning hardness results, our upper bound of 0.3671875 shows that also the previous upper bound of 3/8 was not the final answer. These findings add to the impression that the cryptography problem offers a more complex nature than its simple description might suggest and that understanding the structure of good protocols is highly non-trivial.

We are optimistic that our methods support a further development of improved protocols and bounds. (1) Trivially, investing more computational power or optimizing the automated search might lead to finding better protocols. (2) Our improved protocols might motivate to (manually) find infinite protocol families exploiting implicit properties and structure of these protocols. (3) Our reformulations, e.g., as vector splitting game,

might ease further searches for better protocols and for better candidate functions for a hardness proof.

Bibliography

[AB10] Helmut Alt and Maike Buchin. "Can We Compute the Similarity Between Surfaces?" In: *Discrete & Computational Geometry* 43(1) (2010), pp. 78–99.

[Abb+16a] Amir Abboud, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Ryan Williams. "Simulating Branching Programs with Edit Distance and Friends or: A Polylog Shaved is a Lower Bound Made". In: *Proc. 48th Annual ACM Symposium on Symposium on Theory of Computing (STOC'16)*. To appear. 2016.

[Abb+16b] Amir Abboud, Arturs Backurs, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Or Zamir. "Subtree Isomorphism Revisited". In: *Proc. 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'16)*. 2016, pp. 1256–1271.

[ABVW15a] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. "If the Current Clique Algorithms are Optimal, So is Valiant's Parser". In: *Proc. 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS'15)*. 2015, pp. 98–117.

[ABVW15b] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. "Quadratic-Time Hardness of LCS and other Sequence Similarity Measures". In: *Proc. 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS'15)*. 2015, pp. 59–78.

[AG87] Alberto Apostolico and Concettina Guerra. "The Longest Common Subsequence Problem Revisited". In: *Algorithmica* 2(1) (1987), pp. 316–336.

[AG95] Helmut Alt and Michael Godau. "Computing the Fréchet Distance Between Two Polygonal Curves". In: *International Journal of Computational Geometry & Applications* 5(1–2) (1995), pp. 78–99.

[Aga+14] Pankaj K. Agarwal, Rinat Ben Avraham, Haim Kaplan, and Micha Sharir. "Computing the Discrete Fréchet Distance in Subquadratic Time". In: *SIAM Journal on Computing* 43(2) (2014), pp. 429–449.

[AGW15] Amir Abboud, Fabrizio Grandoni, and Virginia Vassilevska Williams. "Subcubic Equivalences Between Graph Centrality Problems, APSP and Diameter". In: *Proc. 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'15)*. 2015, pp. 1681–1697.

[AHU76] Alfred V. Aho, Daniel S. Hirschberg, and Jeffrey D. Ullman. "Bounds on the Complexity of the Longest Common Subsequence Problem". In: *Journal of the ACM* 23(1) (1976), pp. 1–12.

[AKO10] Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. "Polylogarithmic Approximation for Edit Distance and the Asymmetric Query Complexity". In: *Proc. 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS'10)*. 2010, pp. 377–386.

[AKW04] Helmut Alt, Christian Knauer, and Carola Wenk. "Comparison of Distance Measures for Planar Curves". In: *Algorithmica* 38(1) (2004), pp. 45–58.

[Alt09] Helmut Alt. "The Computational Geometry of Comparing Shapes". In: *Efficient Algorithms: Essays Dedicated to Kurt Mehlhorn on the Occasion of His 60th Birthday*. 2009, pp. 235–248.

[AMR11] David Arthur, Bodo Manthey, and Heiko Röglin. "Smoothed Analysis of the k-Means Method". In: *Journal of the ACM* 58(5) (2011).

[Apo86] Alberto Apostolico. "Improving the Worst-Case Performance of the Hunt-Szymanski Strategy for the Longest Common Subsequence of Two Strings". In: *Information Processing Letters* 23(2) (1986), pp. 63–69.

[Aro+06] Boris Aronov, Sariel Har-Peled, Christian Knauer, Yusu Wang, and Carola Wenk. "Fréchet Distance for Curves, Revisited". In: *Proc. 14th Annual European Symposium on Algorithms (ESA'06)*. 2006, pp. 52–63.

[Aro98] Sanjeev Arora. "Polynomial Time Approximation Schemes for Euclidean Traveling Salesman and other Geometric Problems". In: *Journal of the ACM* 45(5) (1998), pp. 753–782.

[AV09] David Arthur and Sergei Vassilvitskii. "Worst-Case and Smoothed Analysis of the ICP Algorithm, with an Application to the k-Means Method". In: *SIAM Journal on Computing* 39(2) (2009), pp. 766–782.

[AVW14] Amir Abboud and Virginia Vassilevska Williams. "Popular Conjectures Imply Strong Lower Bounds for Dynamic Problems". In: *Proc. 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS'14)*. 2014, pp. 434–443.

[AVWW14] Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. "Consequences of Faster Alignment of Sequences". In: *Proc. 41st International Colloquium on Automata, Languages, and Programming (ICALP'14)*. 2014, pp. 39–51.

[AVWW16] Amir Abboud, Virginia Vassilevska Williams, and Joshua Wang. "Approximation and Fixed Parameter Subquadratic Algorithms for Radius and Diameter". In: *Proc. 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'16)*. 2016, pp. 377–391.

[AWY15] Amir Abboud, Ryan Williams, and Huacheng Yu. "More Applications of the Polynomial Method to Algorithm Design". In: *Proc. 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'15)*. 2015, pp. 218–230.

[BBW09] Kevin Buchin, Maike Buchin, and Yusu Wang. "Exact Algorithms for Partial Curve Matching via the Fréchet distance". In: *Proc. 20th Annual ACM-SIAM Symposium Discrete Algorithms (SODA'09)*. 2009, pp. 645–654.

[BDR12] Gerth Stølting Brodal, Pooya Davoodi, and S. Srinivasa Rao. "On Space Efficient Two Dimensional Range Minimum Data Structures". In: *Algorithmica* 63(4) (2012), pp. 815–830.

[Ber+05] Noam Berger, Christian Borgs, Jennifer T. Chayes, and Amin Saberi. "On the Spread of Viruses on the Internet". In: *Proc. 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'05)*. 2005, pp. 301–310.

[BFC08] Philip Bille and Martin Farach-Colton. "Fast and compact regular expression matching". In: *Theoretical Computer Science* 409(3) (2008), pp. 486–496.

[BG92] Amir M. Ben-Amram and Zvi Galil. "On Pointers versus Addresses". In: *Journal of the ACM* 39(3) (1992), pp. 617–648.

[BHR00] Lasse Bergroth, Harri Hakonen, and Timo Raita. "A Survey of Longest Common Subsequence Algorithms". In: *Proc. 7th International Symposium on String Processing and Information Retrieval (SPIRE'00)*. 2000, pp. 39–48.

[BI15a] Arturs Backurs and Piotr Indyk. "Edit Distance Cannot Be Computed in Strongly Subquadratic Time (unless SETH is false)". In: *Proc. 47th Annual ACM Symposium on Theory of Computing (STOC'15)*. 2015, pp. 51–58.

[BI15b] Arturs Backurs and Piotr Indyk. "Which Regular Expression Patterns are Hard to Match?" In: *ArXiv e-prints* (2015). arXiv:1511.07070 [cs.CC].

[BK14] K. Bringmann and M. Künnemann. "Improved Approximation for Fréchet Distance on c-Packed Curves Matching Conditional Lower Bounds". In: *ArXiv e-prints* (2014). arXiv:1408.1340 [cs.CG].

[BK15a] Karl Bringmann and Marvin Künnemann. "Improved Approximation for Fréchet Distance on c-Packed Curves Matching Conditional Lower Bounds". In: *Proc. 26th International Symposium on Algorithms and Computation (ISAAC'15)*. 2015, pp. 517–528.

[BK15b] Karl Bringmann and Marvin Künnemann. "Quadratic Conditional Lower Bounds for String Problems and Dynamic Time Warping". In: *Proc. 56th IEEE Symposium on Foundations of Computer Science (FOCS'15)*. 2015, pp. 79–97.

[BK15c] Karl Bringmann and Marvin Künnemann. "Quadratic Conditional Lower Bounds for String Problems and Dynamic Time Warping". In: *ArXiv e-prints* (2015). arXiv:1502.01063 [cs.CC].

[BK16] Karl Bringmann and Marvin Künnemann. *Multivariate Fine-Grained Complexity of Longest Common Subsequence*. Unpublished Manuscript. 2016.

[BM16] Karl Bringmann and Wolfgang Mulzer. "Approximability of the Discrete Fréchet Distance". In: *Journal of Computational Geometry* 7(2) (2016), pp. 46–76.

[BMR13] Markus Bläser, Bodo Manthey, and B. V. Raghavendra Rao. "Smoothed Analysis of Partitioning Algorithms for Euclidean Functionals". In: *Algorithmica* 66(2) (2013), pp. 397–418.

[Bra+05] Sotiris Brakatsoulas, Dieter Pfoser, Randall Salas, and Carola Wenk. "On Map-Matching Vehicle Tracking Data". In: *Proc. 31st International Conference on Very Large Data Bases (VLDB'05)*. 2005, pp. 853–864.

[Bra+13] Mark Braverman, Ankit Garg, Denis Pankratov, and Omri Weinstein. "From Information to Exact Communication". In: *Proc. 45th Annual ACM Symposium on Theory of Computing (STOC'13)*. 2013, pp. 151–160.

[Bri14] Karl Bringmann. "Why Walking the Dog Takes Time: Frechet Distance Has No Strongly Subquadratic Algorithms Unless SETH Fails". In: *Proc. 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS'14)*. 2014, pp. 661–670.

[Bri15] Karl Bringmann. "Sampling from Discrete Distributions and Computing Fréchet Distances". PhD thesis. Saarland University, 2015.

[Bro+14] Joshua Brody, Sune K. Jakobsen, Dominik Scheder, and Peter Winkler. "Cryptogenography". In: *Proc. 5th ACM Conference on Innovations in Theoretical Computer Science (ITCS'14)*. 2014, pp. 13–22.

[Bru+14] Tobias Brunsch, Heiko Röglin, Cyriel Rutten, and Tjark Vredeveld. " Smoothed Performance Guarantees for Local Search". In: *Mathematical Programming* 146(1) (2014), pp. 185–218.

[Buc+11] Kevin Buchin, Maike Buchin, Joachim Gudmundsson, Maarten Löffler, and Jun Luo. "Detecting Commuting Patterns by Clustering Subtrajectories". In: *International Journal of Computational Geometry & Applications* 21(3) (2011), pp. 253–282.

[Buc+14] Kevin Buchin, Maike Buchin, Wouter Meulemans, and Wolfgang Mulzer. "Four Soviets Walk the Dog - with an Application to Alt's Conjecture". In: *Proc. 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'14)*. 2014, pp. 1399–1413.

[Car+16] Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. "Nondeterministic Extensions of the Strong Exponential Time Hypothesis and Consequences for Non-reducibility". In: *Proc. 7th ACM Conference on Innovations in Theoretical Computer Science (ITCS'16)*. 2016, pp. 261–270.

[CG86] Bernard Chazelle and Leonidas J. Guibas. "Fractional Cascading: I. A Data Structuring Technique". In: *Algorithmica* 1(2) (1986), pp. 133–162.

[Cha+10] Erin Wolf Chambers, Éric Colin de Verdière, Jeff Erickson, Sylvain Lazard, Francis Lazarus, and Shripad Thite. "Homotopic Fréchet Distance Between Curves or, Walking Your Dog in the Woods in Polynomial Time". In: *Computational Geometry* 43(3) (2010), pp. 295–311.

[Cha15] Yi-Jun Chang. "Conditional Lower Bound for RNA Folding Problem". In: *ArXiv e-prints* (2015). arXiv:`1511.04731 [cs.CC]`.

[Che+11] Daniel Chen, Anne Driemel, Leonidas J. Guibas, Andy Nguyen, and Carola Wenk. "Approximate Map Matching with respect to the Fréchet Distance". In: *Proc. 13th Workshop on Algorithm Engineering and Experiments (ALENEX'11)*. 2011, pp. 75–83.

[Che+15] Ning Chen, Martin Hoefer, Marvin Künnemann, Chengyu Lin, and Peihan Miao. "Secretary Markets with Local Information". In: *Proc. 42nd International Colloquium on Automata, Languages, and Programming (ICALP'15), Part II*. 2015, pp. 552–563.

[CK13] Radu Curticapean and Marvin Künnemann. "A Quantization Framework for Smoothed Analysis of Euclidean Optimization Problems". In: *Proc. 21st Annual European Symposium on Algorithms (ESA'13)*. Received *Best Student Paper Award*. 2013, pp. 349–360.

[CK15] Radu Curticapean and Marvin Künnemann. "A Quantization Framework for Smoothed Analysis of Euclidean Optimization Problems". In: *Algorithmica* 73(3) (2015), pp. 1–28.

[CKK72] Vaclav Chvatal, David A. Klarner, and Donald E. Knuth. *Selected Combinatorial Research Problems*. Tech. rep. CS-TR-72-292. Stanford University, Department of Computer Science, June 1972.

[CKT99] Barun Chandra, Howard Karloff, and Craig Tovey. "New Results on the Old k-Opt Algorithm for the Traveling Salesman Problem". In: *SIAM Journal on Computing* 28(6) (1999), pp. 1998–2029.

[Cor+09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. 3rd. The MIT Press, 2009.

[CW10] A. F. Cook and Carola Wenk. "Geodesic Fréchet Distance Inside a Simple Polygon". In: *ACM Transactions on Algorithms* 7(1) (2010), pp. 193–204.

[DD08] George Danezis and Claudia Diaz. *A Survey of Anonymous Communication Channels*. Tech. rep. MSR-TR-2008-35. Microsoft Research, Jan. 2008.

[Dem+88] Alan J. Demers, Daniel H. Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard E. Sturgis, Daniel C. Swinehart, and Douglas B. Terry. "Epidemic Algorithms for Replicated Database Maintenance". In: *Operating Systems Review* 22(1) (1988), pp. 8–32.

[DFF12] Benjamin Doerr, Mahmoud Fouz, and Tobias Friedrich. "Why Rumors Spread So Quickly in Social Networks". In: *Commununications of the ACM* 55(6) (2012), pp. 70–75.

[DHP13] Anne Driemel and Sariel Har-Peled. "Jaywalking Your Dog: Computing the Fréchet Distance with Shortcuts". In: *SIAM Journal on Computing* 42(5) (2013), pp. 1830–1866.

[DHPW12] Anne Driemel, Sariel Har-Peled, and Carola Wenk. "Approximating the Fréchet Distance for Realistic Curves in Near Linear Time". In: *Discrete & Computational Geometry* 48(1) (2012), pp. 94–127.

[DK13a] Benjamin Doerr and Marvin Künnemann. "How the $(1 + \lambda)$ Evolutionary Algorithm Optimizes Linear Functions". In: *Proc. 15th Annual Conference on Genetic and Evolutionary Computation (GECCO'13)*. 2013, pp. 1589–1596.

[DK13b] Benjamin Doerr and Marvin Künnemann. "Royal Road Functions and the $(1 + \lambda)$ Evolutionary Algorithm: Almost no Speed-up from Larger Offspring Populations". In: *Proc. 2013 IEEE Congress on Evolutionary Computation (CEC'13)*. 2013, pp. 424–431.

[DK14] Benjamin Doerr and Marvin Künnemann. "Tight Analysis of Randomized Rumor Spreading in Complete Graphs". In: *Proc. 11th Workshop on Analytic Algorithmics and Combinatorics (ANALCO'14)*. 2014, pp. 82–91.

[DK15] Benjamin Doerr and Marvin Künnemann. "Optimizing Linear Functions with the Evolutionary Algorithm – Different Asymptotic Runtimes for Different Instances". In: *Theoretical Computer Science* 561 (2015), pp. 3–23.

[DK16a] Benjamin Doerr and Marvin Künnemann. "Improved Protocols and Hardness Results for the Two-Player Cryptogenography Problem". In: *Proc. 43rd International Colloquium on Automata, Languages, and Programming (ICALP'16)*. To appear. 2016.

[DK16b] Benjamin Doerr and Marvin Künnemann. "Improved Protocols and Hardness Results for the Two-Player Cryptogenography Problem". In: *ArXiv e-prints* (2016). arXiv:1603.06113 [cs.CR].

[DKW10] Benjamin Doerr, Marvin Künnemann, and Magnus Wahlström. "Randomized Rounding for Routing and Covering Problems: Experiments and Improvements". In: *Experimental Algorithms (SEA'10)*. 2010, pp. 190–201.

[DKW11] Benjamin Doerr, Marvin Künnemann, and Magnus Wahlström. "Dependent Randomized Rounding: The Bipartite Case". In: *Proc. 13th Workshop on Algorithm Engineering and Experiments (ALENEX'11)*. 2011, pp. 96–106.

[Doe+09] Benjamin Doerr, Tobias Friedrich, Marvin Künnemann, and Thomas Sauerwald. "Quasirandom Rumor Spreading: An Experimental Analysis". In: *Proc. 10th Workshop on Algorithm Engineering and Experiments (ALENEX'09)*. 2009, pp. 145–153.

[Doe+11] Benjamin Doerr, Tobias Friedrich, Marvin Künnemann, and Thomas Sauerwald. "Quasirandom Rumor Spreading: An Experimental Analysis". In: *ACM Journal of Experimental Algorithmics* 16 (2011), Article No. 3.3.

[Doe11] Benjamin Doerr. "Analyzing Randomized Search Heuristics: Tools from Probability Theory". In: *Theory of Randomized Search Heuristics*. Ed. by A. Auger and B. Doerr. World Scientific Publishing, 2011, pp. 1–20.

[DP09] Devdatt Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.

[Dri+89] James R. Driscoll, Neil Sarnak, Daniel Dominic Sleator, and Robert Endre Tarjan. "Making Data Structures Persistent". In: *Journal of Computer and System Sciences* 38(1) (1989), pp. 86–124.

[EM94] Thomas Eiter and Heikki Mannila. *Computing Discrete Fréchet Distance*. Tech. rep. CD-TR 94/64. Christian Doppler Laboratory for Expert Systems, TU Vienna, Austria, 1994.

[Epp+92] David Eppstein, Zvi Galil, Raffaele Giancarlo, and Giuseppe F. Italiano. "Sparse Dynamic Programming I: Linear Cost Functions". In: *Journal of the ACM* 39(3) (1992), pp. 519–545.

[ERV14] Matthias Englert, Heiko Röglin, and Berthold Vöcking. "Worst Case and Probabilistic Analysis of the 2-Opt Algorithm for the TSP". In: *Algorithmica* 68(1) (2014), pp. 190–264.

[Ets15] Michael Etscheid. "Performance Guarantees for Scheduling Algorithms Under Perturbed Machine Speeds". In: *Discrete Applied Mathematics* 195 (2015), pp. 84–100.

[FG85] Alan M. Frieze and Geoffrey R. Grimmett. "The Shortest-Path Problem for Graphs with Random Arc-Lengths". In: *Discrete Applied Mathematics* 10 (1985), pp. 57–77.

[GMN15] Archontia C. Giannopoulou, George B. Mertzios, and Rolf Niedermeier. "Polynomial Fixed-parameter Algorithms: A Case Study for Longest Path on Interval Graphs". In: *Proc. 10th International Symposium on Parameterized and Exact Computation (IPEC'15)*. 2015, pp. 102–113.

[GO95] Anka Gajentaan and Mark H. Overmars. "On a Class of $O(n^2)$ Problems in Computational Geometry". In: *Computational Geometry: Theory and Applications* 5(3) (1995), pp. 165–185.

[God91] Michael Godau. "A Natural Metric for Curves - Computing the Distance for Polygonal Chains and Approximation Algorithms". In: *Proc. 8th Annual Symposium on Theoretical Aspects of Computer Science (STACS'91)*. 1991, pp. 127–136.

[GS13] Joachim Gudmundsson and Michiel Smid. "Fréchet Queries in Geometric Trees". In: *Proc. 21st Annual European Symposium on Algorithms (ESA'13)*. 2013, pp. 565–576.

[Hag98] Torben Hagerup. "Sorting and Searching on the Word RAM". In: *Proc. 15th Annual Symposium on Theoretical Aspects of Computer Science (STACS'98)*. 1998, pp. 366–398.

[Hir77] Daniel S. Hirschberg. "Algorithms for the Longest Common Subsequence Problem". In: *Journal of the ACM* 24(4) (1977), pp. 664–675.

[HM75] J. W. Hunt and M. D. McIlroy. *An Algorithm for Differential File Comparison*. Computing Science Technical Report 41. Bell Laboratories, 1975.

[HPR11] Sariel Har-Peled and Benjamin Raichel. "The Fréchet Distance Revisited and Extended". In: *Proc. 27th Annual Symposium on Computational Geometry (SoCG'11)*. 2011, pp. 448–457.

[HS77] James W. Hunt and Thomas G. Szymanski. "A Fast Algorithm for Computing Longest Subsequences". In: *Communications of the ACM* 20(5) (1977), pp. 350–353.

[Ind02] Piotr Indyk. "Approximate Nearest Neighbor Algorithms for Fréchet Distance via Product Metrics". In: *Proc. 18th Annual Symposium on Computational Geometry (SoCG'02)*. 2002, pp. 102–106.

[IP01] Russell Impagliazzo and Ramamohan Paturi. "On the Complexity of k-SAT". In: *Journal of Computer and System Sciences* 62(2) (2001), pp. 367–375.

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. "Which Problems Have Strongly Exponential Complexity?" In: *Journal of Computer and System Sciences* 63(4) (2001), pp. 512–530.

[IR09] Costas S. Iliopoulos and M. Sohel Rahman. "A New Efficient Algorithm for Computing the Longest Common Subsequence". In: *Theory of Computing Systems* 45(2) (2009), pp. 355–371.

[IS10] Konrad Iwanicki and Maarten van Steen. "Gossip-Based Self-Management of a Recursive Area Hierarchy for Large Wireless SensorNets". In: *IEEE Transactions on Parallel and Distributed Systems* 21(4) (2010), pp. 562–576.

[Jac89] Guy Jacobson. "Space-efficient Static Trees and Graphs". In: *Proc. 30th Annual IEEE Symposium on Foundations of Computer Science (FOCS'89)*. 1989, pp. 549–554.

[Jak14] Sune K. Jakobsen. "Information Theoretical Cryptogenography". In: *Proc. 41st International Colloquium on Automata, Languages, and Programming (ICALP'14)*. 2014, pp. 676–688.

[Jel+07] Márk Jelasity, Spyros Voulgaris, Rachid Guerraoui, Anne-Marie Kermarrec, and Maarten van Steen. "Gossip-Based Peer Sampling". In: *ACM Transactions on Computer Systems* 25(3) (2007).

[JM02] David S. Johnson and Lyle A. McGeoch. "Experimental Analysis of Heuristics for the STSP". In: *The Traveling Salesman Problem and its Variations*. Ed. by Gregory Gutin and Abraham P. Punnen. Kluwer Academic Publishers, 2002. Chap. 9.

[JM97] David S. Johnson and Lyle A. McGeoch. "The Traveling Salesman Problem: A Case Study". In: *Local Search in Combinatorial Optimization*. Ed. by Emile Aarts and Jan Karel Lenstra. John Wiley & Sons, 1997. Chap. 8.

[JO16] Sune K. Jakobsen and Claudio Orlandi. "How To Bootstrap Anonymous Communication". In: *Proc. 7th ACM Conference on Innovations in Theoretical Computer Science (ITCS'16)*. 2016, pp. 333–344.

[Kle08] Jon M. Kleinberg. "The Convergence of Social and Technological Networks". In: *Communications of the ACM* 51 (2008), pp. 66–72.

[KM15] Marvin Künnemann and Bodo Manthey. "Towards Understanding the Smoothed Approximation Ratio of the 2-Opt Heuristic". In: *Proc. 42nd International Colloquium on Automata, Languages, and Programming (ICALP'15), Part I*. 2015, pp. 859–871.

[KO07] David Karger and Krzysztof Onak. "Polynomial Approximation Schemes for Smoothed and Random Instances of Multidimensional Packing Problems". In: *Proc. 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'07)*. 2007, pp. 1207–1216.

[Kos04] Adrian Kosowski. "An Efficient Algorithm for the Longest Tandem Scattered Subsequence Problem". In: *Proc. 11th International Conference on String Processing and Information Retrieval (SPIRE'04)*. 2004, pp. 93–100.

[KPP16] Tsvi Kopelowitz, Seth Pettie, and Ely Porat. "Higher Lower Bounds from the 3SUM Conjecture". In: *Proc. 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'16)*. 2016, pp. 1272–1287.

[Lig99] Thomas M. Liggett. *Stochastic Interacting Systems: Contact, Voter and Exclusion Processes*. Springer, 1999.

[LMS11] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. "Lower Bounds Based on the Exponential Time Hypothesis". In: *Bulletin of the EATCS* 105 (2011), pp. 41–72.

[Lue78] George S. Lueker. "A Data Structure for Orthogonal Range Queries". In: *Proc. 19th Annual IEEE Symposium on Foundations of Computer Science*. SFCS '78. 1978, pp. 28–34.

[Mah+11] Anil Maheshwari, Jörg-Rüdiger Sack, Kaveh Shahbaz, and Hamid Zarrabi-Zadeh. "Fréchet Distance with Speed Limits". In: *Computational Geometry* 44(2) (2011), pp. 110–120.

[Mat+12] Miguel Matos, Valerio Schiavoni, Pascal Felber, Rui Oliveira, and Etienne Riviere. "BRISA: Combining Efficiency and Reliability in Epidemic Data Dissemination". In: *Proc. 26th International Parallel Distributed Processing Symposium (IPDPS'12)*. 2012, pp. 983–994.

[Mat01] Jiří Matoušek. "On Directional Convexity". In: *Discrete & Computational Geometry* 25(3) (2001), pp. 389–403.

[MI13] Nan Ma and Prakash Ishwar. "The Infinite-Message Limit of Two-Terminal Interactive Source Coding". In: *IEEE Transactions on Information Theory* 59(7) (2013), pp. 4071–4094.

[MIG12] Nan Ma, Prakash Ishwar, and Piyush Gupta. "Interactive Source Coding for Function Computation in Collocated Networks". In: *IEEE Transactions on Information Theory* 58(7) (2012), pp. 4289–4305.

[Mit99] Joseph S. B. Mitchell. "Guillotine Subdivisions Approximate Polygonal Subdivisions: A Simple Polynomial-Time Approximation Scheme for Geometric TSP, k-MST, and Related Problems". In: *SIAM Journal on Computing* 28(4) (1999), pp. 1298–1309.

[MM85] Webb Miller and Eugene W. Myers. "A File Comparison Program". In: *Software: Practince and Experience* 15(11) (1985), pp. 1025–1040.

[MP14] Dániel Marx and Michal Pilipczuk. "Everything You Always Wanted to Know About the Parameterized Complexity of Subgraph Isomorphism (But Were Afraid to Ask)". In: *Proc. 31st International Symposium on Theoretical Aspects of Computer Science (STACS'14)*. 2014, pp. 542–553.

[MP80] William J. Masek and Mike Paterson. "A Faster Algorithm Computing String Edit Distances". In: *Journal of Computer and System Sciences* 20(1) (1980), pp. 18–31.

[MP99] Mario E. Munich and Pietro Perona. "Continuous Dynamic Time Warping for Translation-Invariant Curve Alignment with Applications to Signature Verification". In: *Proc. 7th IEEE International Conference on Computer Vision (ICCV'99)*. 1999, pp. 108–115.

[MR11] Bodo Manthey and Heiko Röglin. "Smoothed Analysis: Analysis of Algorithms Beyond Worst Case". In: *it – Information Technology* 53(6) (2011), pp. 280–286.

[MR13] Bodo Manthey and Heiko Röglin. "Worst-Case and Smoothed Analysis of k-Means Clustering with Bregman Divergences". In: *Journal of Computational Geometry* 4(1) (2013), pp. 94–132.

[MV13] Bodo Manthey and Rianne Veenstra. "Smoothed Analysis of the 2-Opt Heuristic for the TSP: Polynomial Bounds for Gaussian Noise". In: *Proc. 24th Annual International Symposium on Algorithms and Computation (ISAAC'13)*. 2013, pp. 579–589.

[Mye86] Eugene W. Myers. "An O(ND) Difference Algorithm and Its Variations". In: *Algorithmica* 1(2) (1986), pp. 251–266.

[Nav01] Gonzalo Navarro. "A Guided Tour to Approximate String Matching". In: *ACM Computing Surveys* 33(1) (2001), pp. 31–88.

[NKY82] Narao Nakatsu, Yahiko Kambayashi, and Shuzo Yajima. "A Longest Common Subsequence Algorithm Suitable for Similar Text Strings". In: *Acta Informatica* 18(2) (1982), pp. 171–179.

[Pap77] Christos H. Papadimitriou. “The Euclidean Traveling Salesman Problem is NP-complete”. In: *Theoretical Computer Science* 4(3) (1977), pp. 237–244.

[Pat+05] Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. “An Improved Exponential-Time Algorithm for k-SAT”. In: *Journal of the ACM* 52(3) (2005), pp. 337–364.

[Pat08] Mihai Patrascu. “Succincter”. In: *Proc. 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS'08)*. 2008, pp. 305–313.

[Pat10] Mihai Patrascu. “Towards Polynomial Lower Bounds for Dynamic Problems”. In: *Proc. 42nd Annual ACM Symposium on Theory of Computing (STOC'10)*. 2010, pp. 603–610.

[PD94] Mike Paterson and Vlado Dancík. “Longest Common Subsequences”. In: *Proc. 19th International Symposium on Mathematical Foundations of Computer Science (MFCS'94)*. 1994, pp. 127–142.

[Pit87] Boris Pittel. “On Spreading a Rumor”. In: *SIAM Journal on Applied Mathematics* 47 (1987), pp. 213–223.

[PS72] George Pólya and Gábor Szegő. *Problems and Theorems in Analysis I*. Springer-Verlag Berlin Heidelberg, 1972.

[PW10] Mihai Pătraşcu and Ryan Williams. “On the Possibility of Faster SAT Algorithms”. In: *Proc. 21nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'10)*. 2010, pp. 1065–1075.

[RBN97] Stephen V. Rice, Horst Bunke, and Thomas A. Nartker. “Classes of Cost Functions for String Edit Distance”. In: *Algorithmica* 18(2) (1997), pp. 271–280.

[RSL77] Daniel J. Rosenkrantz, Richard E. Stearns, and Philip M. Lewis II. “An Analysis of Several Heuristics for the Traveling Salesman Problem”. In: *SIAM Journal on Computing* 6(3) (1977), pp. 563–581.

[RVW13] Liam Roditty and Virginia Vassilevska Williams. “Fast Approximation Algorithms for the Diameter and Radius of Sparse Graphs”. In: *Proc. 45th Annual ACM Symposium on Symposium on Theory of Computing (STOC'13)*. 2013, pp. 515–524.

[Sad07] Kunihiko Sadakane. “Compressed Suffix Trees with Full Functionality”. In: *Theory of Computing Systems* 41(4) (2007), pp. 589–607.

[SC07] Stan Salvador and Philip Chan. “Toward Accurate Dynamic Time Warping in Linear Time and Space”. In: *Intelligent Data Analysis* 11(5) (2007), pp. 561–580.

[SC78] Hiroaki Sakoe and Seibi Chiba. “Dynamic Programming Algorithm Optimization for Spoken Word Recognition”. In: *IEEE Transactions on Acoustics, Speech and Signal Processing* 26(1) (1978), pp. 43–49.

[SM97] João Carlos Setubal and João Meidanis. *Introduction to Computational Molecular Biology*. Computer Science Series. PWS Pub., 1997.

[ST04] Daniel A. Spielman and Shang-Hua Teng. “Smoothed Analysis of Algorithms: Why the Simplex Algorithm Usually Takes Polynomial Time”. In: *Journal of the ACM* 51(3) (2004), pp. 385–463.

[ST09] Daniel A. Spielman and Shang-Hua Teng. "Smoothed Analysis: An Attempt to Explain the Behavior of Algorithms in Practice". In: *Communications of the ACM* 52(10) (2009), pp. 76–84.

[Tar79] Robert Endre Tarjan. "A Class of Algorithms which Require Nonlinear Time to Maintain Disjoint Sets". In: *Journal of Computer and System Sciences* 18(2) (1979), pp. 110–127.

[VW15] Virginia Vassilevska Williams. "Hardness of Easy Problems: Basing Hardness on Popular Conjectures such as the Strong Exponential Time Hypothesis (Invited Talk)". In: *Proc. 10th International Symposium on Parameterized and Exact Computation (IPEC'15)*. 2015, pp. 17–29.

[WF74] Robert A. Wagner and Michael J. Fischer. "The String-to-String Correction Problem". In: *Journal of the ACM* 21(1) (1974), pp. 168–173.

[Wil05] Ryan Williams. "A New Algorithm for Optimal 2-Constraint Satisfaction and its Implications". In: *Theoretical Computer Science* 348(2) (2005), pp. 357–365.

[Wil78] D. E. Willard. "Predicate-Oriented Database Search Algorithms". Report TR-20-78. PhD thesis. Cambridge, MA: Aiken Comput. Lab, Harvard Univ., 1978.

[Wu+90] Sun Wu, Udi Manber, Gene Myers, and Webb Miller. "An O(NP) Sequence Comparison Algorithm". In: *Information Processing Letters* 35(6) (1990), pp. 317–323.

[WW10] Virginia Vassilevska Williams and Ryan Williams. "Subcubic Equivalences Between Path, Matrix and Triangle Problems". In: *Proc. 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS'10)*. 2010, pp. 645–654.

[Yuk98] Joseph E. Yukich. *Probability Theory of Classical Euclidean Optimization Problems*. Vol. 1675. Lecture Notes in Mathematics. Springer, 1998.

Notes

All figures included in this thesis have been produced using Inkscape. The layout of the document is a slight adaptation of the Masters/Doctoral Thesis style by Steven Gunn and Sunil Patel with version 2.0 major modifications by Vel and Johannes Bötcher, licensed under CC BY-NC-SA 3.0 (`http://creativecommons.org/licenses/by-nc-sa/3.0/`)