METRIC VIOLATION AND SIMILARITY

by

Chenglin Fan

APPROVED BY SUPERVISORY COMMITTEE:

Benjamin Raichel, Chair

Sergey Bereg

Ovidiu Daescu

Kyle Fox

To my family, and to all those who helped me in my life.

METRIC VIOLATION AND SIMILARITY

by

CHENGLIN FAN, BS, ME

DISSERTATION

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY IN

COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS

May 2019

ACKNOWLEDGMENTS

First and foremost, I want to thank my advisor Benjamin Raichel. His instruction was the reason I did PhD studies at UT Dallas, and I try to follow his example in my own research and presentation.

Many works in this dissertation are due to collaborations with Benjamin Raichel and Gregory Van Buskirk. It has been an honor to work with them and others, and I can only hope to have more collaborations in the future.

My research was partially supported by NSF CRII Award 1566137 and CAREER Award 1750780, so thanks to NSF.

Thank you to the theory faculty at UTD who offered me with a significant amount of valuable advice, Sergey Bereg, Ovidiu Daescu, Kyle Fox, and Benjamin Raichel, who are also my dissertation committee members.

Thank you to the former and current student members of UTD's theory group in particular Saloni Agarwal, Bogdan Armaselu, Harish Arunachalam, Jon Crain, Hongyao Huang, Xinyi Li, Jiashuai Lu, Hemant Malik, Rashika Mishra, Ka Yaw Teo and Gregory Van Buskirk. My discussions with all of you were important for helping me figure out what to do next.

Finally, I want to thank my family who supports me as I move on to the next stage of life.

April 2019

METRIC VIOLATION AND SIMILARITY

Chenglin Fan, PhD
The University of Texas at Dallas, 2019

Supervising Professor: Benjamin Raichel, Chair

Metric data plays an important role in various settings, for example, in metric-based indexing, clustering, classification, and approximation algorithms in general. Often such tasks require the data to be metric, though for various reasons this basic property may not be satisfied. The first topic we consider is the metric violation distance problem, which seeks to minimally modify the data to make it metric, and thereby finding the nearest metric data set. Three variants are considered, one admitting a polynomial time algorithm. The other variants are shown to be APX-hard, and an approximation is given. We also introduce and give results for the generalized metric violation distance problem, where there are no longer constraints on all pairs.

The second topic concerns the fundamental computational task of assessing the similarity of ordered data sets, i.e. the similarity of two trajectories. Here we introduce the Fréchet gap distance, which in some ways better captures the intuitive notions of curve similarity when comparing to the previous Standard Fréchet distance. In addition, a polynomial time exact algorithm and a much faster $1 + \epsilon$ approximation to compute this measure will be given.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The amount of data produced every day is growing at an ever increasing rate. A large fraction of this data can either be viewed geometrically or is inherently geometric in 3 nature, for example spatial data obtained from the plethora of cheap sensors. This geometry can be leveraged to help analyze this massive amount of data, and to help with basic tasks like assessing how similar two data items are, and the shape of the data. Thus geometric data analysis has wide applications throughout data science. In this dissertation, we focus on two types of geometric data, metric data and ordered geometric data.

In data science and machine learning, often data items can be represented as points. These points can be low dimensional if representing say a physical measurement in three-space. Alternatively, they can be high dimensional if one views each word in a web page or each field in a database record as a different coordinate of a point. Then various standard computational tasks, such as finding a linear subspace of best fit or clustering the data [1], involve applying geometric tools and insights at their core. Often such tasks require the data to be metric, for instance, proximity search in data science usually involves a metric distance function. For various reasons, though, this basic property may not be satisfied. Thus one main focus of this dissertation concerns metric data, and studies minimally modifying (non-metric) data to make it metric, that is finding the nearest metric data set.

Often data sets include additional structure beyond the data points. Here we consider the case when the points are ordered, thus defining polygonal curves. Such data sets arise naturally in many real world applications, for instance when sampling the location of an object over time. With the development of location tracking technologies such as GPS, trajectory analysis has become a topic of increasing importance, with connections to shape analysis. Shape analysis is one of the classical topics in computational geometry, and has applications in areas such as bioinformatics [2], signature matching [3], and more. Here we consider a basic

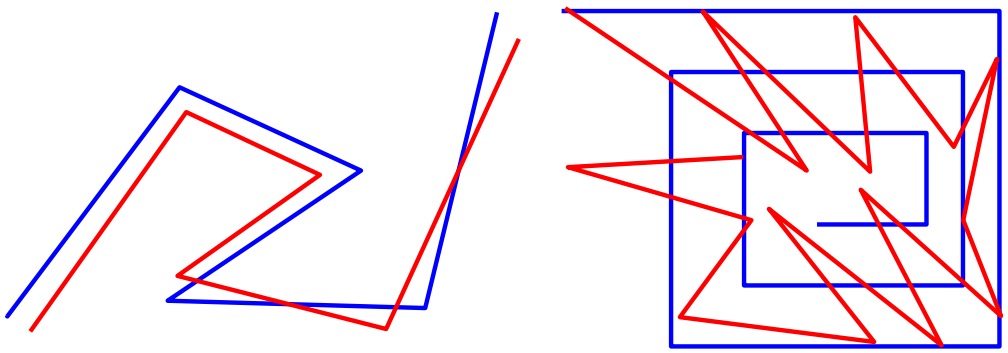


Figure 1.1: In both cases the curves occupy roughly the same space, but the pair on the left appear more similar than the pair on the right.

question, to what extent do two given curves resemble each other? For example, arguably the two curves in the Figure 1.1(a) look more similar than those in Figure 1.1(b). This leads us to ask, what distance function should be used to measure similarity? Thus another focus of this dissertation concerns defining and computing an appropriate distance between curves.

1.1 Metric Violation

Given a large collection of data points for which there is some underlying notion of "distance" between pairs of points, one naturally may wish to perform any number of computational tasks over the data by using these distances (e.g. clustering). The ability to perform these tasks highly depends on the structure these distances obey. There are many settings where the underlying distances arise from a metric space or are at least well modeled by one. Such cases are fortuitous, as certain tasks become provably easier over metric data (e.g. approximating the optimal TSP tour [4]). Here we consider the *metric violation distance* problem, denoted MVD, where given a collection of distances between data points (forming a semi-metric), we seek the smallest sized set of distance values that can be (arbitrarily) modified to produce a metric space overall. Brickell *et al.* [5] studied the metric nearness problem. Similar to MVD, the input is a semi-metric (i.e. triangle inequalities may be violated) and the goal is to find

the closest metric space. The difference is that for MVD a closest metric space is defined by minimizing the number of changed values (irrespective of how much they are changed), and in their case it is defined by minimizing the sum of the changes in value (irrespective of how many are changed). In other words, metric nearness is the linear programming relaxation of MVD, and thus is not NP-hard like MVD. So while metric nearness is semantically similar, the challenges and approach are different, and thus [5] focus on other aspects such as varying the norm of the objective and experimental results.

The MVD problem can be equivalently formulated in terms of graphs. In computer science, graphs are a fundamental structure used to model pairwise relations between objects, abstracting many natural structures in the real world, such as transportation or computer networks. Given data points equipped with a distance function, if we view each data point as a vertex, and the distance between each pair of vertices as the weight of an edge, then the distance function is modeled by an undirected and complete weighted graph. This graph can be stored as an adjacency matrix M, where each entry corresponds to an edge. We call a graph $G(V, E)$ a metric graph if it is an undirected weighted graph, and $\forall (a, b) \in E$, $w(a, b) \leq d(a, b)$, where $w(a, b)$ denotes the weight of edge (a, b), and $d(a, b)$ denotes the shortest path distance between vertex a and b in G. In the graph version of MVD which we study in Chapter 3 in this dissertation, one is given a complete, undirected, and positively-weighted graph $G(V, E)$, and the goal is to compute a minimum size set S of edges whose weights can be modified to convert G into a metric graph. That is, for every triangle in the graph, its weights must satisfy the triangle inequality. In Chapter 4, we consider how the problem changes when we remove the requirement that G is complete, which we then call the *generalized metric violation distance* problem, denoted GMVD.

Since MVD can be modeled as a graph problem, we now briefly review a few fundamental related graph problems. Consider the all pairs shortest paths problem (APSP) [6]. The Floyd–Warshall algorithm is a simple algorithm that runs in $O(n^3)$ time, making it an

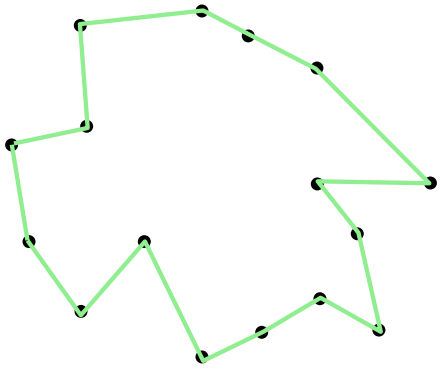


Figure 1.2: An instance of TSP is shown.

ideal algorithm for dense graphs, although a faster $O(n^3 \log^3 \log n / \log^2 n)$ time algorithm is known [7]. Unless P=NP, there are no polynomial time algorithms for the various NP-hard graph problems such as vertex cover [6], MULTICUT [8], and TSP [9]. In the traveling salesperson problem (TSP) [9], one is given a set of vertices (sites), and distances between each pair of vertices, and the goal is to find a shortest route that visits each vertex exactly once and then returns to the start vertex (an example is shown in Figure 1.2). For general positively weighted graphs, it is a standard exercise to show by reduction from Hamiltonian cycle, that TSP cannot be approximated within *any* constant factor unless P=NP. On the other hand if the graph satisfies the triangle inequality, i.e. metric TSP, then Christofides[9] gave a 1.5 approximation. The formal definition of metric, vertex cover, and MULTICUT will be given in the next chapter, along with more details.

1.2 Similarity between Curves

Measuring the similarity of two trajectories is a fundamental computational task. Polygonal curves arise naturally in the modeling of a number computational problems, and for such problems assessing the similarity between two curves is one of the most fundamental tasks. There are several competing measures for defining curve similarity. Hausdorff distance [10] and Fréchet distance [11] are two important measures of similarity between two curves, which we summarize below.

- The Hausdorff distance between two point sets is the maximum distance of point in one set to its nearest point in the other set. It can be used to measure how similar two sequences of points are; however, the disadvantage is that it only takes into account the sets of points but not the ordering. A straightforward, brute force algorithm computes the Hausdorff distance in $O(mn)$ time; by using Voronoi diagrams, the running time can be reduced to $O((m+n)\log(m+n))$ [10] in low-dimensional space, where m, n respectively denote the sizes of the two curves.

- The Fréchet distance between polygonal curves is typically illustrated as follows. Let the two polygonal curves be denoted π and σ, with m and n vertices respectively. Imagine a man and a dog are respectively placed at the starting vertices of π and σ, and they must each move continuously along their curves to their respective ending points. The man and dog are connected by a leash, and the Fréchet distance is the minimum leash length required over all possible walks of the man and dog, where the man and dog can independently control their speed but cannot backtrack. The advantage of Fréchet distance is that takes into account the continuous shape of the curves rather than just the set of points in space they occupy. Alt and Godau [11] presented an $O(mn \log mn)$ time algorithm to compute the Fréchet distance.

To illustrate the difference between Hausdorff and Fréchet distance, consider Figure 1.1. Both pair of curves on the left and on the right have relatively small Hausdorff distance. However, the figure on the left has much smaller Fréchet distance than that on the right.

In the later part of this dissertation, we introduce a new variant of Fréchet distance, which we call the Fréchet Gap distance. In the man and dog analogy, the Fréchet Gap distance minimizes the difference of the longest and smallest leash lengths used over the entire walk. This measure in some ways better captures our intuitive notions of curve similarity, for example giving distance zero to translated copies of the same curve.

1.3 Organization and Contributions

The rest of the dissertation proceeds as follows: We begin by giving a formal overview of the definitions and tools used throughout the dissertation in Chapter 2. Chapter 3 details the metric violation distance problem. Chapter 4 generalizes the metric violation distance problem to general graphs, where the distance graph is no longer required to be complete. Chapter 5 considers the Fréchet Gap distance, a new measure of similarity between curves. Below we provide a high level sketch of the important aspects of each part.

In Chapter 3, which is largely based on the paper [12], we detail the metric violation distance problem, denoted MVD, and break it into three cases to help with the analysis: 1) MVD where one is only allowed to decrease values, 2) MVD where one is only allowed to increase values, and 3) the full MVD problem. The decrease only case is shown to admit a polynomial time algorithm. The other variants are shown to be APX-hard, and at least as hard to approximate as Vertex Cover, and an $O(OPT^{1/3})$-approximation is given, where OPT is the optimal solution size. We also prove necessary and sufficient conditions on the solution, reducing the problem to a purely combinatorial one.

In Chapter 4, which is largely based on the ArXiv version our paper [13] currently in submission, we extend the structural results from MVD to General Metric Violation Distance (denoted as GMVD), where the distance graph is no longer required to be complete. We give a polynomial-time approximation-preserving reduction from MULTICUT to GMVD. The best known approximation factor for MULTICUT is $O(\log n)$, and for GMVD we give an $O(c \log n)$-approximation, where c is a parameter called the graph deficit, defined later.

In Chapter 5, which is largely based on the paper [14], we introduce the Fréchet gap distance, which in some ways better captures our intuitive notions of curve similarity when compared to the previous Standard Fréchet distance. For the computation of the Fréchet gap distance, we provide an exact algorithm with $O(n^5 \log n)$ running time, and a more efficient $(1+\varepsilon)$-approximation algorithm running in near quadratic time.

CHAPTER 2

PRELIMINARIES

We now discuss some important concepts in graph theory and geometry, which are used throughout the dissertation.

2.1 Metric Distance

Definition 2.1.1 (Metric)**.** *A metric on a set X is a function*

$$f : X \times X \to \mathbb{R}$$

such that for any $a, b, c \in X$, the following holds:

$f(a,b) \geq 0$	*Non-Negativitiy*
$f(a,b) = 0 \Leftrightarrow a = b$	*Identity of Indiscernibles*
$f(a,b) = f(b,a)$	*Symmetry*
$f(a,c) \leq f(a,b) + f(b,c)$	***Triangle Inequality***

If the first three properties hold, but not necessarily the triangle inequality, then f is called a *semi-metric* on X.

As a simple example of the power of metric properties, consider the famous NP-Complete traveling salesperson problem (TSP). For general positively weighted graphs, it is a standard exercise to show TSP cannot be approximated within *any* constant factor unless P=NP. On the other hand if the graph satisfies the triangle inequality, Christofides' algorithm gives a 1.5-approximation [9].

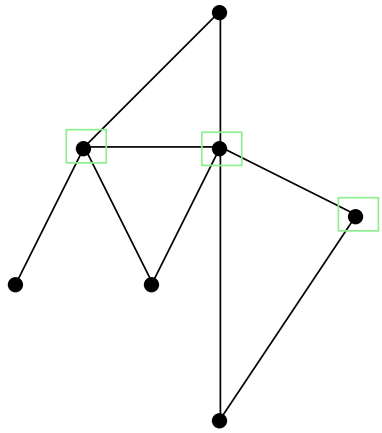


Figure 2.1: A vertex cover instance $G = (V, E)$, and three vertices identified by green boxes which cover the edges of the graph.

2.2 Vertex Cover

In the Vertex Cover problem, one is given an undirected graph $G = (V, E)$, and the goal is to find a minimum size subset $V' \subseteq V$ such that each edge $e \in E$ is incident to at least one vertex in V'.

One can achieve a 2-approximation by repeatedly adding both endpoints of an arbitrary edge to the vertex cover, and then removing them and their adjacent edges from the graph. Moreover, assuming the Unique Games Conjecture (UGC), minimum vertex cover is hard to approximate within a $2 - \varepsilon$ factor for any $\varepsilon > 0$ [15]. We will later use Vertex Cover to prove hardness of the MVD problem.

2.3 MULTICUT and Length Bounded Cut

In the MULTICUT problem, one is given a graph G with k marked (s_i, t_i) vertex pairs, and the goal is to find a minimum size subset of edges S such that there is no path in $G \setminus S$ between s_i and t_i for any $1 \leq i \leq k$.

MULTICUT has been extensively studied, both for directed and undirected graphs. For undirected graphs, the problem captures vertex cover, even when G is a tree, and hence is

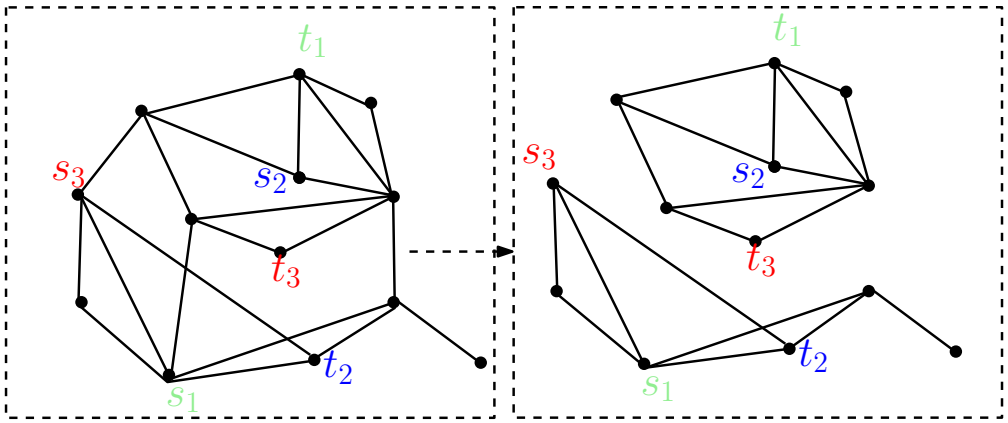


Figure 2.2: A MULTICUT instance $G = (V, E)$, and three marked vertex pairs, (s_1, t_1), (s_2, t_2), and (s_3, t_3). The right figure shows that by removing three edges E', then there is no path in $G \setminus E'$ between s_i and t_i for any $1 \leq i \leq 3$.

APX-Hard. Moreover assuming UGC there is no constant factor approximation [16]. In general the best known approximation factor is $O(\log k)$ [8], which improves to an $O(r)$-approximation when G excludes K_r as a minor [17]. For directed graphs the problem is more challenging. It is hard to approximate within a $2^{\Omega(\log^{1-\epsilon} n)}$ factor [18] assuming NP≠ZPP (zero-error probabilistic polynomial time), and the best approximation factor known is $\tilde{O}(n^{11/23})$ [19], where n is number of vertices in G.

Another closely related problem is Length Bounded Cut (LB-CUT), where given a value L and an (s, t) pair in a graph G, the goal is to delete the minimum number of edges such that there is no path between s and t whose number of edges is $\leq L$. For LB-CUT in undirected graphs, the best known approximation factor is $O(\min\{L, n^2/L^2, \sqrt{m}\})$ [20], when G has n vertices and m edges. For any fixed L, Lee [21] showed that it is hard to approximate within a factor of $\Omega(\sqrt{L})$ in undirected graphs, and within a factor of $\Omega(L)$ for directed graphs. Kuhnle *et al.* [22] showed a similar result for the directed case of Length Bounded Multicut under the assumption NP≠BPP (bounded-error probabilistic polynomial time).

CHAPTER 3

METRIC VIOLATION DISTANCE

3.1 Introduction

Suppose you are given a collection of data points, along with a corresponding set of distances (i.e. dissimilarity scores) between every pair of points. The ability to perform various computational tasks over this collection of data points depends highly on what structural properties these distances obey. Perhaps the most often considered and desired are the basic metric requirements, most notably the triangle inequality (as the other metric requirements are often trivially satisfied). Metric data plays a critical role in various areas such as metric-based indexing, clustering, classification, and approximation algorithms in general. Moreover, given our points come from a metric space, we can potentially obtain additional properties by applying tools such as metric embeddings.

Here we consider the *metric violation distance* problem, denoted MVD, where given a collection of distances between data points (forming a semi-metric), we seek the smallest sized set of distance values that can be (arbitrarily) modified to produce a metric space overall. One can consider the resulting metric as the nearest neighbor from the space of all metrics, and thus one interpretation of the MVD problem is as a measurement of how close the input is to representing a metric space. A solution to this problem is thus desirable, since when the distance is small, one could potentially use the nearest metric as a proxy for the original set of distances, unlocking all the above benefits of metric spaces. Alternatively, the MVD problem arises naturally in the following setting. Suppose you are collecting experimental data by measuring distances between a collection of objects, where you know the distances should form a metric space. The collected data might be non-metric for a variety of reasons, such as measurement error, or incomplete or corrupted data entries. In this setting the MVD problem can thus be used as a means to recover the true underlying data set.

In this chapter we initiate the study of the MVD problem. We prove the problem is APX-hard, and provide approximation algorithms for different variants.

Related Work. Brickell *et al.* [5] studied the metric nearness problem. Similar to MVD, the input is a semi-metric (i.e. triangle inequalities may be violated) and the goal is to find the closest metric space. The difference is that for MVD a closest metric space is defined by minimizing the number of changed values (irrespective of how much they are changed), and in their case it is defined by minimizing the sum of the changes in value (irrespective of how many are changed). In other words, metric nearness is the linear programming relaxation of MVD, and thus is not NP-hard like MVD. So while metric nearness is semantically similar, the challenges and approach are different, and thus [5] focus on other aspects such as varying the norm of the objective and experimental results. Our paper [12] and Gilbert and Jain [23] respectively formulated the Metric Violation Distance (MVD) and the sparse metric repair (SMR) problems, however, in their paper [23], the approximation ratio in the worst case can be $O(n)$.

The MVD problem also bears a clear resemblance to metric embeddings, though one level removed. In a standard metric embedding problem there are two collections of metric spaces of interest, call them S (for source) and T (for target). Given a metric space $X \in S$ the goal is then to injectively map the points from X into the metric space $Y \in T$ which preserves distances as best as possible, when measured by the distortion (roughly the maximum any distance is scaled). Typically the collection T has some desirable structural property that in general is lacking from metrics in S, and thus metric embeddings are a tool to gain these structural benefits. For example, the famous result of Bourgain [24] shows that any arbitrary n-point metric embeds into $\ell_2^{O(\log n)}$ with $O(\log n)$ distortion, and more generally into $\ell_p^{O(\log^2 n)}$ for any p, due to Linial *et al.* [25]. As metric embeddings are too broad to fully cover here, we refer the reader to the surveys [26, 27]. Thus MVD can be viewed as a type of "embedding" problem, where S is the collection of all semi-metrics and T is the collection of all metrics.

The larger difference between metric embedding and MVD, is that for metric embedding the typical goal is to minimize distortion, whereas in our case the goal is to minimize the number of (arbitrarily) changed distances. Thus our work is more akin to isometrically embedding with outliers. Recently Sidiropoulos *et al.* [28] considered isometrically embedding metrics into ultrametrics, trees, and Euclidean space, while minimizing the number of outlier points. Despite their input starting out as a metric, the problems they consider similarly involve satisfying inequalities with a small number of distances (for example, ultrametrics replace the sum with max in the triangle inequality). The larger difference is that since their notion of outliers is point based, when a violated inequality is identified the points can simply be thrown out, whereas in our case the values need to be corrected, and it is not possible to locally determine how to do so (see Figure 3.1, discussed in detail below). Moving towards distance based notions of outliers, the problem of embeddings with *slack* was previously considered. Rather than isometrically embedding, the goal moves back to finding minimum distortion embeddings, but subject to allowing an ε fraction of the distances to be arbitrarily distorted. For example, Abraham *et al.* [29, 30] showed that any metric space can be embedded into $\ell_p^{O(\log^2(1/\varepsilon))}$ for any $p \geq 1$ with $O(\log(1/\varepsilon))$ distortion, i.e. a slack version of Bourgain's theorem.

There are a number of other more loosely related problems. In the matrix completion problem, from the machine learning community, one is given a distance matrix with many missing entries, and the goal is to fill in the missing entries such that the resulting matrix has as low a rank as possible [31]. In the distance realization problem, from the networking community, given a distance matrix the goal is to find a corresponding weighted graph (possibly restricted to be a tree) which minimizes the sum of edge lengths [32]. While related, these problems do not capture the specific challenges of MVD.

Contribution and Results. From a first glance it may not be apparent why the MVD problem is so challenging. Violated triangle inequalities can trivially be identified, so why not

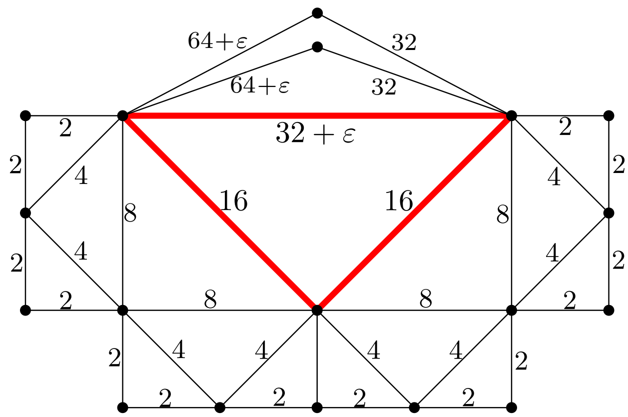


Figure 3.1: A single red thickened violated triangle. Increasing a 16 edge causes a chain of violations, which more generally can be logarithmic in size. Many edges are omitted for simplicity.

fix them one at a time. The first issue is that fixing a triangle may require violating another, as shown in Figure 3.1. In this example, the larger issue is that fixing the inequality requires deciding whether to increase smaller weight edges or decrease the larger one (or both), as well as deciding how much to change them. Increasing one smaller weight edge by ε creates only one new violation, and thus locally appears preferable, as decreasing the large edge by ε creates two violations. Unfortunately, decreasing the lower weight edge creates a chaining effect ultimately requiring a logarithmic number of fixes. Alternatively one can take a more global approach, treating all the violated inequalities as a system of constraints and then trying to solve that system. The issue with this approach is that the objective is to minimize the number of changes rather than the sum of the changed amounts, which is a red flag as roughly speaking this is the difference between integer programming formulations of many NP-Complete problems and their linear programming relaxations.

Here we take a systematic approach to understanding and handling the challenges of the MVD problem. We consider three variants: 1) MVD where one is only allowed to decrease values, 2) MVD where one is only allowed to increase values, and 3) the full MVD problem. When only decreases are allowed we provide a polynomial time solution, making a connection to the all pairs shortest paths problem. For both the increase only and full cases, we show that

the problem is NP-Complete, and moreover is as hard to approximate as vertex cover. We also give a polynomial time $O(OPT^{1/3})$-approximation for both cases, where OPT denotes the optimal solution size. We first present the approximation algorithm for the increase only case, as it is conceptually simpler, and its results can be used to help understand the approach for the full case. That said, the increase only case is still quite challenging, and requires proving necessary and sufficient conditions on the solution, reducing the problem to a purely combinatorial one with connections to fundamental problems such as block design. Several interesting open problems remain such as whether there is a provable hardness gap between the increase only and full versions, and for either problem whether the gap between approximation and hardness can be narrowed.

3.2 Preliminaries

The MVD problem can be equivalently formulated as either a problem on symmetric matrices or on weighted graphs. For the majority of the thesis the graph formulation is used, however the matrix formulation is also presented as this was the original terminology used by Brickel *et al.* [5] for the similar metric nearness problem, and also more naturally connects to the linear program feasibility check we ultimately use to verify our solution.

3.2.1 Problem Definition.

Let a *dissimilarity matrix* be any square symmetric matrix, where diagonal entries are all zero, and all other entries are positive. For an $n \times n$ dissimilarity matrix M, we refer to each entry M_{ij} as the distance from point i to point j. M is said to be a *metric matrix* if for every triple of distinct indices (i, j, k) it holds that $M_{ik} \leq M_{ij} + M_{jk}$, that is the distances in the matrix represent the interpoint distances of an n point metric space. A *symmetric modification* of a distance M_{ij} to a value α, sets $M_{ij} = M_{ji} = \alpha$.

Problem 3.2.1. (Matrix Metric Violation Distance) *Given a dissimilarity matrix M, compute a minimum size set S of distances which can be symmetrically modified to convert M into a metric matrix D.*

$$\overset{\text{Input: } M=[x_{ij}]}{\begin{bmatrix} 0 & x_{12} & x_{13} & \dots & x_{1n} \\ & 0 & x_{23} & \dots & x_{2n} \\ & & & \ddots & \vdots \\ & & & & 0 \end{bmatrix}} \Longrightarrow \overset{\text{Output: } D=[d_{ij}]}{\begin{bmatrix} 0 & d_{12} & d_{13} & \dots & d_{1n} \\ & 0 & d_{23} & \dots & d_{2n} \\ & & & \ddots & \vdots \\ & & & & 0 \end{bmatrix}}$$

$$\begin{aligned} &\text{minimize} \quad |\{d_{ij} | x_{ij} \neq d_{ij}\}| \\ &\text{subject to} \quad d_{ij} + d_{jk} \geq d_{ik},\ i, j, k = 1, \dots, n. \end{aligned}$$

Alternatively, define a *dissimilarity graph* as any complete, undirected, and positively-weighted graph, and define a *metric graph* to be any dissimilarity graph which is its own metric completion.

Problem 3.2.2. (Graph Metric Violation Distance (MVD)) *Given a dissimilarity graph G, compute a minimum size set S of edges whose weights can be modified to convert G into a metric graph.*

Note that Problem 3.2.2 and Problem 3.2.1 are equivalent problems and an instance of one can be trivially converted into an instance of the other. We thus freely interchange between the two in the text when needed, though with the exception Section 3.2.2, we mainly defer to the graph formulation as it avoids the confusion of symmetric modification, and thus typically use MVD to refer to the graph version.

Two variants of the MVD problem will also be considered in this chapter, one where weights are only allowed to decrease, and the other where weights are only allowed to increase.

Problem 3.2.3. (Graph Metric Violation Decrease Distance (MVDD) *Given a dissimilarity graph G, compute a minimum size set S of edges whose weights can be decreased to covert G into a metric graph.*

Problem 3.2.4. (Graph Metric Violation Increase Distance (MVID)) *Given a dissimilarity graph G, compute a minimum size set S of edges whose weights can be increased to covert G into a metric graph.*

3.2.2 Feasibility Checking.

Let M be an $n \times n$ dissimilarity matrix. Let S be a set of entries from M, namely a set of pairs of distinct integers $\{i, j\}$, with $1 \leq i, j \leq n$. Here we show that checking whether there is a solution to MVD for M which symmetrically modifies only the entries in S, is polynomial time solvable by writing the problem as an instance of linear programming feasibility. Specifically, for each entry M_{ij} we define a new matrix entry $\alpha_{ij} = M_{ij} + x_{ij}$, where x_{ij} is a variable representing the deviation from the original entry. If there is a solution for the MVD problem that only symmetric modifies entries from S, then the α_{ij}'s must satisfy all triangle inequalities, and for all $\{i, j\} \notin S$ we must have $x_{ij} = x_{ji} = 0$. Thus the problem is equivalent to the feasibility of the following linear program.

$$\begin{aligned}
&\alpha_{ij} = M_{ij} + x_{ij} > 0 && \forall \text{ distinct pairs } \{i, j\} \\
&\alpha_{ik} \leq \alpha_{ij} + \alpha_{jk} && \forall \text{ distinct triples } \{i, j, k\} \\
&x_{ij} = x_{ji} = 0 && \forall \{i, j\} \notin S \\
&x_{ij} = x_{ji} && \forall \{i, j\} \in S
\end{aligned}$$

Later in the chapter we consider variants of MVD where entries are only allowed to be increased. Note that the above linear program can trivially be modified to handle this case. Namely modify the last constraint to be $x_{ij} = x_{ji} \geq 0 \ \ \forall \{i, j\} \in S$.

3.2.3 Notation and an Observation.

We now list some notation which will be used for our approximation algorithms. Given a dissimilarity graph $G = (V, E)$, a subgraph $C = (V', E')$ is called a k-cycle if $|V'| = |E'| = k$ and the subgraph is connected with every vertex having degree exactly 2. We often overload this notation and use C to denote either the cyclically ordered list of vertices or edges from this subgraph. Given a k-cycle in G, if the weight of a single edge is strictly larger than sum of the weights of the other edges in the cycle, we say it is an *unbalanced k-cycle*. For a given unbalanced k-cycle, call the largest edge of the cycle the *top edge* and the other edges of the cycle the non-top edges. Call any edge from G connecting two vertices which are non-adjacent in a given cycle, a *chord* of that cycle.

We define three notions of covering unbalanced cycles, which correspond to our three problem variants. Specifically, let $\mathbb{C}$ be any collection of unbalanced cycles from a dissimilarity graph $G = (V, E)$. We say an edge subset $F \subseteq E$ is a (i) *regular cover*, (ii) *non-top cover*, or (iii) *top cover* of $\mathbb{C}$, if F contains at least one (i) edge, (ii) non-top edge, or (iii) top edge of every unbalanced cycle in $\mathbb{C}$. In particular, if $\mathbb{C}$ is the set of *all* unbalanced cycles in G then we say F (i) regular covers, (ii) non-top covers, or (iii) top covers G, respectively.

The following simple lemma implies that all unbalanced cycles must be regular covered for MVD, non-top covered for MVID, and top covered for MVDD.

Lemma 3.2.5. *For any dissimilarity graph G, if there exists an unbalanced k-cycle for $k \geq 4$, then there exists an unbalanced 3-cycle, i.e. an unsatisfied triangle inequality.*

Proof. Let k be the smallest value such that there is an unbalanced k-cycle, and suppose for contradiction that $k \geq 4$. Let C be an unbalanced k-cycle, and let x and y be any two non-top edges of C which are adjacent in C. These two edge form a unique triangle in the complete input graph G with some third edge z (note as $k \geq 4$, z is not in C). Let C' be the cycle obtained from C by removing x and y and adding the edge z. Since there are no

unbalanced 3-cycles, $w(z) \leq w(x) + w(y)$. This implies C' is an unbalanced $k-1$ cycle, which is a contradiction with the definition of k. □

3.3 Metric Violation Distance Complexity

In this section we prove the following decision version of MVD is NP-hard, and moreover the optimization version is APX-hard. The hardness of MVID will then follow by essentially the same proof. Finally, we show that conversely MVDD is polynomial time solvable.

Problem 3.3.1. *Given a dissimilarity graph G, can G be converted into a metric graph by modifying the weights of at most k edges?*

Problem 3.3.2 (Vertex Cover)**.** *Given an undirected graph $G = (V, E)$, is there a subset $V' \subseteq V$ of size $|V'| \leq k$ such that each edge $e \in E$ is incident to at least one vertex in V'?*

The proof of hardness will be by reduction from vertex cover. As our reduction will be approximation preserving, it actually implies MVD is APX-hard, as minimum vertex cover is APX-hard.

Theorem 3.3.3. *Problem 3.3.1 is NP-Complete. Moreover, MVD is APX-hard, and assuming the Unique Games Conjecture, is hard to approximate within a factor of $2 - \varepsilon$ for any $\varepsilon > 0$.*

Proof. Let $G = (V, E)$ be an instance of Problem 3.3.2. We construct a corresponding instance $H = (V \cup \{v_0\}, E')$ of Problem 3.3.1 with weight function $w : E' \to \mathbb{R}^+$, where v_0 is a newly added "apex" vertex. Note by definition H is a complete graph. Partition the edges in E' into three named groups: The set E of edges from G, the set P consisting of all edges adjacent to v_0 called *purple edges*, and the set R of all other edges (i.e. edges in the complement of G) called *red edges*. See Figure 3.2 for an illustration. The idea is to set edge weights such that modifying the weight of a given purple edge to satisfy the triangle

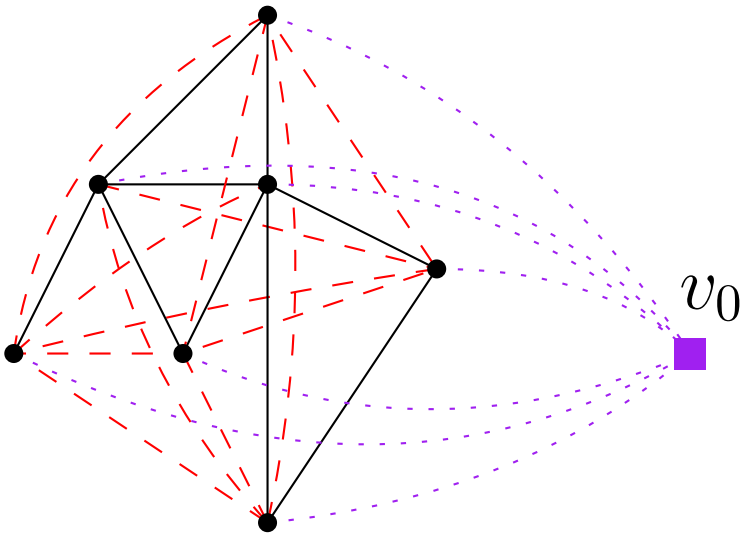


Figure 3.2: Vertex cover reduction: E edges are solid black, red are dashed, and purple are dotted.

inequalities it is involved in, corresponds to selecting the non-apex endpoint to cover its adjacent edges from E. Specifically, for $e \in E$ set $w(e) = 2 + \epsilon$, for $e \in P$ set $w(e) = 1$, and for $e \in R$ set $w(e) = 2$, where ϵ is a sufficiently small constant.

It is easy to verify that any unsatisfied triangle inequality in H must involve both an edge from E and its two adjacent purple edges, that is it must be of the form $w(x, y) = 2 + \epsilon > 1 + 1 = w(v_0, x) + w(v_0, y)$ where the edge $\{x, y\} \in E$. Thus there is a one-to-one correspondence between E and the unsatisfied triangle inequalities. So let F be the set of modified edges from a solution to Problem 3.3.1 on H. We now show how to construct a vertex cover V' of G such that $|V'| \leq |F|$. For $e \in F$, if e is purple, i.e. $e = \{v_0, x\}$, then add x to V'. Otherwise, if $e = \{x, y\} \in E$ then add either x or y to V' (and if it is red ignore it). Clearly $|V'| \leq |F|$ and moreover it is a valid vertex cover, since for the unsatisfied triangle inequality corresponding to a given edge in $e \in E$, the solution F must contain either e or one of its adjacent purple edges (and in either case we add an endpoint of e to V').

Conversely, observe that for each unsatisfied inequality corresponding to an edge $\{x, y\} \in E$, if we change the weight of either the purple edge $w(v_0, x)$ or $w(v_0, y)$ (or both) to be $1 + \epsilon$

then the inequality will be satisfied. Moreover, observe that if for a given subset of purple edges, if we increase all their weights to be $1+\epsilon$, and not modify any other edge weights, then we will not create any new unsatisfied inequalities. So let V' be any solution to the vertex cover problem on G. The above implies that if for every $x \in V'$ we set $w(v_0, x) = 1+\epsilon$, and leave all other weights unchanged, then we have a valid solution to Problem 3.3.1 on H, such that $|F| = |V'|$ where F is the set of modified edges.

The above reduction shows that Problem 3.3.1 is NP-Complete. In terms of approximating the MVD optimization problem, observe that the above reduction actually implies that a minimum sized solution to the described instance of MVD corresponds to a minimum vertex cover from G of the same size (which we can even be read off). Thus any approximation to MVD yields the same approximation to the minimum vertex cover problem. □

It is not difficult to argue that the hardness proof above for MVD also applies to MVID, since in the proof weights only ever needed to be increased (we omit the details as they are nearly identical to the above). Thus we have the following corollary of Theorem 3.3.3.

Corollary 3.3.4. *The decision version of MVID is NP-Complete. Moreover, MVID is APX-hard, and assuming the Unique Games Conjecture, is hard to approximate within a factor of* $2-\varepsilon$ *for any* $\varepsilon > 0$.

3.3.1 Metric Violation Decrease Distance is Polynomial Time Solvable.

In this section we show that MVDD is actually polynomial time solvable by making a connection to the all pairs shortest paths problem.

Lemma 3.3.5. *For an instance* $G = (V, E)$ *of MVDD (Problem 3.2.3), a minimum size subset* $S \subseteq E$ *such that only decreasing edges from* S *turns* G *into a metric graph can be found in polynomial time. More precisely, the running time is the same as computing the set of all pairs shortest path distances.*

Proof. The algorithm is simple. Compute the all pairs shortest paths distances, and for every edge $e = \{u, v\} \in E$, set $w(e)$ equal to the shortest path distance between u and v. Thus S is the set of edges whose weight is larger than the shortest path distance between its end points, as these were the only weights which were changed.

First observe that it is necessary to change the weights of the edges in S. Specifically, let $e = \{u, v\}$, and suppose $w(e)$ is larger than the shortest path distance between u and v. In this case the shortest u, v path together with the edge e forms an unbalanced cycle, which by Lemma 3.2.5 we must balance. In order to balance this cycle, $w(e)$ needs to be less than or equal to the sum of the weights of the other edges in the cycle, and since edge weights can only decrease, this implies balancing this cycle requires decreasing $w(e)$.

Now we argue that setting the edge weights in S to their shortest path distance suffices to solve this instance of Problem 3.2.3. First observe that for any edge $e = \{u, v\}$ from any positively weighted graph if we set $w(e)$ equal to its shortest path distance, then the shortest path distance between every pair of vertices in the graph remains the same. (This is because the only path lengths which can change are those which used the edge e, and for any such path if we replace e with the shortest u, v path from the original graph, we will get a walk in the original graph of the same total length.) Now let H be the graph obtained from G after setting the weights in S to their shortest path distance from G. The above observation implies that shortest path distances in H are the same as those in G, as one can imagine obtaining H from G by modifying one edge from S at a time. As H was obtained from G by setting weights of edges to their shortest path distance in G, this implies that in H the weight of every edge is equal to the shortest path distance between its endpoints, implying there can be no unsatisfied triangle inequalities and so we are done. □

3.4 Approximation Algorithm for MVID

In this section we first provide the details for our approximation to the MVID problem. In the next section, our approximation for the MVD problem will then follow by a more intricate version of the same argument.

3.4.1 Unbalanced Cycles.

Before providing our approximation algorithm for MVID, in this subsection we first show that solutions to MVID can be characterized in terms of unbalanced cycles. Recall Lemma 3.2.5, which implies that any solution to a MVID instance must non-top cover all unbalanced cycles. We argue the more surprising fact that any such non-top cover is also sufficient.

Lemma 3.4.1. *If G is an instance of MVID and S is a non-top cover of all unbalanced cycles, then G can be converted into a metric graph by only increasing weights of edges in S.*

Proof. For now assume all edge weights are integers and let L denote the largest edge weight. We describe a procedure which only modifies edges from S, producing a new instance G', such that (i) S remains a non-top cover of G', (ii) the weight of at least one edge strictly increases and none decrease, and (iii) no edge weight is ever increased above L. For any instance G and non-top cover S, if we prove such a procedure exists whenever not all the edges in S have weight L, then this will imply the lemma. Specifically, after applying the procedure at most $|S| \cdot L$ times, all edge weights will be equal to L, and hence no unbalanced cycle (and so no unsatisfied triangle inequality) can remain, since otherwise S would cover that unbalanced cycle with an edge of weight L, which is at least the weight of that cycle's top edge (i.e. the cycle is not actually unbalanced). Moreover, this procedure only increases weights of edges in S, as desired. Note also that as the number of steps in this existential argument was irrelevant, so long as it was finite, this procedure will imply the claim for rational input weights as well.

We now prove the above described procedure exists. Let G and S be as in the lemma statement, and fix any unbalanced triangle, which must exist otherwise all triangle inequalities are already satisfied. Let a, b, and t be the edges of this triangle, where t is the top edge, i.e. $w(t) > w(a) + w(b)$. Note at least one of either a or b must be in the set S. We break the analysis into two cases based on whether just one or both are in S. In the following, for any cycle $\mathbb{C}$, let $w(\mathbb{C})$ denote the sum of the weights of the edges in $\mathbb{C}$.

Case 1: only one of a or b is in S. Without loss of generality suppose a is in S. We then increase $w(a)$ to be $w(t) - w(b)$. If no new unbalanced cycles are created after the increase, then we are done. Otherwise, if some new unbalanced cycle $\mathbb{C}$ was created after increasing $w(a)$, then a must be the top edge of $\mathbb{C}$. Let $\pi(\mathbb{C} \setminus a)$ denote the sub-path of cycle $\mathbb{C}$ starting at the common vertex of a and t, and ending at the common vertex of a and b. Consider the closed walk $\sigma = \pi(\mathbb{C} \setminus a) \circ b \circ t$, where '$\circ$' denotes walk concatenation. Note σ may not be a (simple) cycle, however σ must contain a cycle C which includes the edge t. Note that C is unbalanced with top edge t, since it only contains edges from $(\mathbb{C} \setminus a) \cup \{b, t\}$, and so $w(C \setminus t) = w(C) - w(t) \leq w(\mathbb{C} \setminus a) + w(b) = w(\mathbb{C} \setminus a) - w(a) + w(t) < w(t)$, where the last equality follows since we set $w(a) = w(t) - w(b)$. Since C is unbalanced, it must be non-top covered by S, which implies $\mathbb{C}$ is non-top cover by S. This is because C only contains edges from $(\mathbb{C} \setminus a) \cup \{b, t\}$, has top edge t, and by assumption $b \notin S$.

Case 2: both a and b are in S. First, increase $w(a)$ to the largest value possible that does not create any new cycles that are both unbalanced and not non-top covered. (Note the largest such value is well defined as equality satisfies the triangle inequality.) If afterwards $w(a) \geq w(t) - w(b)$, then we are done as the weight of a has strictly increased since the previously violated triangle is now satisfied. Otherwise, $w(a) < w(t) - w(b)$ and there is some just balanced cycle $\mathbb{C}_1$ such that $(\mathbb{C}_1 \setminus a) \cap S = \emptyset$, which would become unbalanced if $w(a)$ were any larger. Now increase $w(b)$ to $w(b) = w(t) - w(a)$. If this does not create any non-top uncovered unbalanced cycle then we are done. Otherwise, there is some non-top

uncovered unbalanced cycle $\mathbb{C}_2$ created by setting $w(b) = w(t) - w(a)$. Observe for later that after the change to $w(b)$, it still holds that $w(\mathbb{C}_1 \setminus a) = w(a)$ as $(\mathbb{C}_1 \setminus a) \cap S = \emptyset$ and $b \in S$. Let $\pi(\mathbb{C}_1 \setminus a)$ denote the sub-path of cycle $\mathbb{C}_1$ starting at the common vertex of a and t, and ending at the common vertex of a and b, and let $\pi(\mathbb{C}_2 \setminus b)$ denote the sub-path of cycle $\mathbb{C}_2$ starting at the common vertex of a and b, and ending at the common vertex of b and t. Then consider the closed walk $\sigma = \pi(\mathbb{C}_1 \setminus a) \circ \pi(\mathbb{C}_2 \setminus b) \circ t$. As σ is closed it must contain some cycle C which includes t. Note that C is unbalanced with top edge t, since it only contains edges from $(\mathbb{C}_1 \setminus a) \cup (\mathbb{C}_2 \setminus b) \cup \{t\}$, and so $w(C \setminus t) = w(C) - w(t) \leq w(\mathbb{C}_1 \setminus a) + w(\mathbb{C}_2 \setminus b) = w(a) + w(\mathbb{C}_2 \setminus b) < w(a) + w(b) = w(t)$. Since C is unbalanced, it must be non-top covered by S, which implies $\mathbb{C}_2$ is non-top covered by S. This is because C only contains edges from $(\mathbb{C}_1 \setminus a) \cup (\mathbb{C}_2 \setminus b) \cup \{t\}$, $(\mathbb{C}_1 \setminus a) \cap S = \emptyset$, and t is the top edge of C.

In either case, the weight of at least one edge from S was strictly increased, no edges were decreased, and any newly created unbalanced cycle is still covered by S, and thus the conditions of the above described procedure are satisfied. □

Lemma 3.2.5 and Lemma 3.4.1 immediately imply the following.

Corollary 3.4.2. *Let G be an instance of MVID. If $S \subset E$ is a minimum size set which non-top covers all the unbalanced cycles in G, then S is an optimal solution to MVID.*

3.4.2 Small Cycle Covers are Almost Enough.

Corollary 3.4.2 showed the equivalence between minimum non-top covers of all unbalanced cycles and solutions to MVID. The question is how hard is it to find or approximate such a cover. We have already argued MVID is at least as hard as vertex cover, however, covering all unbalanced cycles appears a step closer to the more general set cover problem, which is logarithmically hard to approximate. Worse still, we cannot (at least explicitly) write down our non-top cover problem as a set cover instance as there are potentially an exponential number of unbalanced cycles.

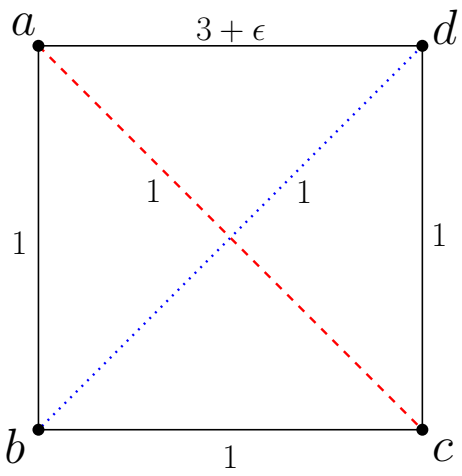


Figure 3.3: Unbalanced 4-cycle not covered by the dashed 3-cycle cover.

One natural question is whether explicitly covering larger cycles is necessary, i.e. perhaps non-top covering all unbalanced 3 cycles implies non-top covering all larger unbalanced cycles, however the example in Figure 3.3 quickly dispels this idea. This example consists of single unbalanced 4-cycle, whose vertices in cyclic order are (a, b, c, d), whose top edge $\{a, d\}$ has weight $3 + \varepsilon$, and all other edges have weight 1. Here there are are two unbalanced 3-cycles, (a, b, d) and (a, c, d). Thus all unbalanced 3-cycles can be covered by the edges, $\{a, c\}$ and $\{b, d\}$, however this does not cover the unbalanced 4-cycle. Unfortunately, this can be generalized for any integer n, by creating a cycle of length n where all edges in the graph have weight 1 except for a single cycle edge which has weight $n - 1 + \varepsilon$. Thus for no $k < n$ does a non-top cover of all unbalanced cycles of size $\leq k$ imply a non-top cover of all unbalanced cycles.

So we must consider larger unbalanced cycles, which motivates the following definitions.

Definition 3.4.3. *Let C be an unbalanced cycle of length $k \geq 4$, and let $(v_1, v_2, \ldots, v_k)$ be the cyclic ordering of the vertices in C, where $t = \{v_k, v_1\}$ is the top edge of C. For any chord $e = \{v_i, v_j\}$ of C (i.e. $i < j$ and differ by more than 1 mod k), we define two cycles. The* top cycle*, $top(C, e) = (v_1, \ldots, v_i, v_j, \ldots, v_k)$ containing the top edge t, and the* bottom cycle*, $bot(C, e) = (v_i, v_{i+1} \ldots, v_j)$.*

For an unbalanced cycle C, if there exists a chord e of C such that $bot(C, e)$ is unbalanced and *e is the top edge of $bot(C, e)$, then C is called a* non-unit *cycle, otherwise C is called a* unit *cycle.*

The following simple observation implies we can limit attention to unit cycles.

Observation 3.4.4. *Let $G = (V, E)$ be an instance of MVID. If $S \subset E$ non-top covers all unit cycles in G, then S must non-top cover all unbalanced cycles, and hence is a solution to MVID. This is because by definition any non-unit cycle C must have a chord e whose bottom cycle is unbalanced with top edge e, and in particular the smallest such bottom cycle C' must be a unit cycle. Thus as the non-top edges of C' are a subset of those of C, if S non-top covers C' it must non-top cover C.*

In general there can still be large unit cycles, however, we have the following useful property.

Lemma 3.4.5. *For any chord e of a unit cycle C, $top(C, e)$ is unbalanced, with the same top edge as C.*

Proof. First, observe that since C is a unit cycle, either the bottom cycle of e is balanced, or it is unbalanced but the top edge is not e. In either case, $w(e) \leq w(bot(C, e)) - w(e)$. Let t be the top edge of C. We thus have $w(t) > w(C) - w(t) = w(top(C, e)) + w(bot(C, e)) - 2w(e) - w(t) \geq w(top(C, e)) - w(t)$. □

To see why the above lemma is useful for handling larger unit cycles, we must first get rid of smaller unbalanced cycles, which can be easily done.

Lemma 3.4.6. *Let $G = (V, E)$ be an instance of MVID, let $C_{\leq k}$ be the set of all unbalanced cycles with at most k edges for some constant k, and let opt be a minimum size non-top cover of $C_{\leq k}$. Then in polynomial time one can compute a constant factor approximation to opt.*

More precisely, in $O(|V|^k)$ time, one can compute a set $S_{\leq k}$ which non-top covers $C_{\leq k}$, and such that $|S_{\leq k}| \leq (k-1)|opt|$.

Proof. The proof follows by applying the standard hitting set approximation for bounded size sets. Specifically, each unbalanced cycle in $C_{\leq k}$ defines a set of $\leq k-1$ non-top edges. Let H denote the collection of all such sets, and observe a non-top cover of $C_{\leq k}$ corresponds to a hitting set for H. So initially let $S_{\leq k}$ be the empty set, and consider the sets in H one at a time. If when considering a set $h \in H$, if h has not been hit, then add all edges of h to $S_{\leq k}$, and otherwise do nothing. Clearly the final set $S_{\leq k}$ is a hitting set, and has size at most $(k-1)|opt|$ since each time we add all edges of a set, the set was uncovered, and so any solution had to add at least one of those edges. The running time follows as we make a single linear pass over H, and note that $C_{\leq k}$ and hence H can be enumerated in $O(|V|^k)$ time. □

Now for larger unit cycles we have the following useful corollary of Lemma 3.4.5.

Corollary 3.4.7. *Let $S_{\leq k}$ be a non-top cover of all unbalanced cycles of length $\leq k$. Then for any unit cycle C with strictly more than k edges, if C is* not *non-top covered by $S_{\leq k}$, then for any chord e such that $top(C, e)$ has $\leq k$ edges, $e \in S_{\leq k}$.*

Proof. Let e be any chord such that $top(C, e)$ has $\leq k$ edges. As C is a unit cycle, by Lemma 3.4.5, $top(C, e)$ is unbalanced, with the same top edge as C. Thus $S_{\leq k}$ must non-top cover $top(C, e)$. Note however, that with the exception of the edge e, the non-top edges of $top(C, e)$ are a subset of those of C. Thus since C is not non-top covered by $S_{\leq k}$, e must be the edge in $S_{\leq k}$ which non-top covers $top(C, e)$. □

By Observation 3.4.4 we know it suffices to non-top cover all unit cycles, however, the issue is that in general unit cycles may be large. On the other hand, Corollary 3.4.7 tells us that if we first cover smaller unbalanced cycles, then while we may not cover larger unit cycles, we do cover all chords near their top edges. We first describe a procedure which adds a carefully chosen set of edges to cover larger unit cycles, and then in the next section argue how to bound the size of the added set by charging to these chords.

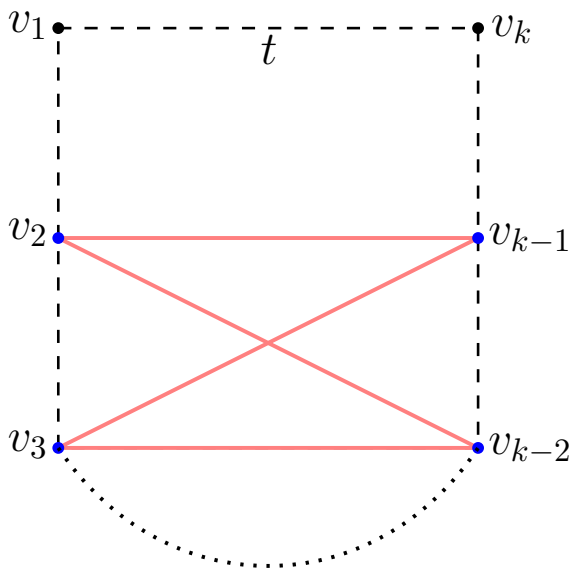


Figure 3.4: Cycle $C = (v_1, v_2 \ldots, v_k)$, and edges of $embed4(C)$ in solid red.

Definition 3.4.8. *Let $S_{\leq 6}$ be a non-top cover of all unbalanced cycles with ≤ 6 edges. Consider any unbalanced cycle $C = (v_1, v_2, v_3 \ldots, v_{k-2}, v_{k-1}, v_k)$, with $k > 6$, where $t = \{v_k, v_1\}$ is the top edge of C. For any such cycle C, we define the* embedded 4-cycle *of C to be the cycle defined by the cyclically ordered edges $embed4(C) = (\{v_2, v_{k-2}\}, \{v_{k-2}, v_3\}, \{v_3, v_{k-1}\}, \{v_{k-1}, v_2\})$. See Figure 3.4.*

Observe that for a unit cycle C with more than 6 edges, Corollary 3.4.7 implies that if C is not covered by $S_{\leq 6}$, then $S_{\leq 6}$ must contain all edges of $embed4(C)$. Next observe that the chords of $embed4(C)$, namely the edges $\{v_2, v_3\}$ and $\{v_{k-2}, v_{k-1}\}$ are actually non-top edges of the cycle C. In other words, if we add to $S_{\leq 6}$ any set of edges containing at least one chord from the embedded 4 cycle of each unit cycle, then we will have a non-top cover of all unit cycles, and hence a solution to MVID by Observation 3.4.4. Recall we cannot list all unit cycles, and hence it is not clear how to precisely list all their embedded 4 cycles. On the other hand, as all edges of an embedded 4 cycle of a unit cycle are contained in $S_{\leq 6}$, we can instead consider the potentially larger set of all 4 cycles in the graph defined by the edge set $S_{\leq 6}$. This implies the following lemma.

Lemma 3.4.9. *Let G be an instance of MVID, and let $S_{\leq 6}$ be a non-top cover of all unbalanced cycles with ≤ 6 edges. Let S_c be a set of edges containing at least one chord from all possible 4 cycles defined by the edges of $S_{\leq 6}$. Then $S = S_{\leq 6} \cup S_c$ is a valid solution to the given instance of MVID, that is the edges in S can be increased to convert G into a metric graph.*

3.4.3 Chording Cycles.

The goal of this section is to compute a set S_c as described in Lemma 3.4.9 which contains at least one chord of every induced 4-cycle of a subset of edges. So let $G = (V, E)$ be a complete graph, and for any edge subset $F \subseteq E$, consider the subgraph $H = (V, F)$. Let $Cycle(F)$ denote the set of all cycles in H with with exactly 4 edges. Let $chord4(F)$ denote the following iterative procedure. Initially let $S = \emptyset$, and consider each cycle $C \in Cycle(F)$ one at time. If either chord of C appears in S ignore C, and otherwise add both chords to S. (Note chords are considered from the complete graph G, i.e. regardless of whether they appear in H.) After all cycles have been considered output S.

Clearly $chord4(F)$ outputs a set S containing at least one chord from each cycle in $Cycle(F)$ and does so in polynomial time. The question is how big is $|S|$ relative to $|F|$. So let $\mathbb{C}$ denote the set of cycles for which $chord4(F)$ added its chords to S. Observe that no two cycles in $\mathbb{C}$ can share a chord (as otherwise $chord4(F)$ would not have added the latter of these two cycles to $\mathbb{C}$). So we have $|S|/2 = |\mathbb{C}|$, and also note that $|edges(\mathbb{C})| \leq |F|$, where $edges(\mathbb{C})$ is the set of all edges from cycles in $\mathbb{C}$. Thus to bound the size of $|S|$ in terms of $|F|$, it suffices to bound $|\mathbb{C}|$ in terms of $|edges(\mathbb{C})|$. This gives the following purely combinatorial problem.

Lemma 3.4.10. *Let $G = (V, E)$ be a graph whose edge set E is the union of the edges of a collection of 4-cycles, $\mathbb{C}$, such that no two 4-cycles in $\mathbb{C}$ share a chord. (Note this applies to*

all chords in the complete graph on V, i.e. regardless of whether they appear in E.) Then $|\mathbb{C}| = O(|E|^{4/3})$.

Proof. We partition the vertex set V into two groups based on degree. Specifically, let V_s be the set of all vertices of degree $\leq |E|^{1/3}$ and let V_l be the vertices with degree $> |E|^{1/3}$. Now we separately bound the number of cycles containing at least one vertex from V_s and the number of cycles only containing vertices from V_l, and hence the total number of cycles is the sum of these two numbers. First, we further partition V_s into groups $g_1, \ldots, g_k$ such that for all $1 \leq i \leq k$, $|E|^{1/3} \leq \sum_{v \in g_i} degree(v) \leq 2|E|^{1/3}$. (Such a partition can be computed by iterating over the vertices in V_s with a running degree sum total, setting aside a group and reseting the total to zero each time the sum is $\geq |E|^{1/3}$.) Note that as the total degree of each g_i is at least $|E|^{1/3}$, by the degree sum formula $k|E|^{1/3} \leq 2|E|$, and so $k \leq 2|E|^{2/3}$. Now any two edges adjacent to the same vertex can both appear together in at most one 4-cycle, as otherwise this would imply the two 4-cycles share a chord. Thus each distinct pair of edges adjacent to a vertex v can correspond to at most one cycle, and hence the total number of cycles involving v is bounded by $\binom{degree(v)}{2} \leq (degree(v))^2/2$. We thus have that number of cycles involving vertices from V_s is

$$\leq \frac{1}{2} \sum_{i=1}^{k} \sum_{v \in g_i} (degree(v))^2 \leq \frac{1}{2} \sum_{i=1}^{k} \left(\sum_{v \in g_i} degree(v) \right)^2$$
$$\leq \frac{1}{2} \sum_{i=1}^{k} 4|E|^{2/3} \leq |E|^{2/3} \cdot 4|E|^{2/3} = 4|E|^{4/3},$$

where the second inequality follows from the fact that $a^2 + b^2 \leq (a+b)^2$ for $a, b \geq 0$.

Now consider the vertices in V_l. As each vertex in V_l has degree $> |E|^{1/3}$, by the degree sum formula, $2|E| \geq \sum_{v \in V_l} degree(v) \geq |V_l| \cdot |E|^{1/3}$. Therefore $|V_l| \leq 2|E|^{2/3}$. Now as discussed above, we only need to consider cycles composed entirely of vertices from V_l. Now V_l can define at most $\binom{|V_l|}{2} \leq |V_l|^2/2 \leq 2|E|^{4/3}$ chords, and since no two cycles can share a chord, this implies V_l can define at most $|E|^{4/3}$ cycles, and thus the lemma statement follows. □

By the discussion before the lemma, we thus have the following.

Corollary 3.4.11. *Let F be any subset of edges from a complete graph $G = (V, E)$. Then in polynomial time chord4(F) outputs a set of edges S such that (i) S contains at least one chord of every 4-cycle induced by the edges of F, and (ii) $|S| = O(|F|^{4/3})$.*

The reader may wonder whether the analysis of Lemma 3.4.10 is tight. We provide a corresponding lower bound showing that indeed the 4/3 exponent is tight in the worst case for this combinatorial problem. This lower bound uses an interesting and non-trivial probabilistic method argument, we put it into Section 3.6.

3.4.4 The Result.

The approach outlined in Section 3.4.2 (particularly Lemma 3.4.9), together with the bound from Corollary 3.4.11 readily give the following.

Input : An instance G of MVID
Output : Valid solution S to the given instance.
1 Compute a non-top cover $S_{\leq 6}$ of all unbalanced cycles with ≤ 6 edges using Lemma 3.4.6
2 Compute a chord cover $S_c = chord4(S_{\leq 6})$ using Corollary 3.4.11
3 Formulate and return any feasible solution to the LP described in Section 3.2.2 for the edge set $S = S_{\leq 6} \cup S_c$.

Algorithm 1: Finds a valid solution for MVID.

Theorem 3.4.12. *Algorithm 1 gives a polynomial time $O(OPT^{1/3})$-approximation to any instance G of MVID (Problem 3.2.4), where OPT denotes the size of an optimal solution.*

Proof. Algorithm 1 returns a set $S = S_{\leq 6} \cup S_c$. By Lemma 3.4.6, $S_{\leq 6}$ is a non-top cover of all unbalanced cycles with ≤ 6 edges, and by Corollary 3.4.11, S_c contains at least one chord from every 4-cycle induced by the edges of $S_{\leq 6}$. Thus by Lemma 3.4.9, S is a valid solution to the given instance of MVID.

The running time is clearly polynomial as the procedures of Lemma 3.4.6, Corollary 3.4.11, and the feasibility checking of Section 3.2.2, were all already argued to run in polynomial time. As for the approximation quality, note that Lemma 3.4.1 implies OPT is the size of a minimum sized set of edges which non-top covers all unbalanced cycles. Moreover, Lemma 3.4.6 tells us $|S_{\leq 6}| \leq 5|opt_6| = O(OPT)$, where opt_6 denotes any minimum size non-top cover of all unbalanced cycles of length ≤ 6. Finally, Corollary 3.4.11 tells us $|S_c| = O(|S_{\leq 6}|^{4/3})$, and therefore $|S| = O(OPT^{4/3})$. □

3.5 Approximation Algorithm for MVD

Now we consider the MVD problem, where both increasing and decreasing edge weights are allowed. The high level argument will be similar to that used for MVID.

3.5.1 Unbalanced Cycles.

Recall our different notions of covering from Section 3.2.3. Lemma 3.2.5 implies any solution to a MVD instance must regular cover all unbalanced cycles. We now prove the more surprising fact that any such regular cover is also sufficient, that is the analog of Lemma 3.4.1 but for MVD. However, the proof here is far more intricate, and in particular requires the following helper lemma.

Lemma 3.5.1. *Let $G = (V, E)$ be an instance of MVD. If S is a regular cover of all unbalanced cycles, then S can be partitioned into two disjoint sets S^+ and S^- such that each unbalanced cycle is either non-top covered by S^+ or top covered by S^-.*

Proof. Let S be a regular cover of all unbalanced cycles and let S^+ and S^- initially be empty sets. We now define an interative procedure, which in each iteration removes one edge from S and adds it to either S^+ or S^-. We maintain the invariant that each unbalanced cycle is either top covered by $S^- \cup S$ or non-top covered by $S^+ \cup S$. Thus, after a finite number of

iterations, S will be empty, and S^+ and S^- will be two disjoint sets such that each unbalanced cycle is either non-top covered by S^+ or top covered by S^-.

Suppose at step i, we pick an edge b from set S. There are two cases. Case 1: if each unbalanced cycle is either top covered by $S^- \cup (S \setminus b)$ or non-top covered by $(S^+ \cup b) \cup (S \setminus b)$, then we add b to S^+. Case 2: if each unbalanced cycle is either top covered by $(S^- \cup b) \cup (S \setminus b)$ or non-top covered by $S^+ \cup (S \setminus b)$, then we add b to S^-. We now argue by contradiction that these are the only possible cases.

If Case 1 does not hold, then there must be an unbalanced cycle $\mathbb{C}_1$ which is neither top covered by $S^- \cup (S \setminus b)$ nor non-top covered by $(S^+ \cup b) \cup (S \setminus b)$. As $\mathbb{C}_1$ must be top covered by $S^- \cup S$ or non-top covered by $S^+ \cup S$ (by induction), this implies that b must top cover $\mathbb{C}_1$. If Case 2 does not hold then there is an unbalanced cycle $\mathbb{C}_2$ which is neither top covered $(S^- \cup b) \cup (S \setminus b)$ nor non-top covered by $S^+ \cup (S \setminus b)$, and similarly this implies b must non-top cover $\mathbb{C}_2$. Now consider the closed walk $\sigma = \pi(\mathbb{C}_1 \setminus b) \circ \pi(\mathbb{C}_2 \setminus b)$ (see Figure 3.5). Let t be the top edge of $\mathbb{C}_2$. There exists a cycle C only containing edges from σ whose top edge is t, and is unbalanced because $w(t) > w(\mathbb{C}_2 \setminus t) = w(\mathbb{C}_2 \setminus \{t, b\}) + w(b) > w(\mathbb{C}_2 \setminus \{t, b\}) + w(\mathbb{C}_1 \setminus b) \geq w(C) - w(t)$. Observe that $(\mathbb{C}_1 \setminus b) \cap (S^+ \cup S) = \emptyset$ and $(\mathbb{C}_2 \setminus t) \cap (S^+ \cup (S \setminus b)) = \emptyset$ which implies $(C \setminus t) \cap (S^+ \cup S) = \emptyset$ because $b \notin C$. Additionally, we have that $t \notin (S^- \cup S)$ as otherwise $\mathbb{C}_2$ would be top covered by $(S^- \cup b) \cup (S \setminus b)$. As C must be top covered by $S^- \cup S$ or non-top covered by $S^+ \cup S$ (again by induction), this gives a contradiction as we showed $t \notin (S^- \cup S)$ and $(C \setminus t) \cap (S^+ \cup S) = \emptyset$.

Therefore, we can add b to S^+ or S^- (according to case 1 or 2) and remove it from S such that each unbalanced cycle remains either top covered by $S^- \cup S$ or non-top covered by $S^+ \cup S$. Thus, after at most $|S_0|$ rounds, where S_0 is the initial state of S, S^+ and S^- will be as in the lemma statement. □

Lemma 3.5.2. *If G is an instance of MVD and S is a regular cover of all unbalanced cycles, then G can be converted into a metric graph by only changing weights of edges in S.*

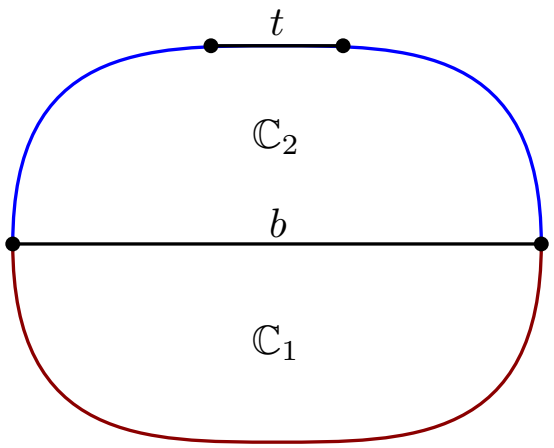


Figure 3.5: Edge b top covers $\mathbb{C}_1$ and non-top covers $\mathbb{C}_2$.

Proof. For now assume all edge weights are integers and let L denote the largest edge weight. First use Lemma 3.5.1 to partition S into two disjoint sets S^+ and S^-, such that every unbalanced cycle is either non-top covered by S^+ or top covered by S^-. We describe a procedure, producing a new instance G', such that (i) every unbalanced cycle of G' is either non-top covered by S^+ or top covered by S^-, (ii) the weight of either at least one S^+ edge strictly increases or at least one S^- edge strictly decreases (and no other edge weights are modified), and (iii) no edge weight is ever increased above L or below 0. Proving such a procedure exists for any instance G and regular cover S, will imply the lemma. Specifically, after applying the procedure at most $|S| \cdot L$ times, all edge weights in S^+ will be equal to L, and all edge weights in S^- will be equal to 0. This implies there are no remaining unbalanced cycles, since otherwise the unbalanced cycle is either non-top covered by an edge of weight L or top covered by an edge of weight 0, in either case implying the cycle is not actually unbalanced. Moreover, this procedure only modifies edge weights from S as desired. As the number of steps in this existential argument is irrelevant, so long as it is finite, this procedure will imply the claim for rational input weights as well.

We now prove the above described procedure exists. Let G and S be as in the lemma statement, and fix any unbalanced triangle, which must exist otherwise all triangle inequalities

are already satisfied. Let a, b, and t be the edges of this triangle, where t is the top edge. Note that either $\{a, b\} \cap S^+ \neq \emptyset$ or $t \in S^-$. We break the analysis into several cases based on whether just one or both of a,b are in S^+, or neither are in S^+ and t is in S^-. If a cycle is not non-top covered by S^+ or not top covered by S^-, we say that cycle is uncovered.

Case 1: $t \notin S^-$, and exactly one of a or b is in S^+. Without loss of generality suppose $a \in S^+$, $b \notin S^+, t \notin S^-$. We then increase $w(a)$ to be $w(t) - w(b)$. If no new unbalanced cycles are created after the increase, then we are done. Otherwise, if some new unbalanced cycle $\mathbb{C}$ was created after increasing $w(a)$, then a must be the top edge of $\mathbb{C}$. Let $\pi(\mathbb{C} \setminus a)$ denote the sub-path of cycle $\mathbb{C}$ which starts at the common vertex of a and t, and ends at the common vertex of a and b. Consider the closed walk $\sigma = \pi(\mathbb{C} \setminus a) \circ b \circ t$. Note σ may not be a (simple) cycle, but must contain a cycle C which includes the edge t. Note that C is unbalanced with top edge t, since it only contains edges from $(\mathbb{C} \setminus a) \cup \{b, t\}$, and so $w(C \setminus t) = w(C) - w(t) \leq w(\mathbb{C} \setminus a) + w(b) = w(\mathbb{C} \setminus a) - w(a) + w(t) < w(t)$, where the last equality follows since we set $w(a) = w(t) - w(b)$. Since C is unbalanced, it must be non-top covered by S^+ as $t \notin S^-$, which implies $\mathbb{C}$ is non-top covered by S^+, because C only contains edges from $(\mathbb{C} \setminus a) \cup \{b, t\}$, has top edge t, and by assumption $b \notin S^+$.

Case 2: $t \notin S^-$ and $a, b \in S^+$. First, increase $w(a)$ to the largest value possible that does not create any uncovered unbalanced cycle. If afterwards $w(a) \geq w(t) - w(b)$, then we are done as the weight of a has strictly increased since the previously violated triangle is now satisfied. Otherwise, $w(a) < w(t) - w(b)$ and there is some just balanced cycle $\mathbb{C}_1$ that $(\mathbb{C}_1 \setminus a) \cap S^+ = \emptyset$, which would become unbalanced if $w(a)$ were any larger. Now increase $w(b)$ to $w(t) - w(a)$. If this does not create any uncovered unbalanced cycle then we are done; otherwise, some unbalanced cycle $\mathbb{C}_2$ was created. Observe for later that after the change to $w(b)$, it still holds that $w(\mathbb{C}_1 \setminus a) = w(a)$ as $(\mathbb{C}_1 \setminus a) \cap S^+ = \emptyset$ and $b \in S^+$. Let $\pi(\mathbb{C}_1 \setminus a)$ denote the sub-path of $\mathbb{C}_1$ starting at the common vertex of a and t, and ending at the common vertex of a and b, and let $\pi(\mathbb{C}_2 \setminus b)$ denote the sub-path of $\mathbb{C}_2$ starting at the common vertex of a and b, and ending

at the common vertex of b and t. Then consider the closed walk $\sigma = \pi(\mathbb{C}_1 \setminus a) \circ \pi(\mathbb{C}_2 \setminus b) \circ t$. As σ is closed it must contain some cycle C which includes the edge t. Note that C is unbalanced with top edge t, since it only contains edges from $(\mathbb{C}_1 \setminus a) \cup (\mathbb{C}_2 \setminus b) \cup \{t\}$, and so $w(C \setminus t) = w(C) - w(t) \leq w(\mathbb{C}_1 \setminus a) + w(\mathbb{C}_2 \setminus b) = w(a) + w(\mathbb{C}_2 \setminus b) < w(a) + w(b) = w(t)$. Since C is unbalanced, it must be non-top covered by S^+ as $t \notin S^-$, which implies $\mathbb{C}_2$ is non-top covered by S^+. This is because C only contains edges from $(\mathbb{C}_1 \setminus a) \cup (\mathbb{C}_2 \setminus b) \cup \{t\}$, and $(\mathbb{C}_1 \setminus a) \cap S^+ = \emptyset$, and t is the top edge of C.

Case 3: $t \in S^-$ and $a, b \notin S^+$. First, decrease $w(t)$ to be $w(a)+w(b)$. If no new unbalanced cycles are created, then we are done. Otherwise, some new unbalanced cycle $\mathbb{C}$ was created, which includes t and has top edge $t' \neq t$. Let $\pi(\mathbb{C} \setminus t)$ denote the sub-path $\mathbb{C}$ starting at the common vertex of t and a, and ending at the common vertex of t and b. Consider the closed walk $\sigma = \pi(\mathbb{C} \setminus t) \circ a \circ b$, and let C be a cycle contained in the σ which includes t'. Note that C is unbalanced with top edge t', since it only contains edges from $(\mathbb{C} \setminus t) \cup \{a, b\}$, and so it holds that $w(C \setminus t') = w(C) - w(t') \leq w(\mathbb{C}) + w(a) + w(b) - w(t) - w(t') = w(\mathbb{C} \setminus t') < w(t')$, where the last equality follows since we set $w(t) = w(a) + w(b)$. Since C is unbalanced, it must be either top covered by S^- or non-top covered by S^+, which implies $\mathbb{C}$ is either top covered by S^- or non-top covered by S^+, because C only contains edges from $(\mathbb{C} \setminus t) \cup \{a, b\}$, and by assumption $a, b \notin S^+$.

Case 4: $t \in S^-$ and exactly one of a or b is in S^+. Without loss of generality suppose $a \in S^+$, $b \notin S^+$. First, increase $w(a)$ to the largest value possible that does not create any new uncovered unbalanced cycle. If afterwards $w(a) \geq w(t) - w(b)$, then we are done. Otherwise, $w(a) < w(t) - w(b)$ and there is some just balanced cycle $\mathbb{C}_1$ with $(\mathbb{C}_1 \setminus a) \cap S^+ = \emptyset$, which would be unbalanced if $w(a)$ were any larger. Now decrease $w(t)$ to $w(t) = w(b) + w(a)$. If this does not create an unbalance cycle, then we are done. Otherwise, some unbalanced cycle $\mathbb{C}_2$ containing t was created, with top edge $t' \neq t$. Let $\pi(\mathbb{C}_1 \setminus a)$ denote the sub-path of $\mathbb{C}_1$ starting at the common vertex of a and t and ending at the common vertex of a and b, and let

$\pi(\mathbb{C}_2 \setminus b)$ denote the sub-path of $\mathbb{C}_2$ starting at the common vertex of b and t and ending at the common vertex of a and t. Consider the closed walk $\sigma = \pi(\mathbb{C}_1 \setminus a) \circ b \circ \pi(\mathbb{C}_2 \setminus t)$, and let C be a cycle in σ including t'. Note C is unbalanced with top edge t', since it only contains edges from $(\mathbb{C}_1 \setminus a) \cup (\mathbb{C}_2 \setminus t) \cup \{b\}$, and so $w(C \setminus t') \leq w(C) - w(t') \leq w(\mathbb{C}_1 \setminus a) + w(\mathbb{C}_2 \setminus t) + w(b) - w(t') = w(a) + w(b) + w(\mathbb{C}_2) - w(t) - w(t') = w(\mathbb{C}_2) - w(t') = w(\mathbb{C}_2 \setminus t') < w(t')$. Since C is unbalanced, it is either non-top covered by S^+ or top covered by S^-, implying $\mathbb{C}_2$ is either non-top covered by S^+ or top covered by $t' \in S^-$, since C only has edges from $(\mathbb{C}_1 \setminus a) \cup (\mathbb{C}_2 \setminus t) \cup \{b\}$, and $(\mathbb{C}_1 \setminus a) \cap S^+ = \emptyset$, $b \notin S^+$, and t' is the top edge of C.

Case 5: $t \in S^-$ and $a, b \in S^+$. First, increase $w(a)$ to the largest value possible that does not create any uncovered unbalanced cycles. If afterwards $w(a) \geq w(t) - w(b)$, then we are done. Otherwise, $w(a) < w(t) - w(b)$ and there is some just balanced cycle $\mathbb{C}_1$ with $(\mathbb{C}_1 \setminus a) \cap S^+ = \emptyset$, which would be unbalanced if $w(a)$ were any larger. Now increase $w(b)$ to the largest value possible that does not create any new unbalanced cycles that are not non-top covered. If afterwards $w(b) \geq w(t) - w(a)$, then we are done. Otherwise, $w(b) < w(t) - w(a)$ and there is some just balanced cycle $\mathbb{C}_2$ with $(\mathbb{C}_2 \setminus b) \cap S^+ = \emptyset$, which would be unbalanced if $w(b)$ were any larger. Now decrease $w(t)$ to $w(a) + w(b)$. If this does not create any uncovered unbalanced cycle then we are done. Otherwise, some uncovered unbalanced cycle $\mathbb{C}_3$ is created which contains t and whose top edge is $t' \neq t$. Let $\pi(\mathbb{C}_1 \setminus a)$ denote the sub-path of $\mathbb{C}_1$ starting at the common vertex of a and t, and ending at the common vertex of a and b, let $\pi(\mathbb{C}_2 \setminus b)$ denote the sub-path of $\mathbb{C}_2$ starting at the common vertex of a and b, and ending at the common vertex of b and t, and let $\pi(\mathbb{C}_3 \setminus t)$ denote the sub-path of $\mathbb{C}_3$ starting at the common vertex of b and t, and ending at the common vertex of a and t. Consider the closed walk $\sigma = \pi(\mathbb{C}_1 \setminus a) \circ \pi(\mathbb{C}_2 \setminus b) \circ \pi(\mathbb{C}_3 \setminus t)$. As σ is closed it must contain some cycle C which includes the edge t'. Note that C is unbalanced with top edge t', since it only contains edges from $(\mathbb{C}_1 \setminus a) \cup (\mathbb{C}_2 \setminus b) \cup (\mathbb{C}_3 \setminus t)$, and so it holds that $w(C \setminus t') = w(C) - w(t') \leq w(\mathbb{C}_1 \setminus a) + w(\mathbb{C}_2 \setminus b) + w(\mathbb{C}_3 \setminus t) - w(t') = w(a) + w(b) - w(t) + w(\mathbb{C}_3 \setminus t') = w(\mathbb{C}_3 \setminus t') < w(t')$.

Since C is unbalanced, it must be either non-top covered by S^+ or top covered by t', which implies $\mathbb{C}_3$ is either non-top covered by S^+ or top covered by t'. This is because C only contains edges from $(\mathbb{C}_1 \setminus a) \cup (\mathbb{C}_2 \setminus b) \cup (\mathbb{C}_3 \setminus t)$, and $(\mathbb{C}_1 \setminus a) \cap S^+ = \emptyset, (\mathbb{C}_2 \setminus b) \cap S^+ = \emptyset$, and t' is the top edge of C.

In every case, the weight of at least one edge from S^+ was strictly increased or from S^- was strictly decreased, and any newly created unbalanced cycle is either top covered by S^- or non-top covered by S^+, and thus the conditions of the above described procedure are satisfied. □

3.5.2 The Result.

By the previous subsection, we know that optimal solutions to MVD are minimum regular covers of all unbalanced cycles. Surprisingly we can approximate such regular covers using the exact same algorithm used to approximate non-top covers, namely output a set $S = S_{\leq 6} \cup S_c$ where $S_{\leq 6}$ is a regular cover of all unbalanced cycles with ≤ 6 edges, and S_c is a set containing at least one chord from every induced 4-cycle of $S_{\leq 6}$. While this still produces a valid solution, the reason somewhat differs from before, as outlined in proof below. First, we need the following lemma, whose proof is omitted as it is identical to Lemma 3.4.6, except for the change from non-top to regular covers.

Lemma 3.5.3. *Let $G = (V, E)$ be an instance of MVD, let $C_{\leq k}$ be the set of all unbalanced cycles with at most k edges for some constant k, and let opt be a minimum size regular cover of $C_{\leq k}$. Then in polynomial time one can compute a constant factor approximation to opt.*

More precisely, in $O(|V|^k)$ time, one can compute a set $S_{\leq k}$ which regular covers $C_{\leq k}$, and such that $|S_{\leq k}| \leq k|opt|$.

Theorem 3.5.4. *Algorithm 2 gives a polynomial time $O(OPT^{1/3})$-approximation to any instance G of MVD (Problem 3.2.2), where OPT denotes the size of an optimal solution.*

Input : An instance G of MVD
Output : Valid solution S to the given instance.
1 Compute a regular cover $S_{\leq 6}$ of all unbalanced cycles with ≤ 6 edges using Lemma 3.5.3
2 Compute a cover $S_c = \mathit{chord4}(S_{\leq 6})$ using Corollary 3.4.11
3 Formulate and return any feasible solution to the LP described in Section 3.2.2 for the edge set $S = S_{\leq 6} \cup S_c$.

Algorithm 2: Finds a valid solution for MVD.

Proof. First we argue that Algorithm 2 returns a regular cover of all unbalanced cycles and hence is a valid solution by Lemma 3.5.2. So consider some unbalanced cycle C'. If it is regular covered by $S_{\leq 6}$ then we are done, so assume otherwise. We now argue C' must be regular covered by S_c. Among all unbalanced bottom cycles defined by a chord of C' such that the chord is the top edge of the bottom cycle and not covered by $S_{\leq 6}$, let $C = \mathit{bot}(C', e')$ be the one with the minimum number of edges. If C' has no such unbalanced non-regular covered bottom cycle set $C = C'$. Let t denote the top edge of C.

As C is not regular covered by $S_{\leq 6}$, clearly it has > 6 edges, and thus $\mathit{embed4}(C)$ is well defined (see Definition 3.4.8). Fix any edge $e \in \mathit{embed4}(C)$, and let $C_1 = \mathit{top}(C, e)$ and $C_2 = \mathit{bot}(C, e)$. We now argue $e \in S_{\leq 6}$. Note that C_1 has at most 6 edges, thus if C_1 is unbalanced, then it must be regular covered by $S_{\leq 6}$, and in particular this implies $e \in S_{\leq 6}$ since e is the only edge of C_1 not in C. Otherwise, C_1 is balanced, in which case $w(e) \geq w(t) - w(C_1 \setminus \{t, e\})$, which implies C_2 is unbalanced as $w(e) \geq w(t) - (w(C_1) - w(t) - w(e)) = 2w(t) - (w(C_1) - w(e)) = 2w(t) - (w(C) - w(C_2 \setminus e)) > w(C_2 \setminus e)$ since $w(t) > w(C) - w(t)$. Thus $C_2 = \mathit{bot}(C, e) = \mathit{bot}(C', e)$ is an unbalanced bottom cycle of C' with length less than C and top edge be e, and so must be regular covered by $S_{\leq 6}$. As C is not regular covered, and e is the only edge of C_2 not in C, this implies $e \in S_{\leq 6}$. Thus in either case, for any $e \in \mathit{embed4}(C)$, $e \in S_{\leq 6}$. This implies S_c will contain a chord of the cycle $\mathit{embed4}(C)$, and therefore C' will be regular covered by S_c.

The above argues we return a valid solution. The approximation ratio and time complexity analysis are nearly identical to that in the proof of Theorem 3.4.12, and so are omitted here. □

3.6 Matching Lower Bound

The following is a matching lower bound to the combinatorial problem from Lemma 3.4.10.

Lemma 3.6.1. *Let $G = (V, E)$ be a graph whose edge set E is the union of the edges of a collection of 4-cycles, $\mathbb{C}$, such that no two 4-cycles in $\mathbb{C}$ can share a chord. (Note this applies to all chords in the complete graph on V, i.e. regardless of whether they appear in E.) Then in the worst case $|\mathbb{C}| = \Omega(m^{4/3})$, where $m = |E|$.*

Proof. Construct a graph $G = (V, E)$, where V is the disjoint union of four sets of vertices X_1, X_2, X_3, and X_4, each containing exactly t vertices, where t is a value to be determined shortly. The edge set E is sampled as follows. For $1 \leq i \leq 4$, for each pair $(u, v) \in X_i \times X_{i+1}$ (where $X_5 = X_1$), the edge (u, v) is sampled into E independently with probability

$$p = \frac{m}{8t^2} \ll 1.$$

Let $\mathbb{C}$ be the set of 4-cycles defined by E. Any cycle $C \in \mathbb{C}$ must contain exactly one vertex from each of X_1, X_2, X_3, and X_4. The probability that any quadruple of vertices $(i_1, i_2, i_3, i_4) \in X_1 \times X_2 \times X_3 \times X_4$ defines a cycle in $\mathbb{C}$ is p^4. As such, the expected size of $\mathbb{C}$ is

$$\alpha = p^4 t^4 = \left(\frac{m}{8t^2}\right)^4 t^4 = \left(\frac{m}{8t}\right)^4.$$

Consider such a cycle $C = (i_1, i_2, i_3, i_4)$ that we know exists in the graph. Any cycle which shares the chord $\{i_1, i_3\}$ with C clearly shares the vertices i_1 and i_3. Now such a cycle either shares a third vertex or not. The expected number of cycles which share the chord $\{i_1, i_3\}$ and no other vertex is at most $t^2 p^4$. The expected number of cycles which share the chord

$\{i_1, i_3\}$ and one other vertex is at most $2tp^2$. Let X_C be a random variable denoting the number of cycles sharing either chord (i.e., $\{i_1, i_3\}$ or $\{i_2, i_4\}$) with C. Assuming $tp^2 \leq 1$ we have,

$$\mathbf{E}[X_C|\ C \text{ exists}] \leq 2(2tp^2 + t^2p^4) \leq 2(2 + tp^2)tp^2 \leq 6tp^2.$$

Assume further that $6tp^2 \leq 1/10$, then by Markov's inequality we have

$$\begin{aligned}\beta(C) &= \mathbf{Pr}[\text{no cycle shares a chord with } C \mid C \text{ exists}] \\ &= 1 - \mathbf{Pr}[X_C \geq 1|\ C \text{ exists}] \geq \frac{9}{10}.\end{aligned}$$

Let Y be a random variable denoting the number of cycles that exists in the graph and don't share a chord with any other cycle that exists in the graph. We have that

$$\begin{aligned}\delta &= \mathbf{E}[Y] \\ &= \sum_C \mathbf{Pr}[(\text{no cycle shares chord with } C) \cap (C \text{ exists})] \\ &= \sum_C \beta(C) \cdot \mathbf{Pr}[C \text{ exists}] \geq \frac{9}{10}\alpha.\end{aligned}$$

Note that as α was the expected number of cycles overall, this implies $\delta = \mathbf{E}[Y] = \Theta(\alpha)$.

Recall that we assumed $6tp^2 \leq 1/10$, which plugging in for p becomes,

$$\frac{1}{10} \geq 6t\Big(\frac{m}{8t^2}\Big)^2 \geq \frac{3}{32}\frac{m^2}{t^3} \quad \Longrightarrow \quad t \geq (30/32)^{1/3}m^{2/3}.$$

Thus setting $t = m^{2/3}$ (which up to constants minimizes α) implies the expected number of cycles that do not share a diagonal is

$$\begin{aligned}\delta = \Theta(p^4t^4) = \Theta\left(\left(\frac{m}{t^2}\right)^4 t^4\right) &= \Theta\left(\frac{m^4}{t^4}\right) \\ &= \Theta\left(\frac{m^4}{m^{8/3}}\right) = \Theta\big(m^{4/3}\big)\,.\end{aligned}$$

On the other hand, the expected number of edges is $4t^2p = m/2$, and moreover by the Chernoff bound with high probability is at most m. Thus by the probabilistic method there

exists a graph where $|E| \leq m$ and the number of 4-cycles which don't share a chord is $\Omega(m^{4/3})$. (Note to match the lemma statement, in the above construction one should only keep edges which were in cycles that did not share a chord with any other cycle.) $\square$

CHAPTER 4

GENERAL METRIC VIOLATION DISTANCE

4.1 Introduction

In this chapter we consider the *General Metric Violation Distance* (GMVD) problem, where the input is an undirected and positively-weighted graph, and the goal is to identify a minimum cardinality subset of edges whose weights can be modified such that the resulting graph is a metric graph, i.e. each edge is its own shortest path. Thus viewing the vertices as data points and edge weights as distances, GMVD models the problem of finding the nearest metric, where nearest is defined by the minimum number of distances values that need to be altered.

In the last chapter, we introduced the Metric Violation Distance (MVD) problem, which is a restriction of GMVD to instances where the input graph is complete. Thus MVD models the situation where one is given complete (though possibly erroneous) distance information. However, such complete information may not always be available (e.g. when measuring distances is expensive). By not requiring a complete graph, GMVD on the other hand can effectively model such incomplete data situations. Moreover, by considering the more general GMVD problem, in this chapter we are able to give stronger hardness results, and to provide a faster approximation algorithm, when compared to what was shown in Chapter 3 for MVD.

Related Work. Naturally, the most relevant previous work is Chapter 3, where we introduced the MVD problem, which as described above is a special case of the GMVD problem considered in this chapter. For MVD we gave an approximation preserving reduction from Vertex Cover, hence showing MVD is APX-hard, and moreover hard to approximate within a factor of 2 assuming the Unique Games Conjecture (UGC) [33]. We then provided an $O(OPT^{1/3})$-approximation, where OPT is the size of the optimal solution. The running time

of this algorithm is $\Omega(n^6)$, as it enumerates all cycles of length ≤ 6, and so the algorithm may not be practical from an implementation standpoint. The same hardness and approximation results were also shown for MVID, which is the variant where edge weights are only allowed to increase. The problem is polynomial time solvable when weights are only allowed to decrease.

GMVD is also related to a large number of other previously studied problems. A short list includes: metric nearness, seeking the metric minimizing the sum of distance value changes [5]; metric embedding with outliers, seeking the fewest points whose removal creates a metric [28]; matrix completion, seeking to fill missing matrix entries to produce a low rank [31]; and many more. We refer the reader to the previous chapter for a more detailed discussion of these and other problems. Gilbert and Sonthalia [34] and our paper [13] considered the GMVD problem respectively, however, we [13] focused on approximation and hardness while they [34] focused on FPT algorithm to the GMVD.

Instead here we focus on the deep connections to certain cutting problems, which underly several results in this chapter, and which were not discussed in Chapter 3. In particular, our problem is closely related to MULTICUT, where given a weighted graph G with k marked (s_i, t_i) vertex pairs, the goal is to find a minimum weight subset of edges S such that there is no path in $G \setminus S$ between s_i and t_i for any $1 \leq i \leq k$.

Our Results. In this chapter we introduce the GMVD problem. This a generalization of the MVD problem introduced in the chapter above, to allow missing edges, thus modeling the scenario of incomplete data. In addition to defining the GMVD problem, we prove the following results.

- In Section 4.3 we prove a key structural result for the GMVD problem. Define an unbalanced cycle to be a cycle whose largest edge weight is larger than the sum of the weights of all its other edges. Then we argue that in order for a subset of edges to be a valid solution to GMVD, it suffices for the subset to cover all unbalanced cycles.

As this condition is also trivially necessary, it gives a complete characterization of the GMVD solution set.

- In Section 4.4 we give polynomial-time approximation-preserving reductions from MULTICUT and LB-CUT to GMVD. This connection to the well studied MULTICUT problem is interesting in its own right, though it also implies that GMVD is NP-hard, and moreover cannot be approximated within any constant factor assuming UGC. Also recall that in general the best known approximation factor for MULTICUT is $O(\log n)$. Our reduction from LB-CUT implies that, for any fixed L, the set of instances of GMVD with maximum edge weight L (and minimum weight 1) are hard to approximate within a factor of $\Omega(\sqrt{L})$.
- In Section 4.5 we give an approximation algorithm for GMVD. Our approximation factor depends on *deficit* values, which for a given cycle is the largest single edge weight minus the sum of the weights of all its other edges. Note that the unbalanced cycles are precisely those with positive deficit. We give an $O(c \log n)$-approximation algorithm for GMVD assuming integer weights, where c is the number of distinct positive cycle deficit values in the input graph. Note there are several natural cases when c is small: when the number of unbalanced cycles is small; or for integer edge weights when all unbalanced cycles are within a bounded amount from being balanced; or alternatively when the maximum edge weight is bounded. Note also that our reductions in Section 4.4 suggest that the terms in our approximation factor may each be necessary in some form. In particular, for the $\log n$ term recall that we reduce from MULTICUT, whose best known approximation factor is $O(\log n)$. Moreover, for integer weights, if the maximum edge weight is L then $c \leq L$, and so our reduction from LB-CUT implies such instance are hard to approximate within a factor of $\Omega(\sqrt{c})$.

  Our algorithm runs in $O((n^3 + m^2) \cdot OPT \cdot c \log n)$ time and is practical in the sense that it only relies on basic counting and shortest path computations. It should be

noted that our new algorithm also works for MVD, as it is a special case of GMVD. Conversely, it is not obvious that the $OPT^{1/3}$-approximation from before for MVD would apply to GMVD, and moreover from a running time perspective, that algorithm must enumerate all cycles of length ≤ 6.

As was done in Chapter 3 for MVD, throughout the chapter we also consider the variant of GMVD where edge weights are only allowed to increase, denoted GMVID. For this problem, we show the same hardness and approximation results described above for GMVD. (We also get a similar sufficient condition in Section 4.3, except that cycles must be covered with edges other than their largest weight edge.) In fact, we argue in Section 4.4, that there is a polynomial time reduction from GMVID to GMVD. It should be noted, however, that it is not clear such a reduction from MVID to MVD holds.

4.2 Preliminaries

In Chapter 3, MVD and two restricted variants (MVDD and MVID) were also considered, where edge weights are only allowed to be decreased and increased, respectively. In this chapter we consider a more general version of the above problems, where the graph $G = (V, E)$ is undirected and positively weighted, though is not necessarily complete. We write $n = |V|$ and $m = |E|$, and for simplicity we assume G is connected and so we always have $n - 1 \leq m$. Recall that a dissimilarity graph G is a *metric graph* if it is its own metric completion, i.e. the weight $w(e)$ on each edge $e \in E$ is the shortest path distance between its endpoints. Define a *general dissimilarity graph* to be any undirected, and positively-weighted graph, and a *general metric graph* to be any general dissimilarity graph, $G = (V, E)$, such that $w(e) \leq d(e_u, e_v)$ on each edge $e \in E$, where $d(e_u, e_v)$ is the shortest path distance between the endpoints of u. (The word 'general' is sometimes dropped if understood from the context.)

Problem 4.2.1. (General Metric Violation Distance (GMVD)) *Given a general dissimilarity graph G, compute a minimum size set S of edges whose weights can be modified to convert G into a general metric graph.*

Again, we define two restricted variants GMVDD and GMVID, where edge weights are only allowed to be decreased and increased, respectively. In Chapter 3 it was argued that the optimal solution to MVDD is precisely the set of edges whose weight is larger than the shortest path distance between its endpoints. Hence MVDD is polynomial time solvable by computing the all pairs shortest paths distances, and it is not hard to see this similarly holds for GMVDD. Thus for the remainder of the chapter we only consider the GMVD and GMVID problems.

Covering Cycles.

We repeat some old defintions in Chapter 3 and define some new concepts here. Given an undirected graph $G = (V, E)$, a subgraph $C = (V', E')$ is called a k-cycle if $|V'| = |E'| = k$, and the subgraph is connected with every vertex having degree exactly 2. We often overload this notation and use C to denote either the cyclically ordered list of vertices or edges from this subgraph. We also often write $C \setminus e$ to denote the set of edges of C after removing the edge e, and $\pi(C \setminus e)$ to denote the corresponding induced path between the endpoints of e. Call any edge from G connecting two vertices which are non-adjacent in a given cycle, a *chord* of that cycle. Given a k-cycle in G, if the weight of a single edge is strictly larger than the sum of the weights of the other edges in the cycle, we say it is an *unbalanced k-cycle*. For a given unbalanced k-cycle, call its largest weight edge its *top edge* and the other edges of the cycle the *non-top* edges. Let the *deficit* of a cycle C, denoted $\delta(C)$, be equal to the weight of its top edge minus the sum of the weights of all other edges in C. Similarly, let $\delta(G)$ denote the maximum value of $\delta(C)$ over all cycles. Note the set of unbalanced cycles is equivalently

the set of cycles with strictly positive deficit. For three notions of covering unbalanced cycles (*regular cover*, *non-top cover*, *top cover*), see Chapter 3 for definitions with more details.

It is not hard to see that in order for a dissimilarity graph to be a metric graph, there cannot be any unbalanced cycles, and the same is true for general dissimilarity graphs being general metric graphs. (See Chapter 3 for a proof for dissimilarity graphs, which trivially extends to the general case.) Thus a solution to MVD or GMVD must necessarily cover all unbalanced cycles (and similarly non-top cover for MVID and GMVID, and top cover for MVDD and GMVDD). What is more surprising, is that for MVD this is also sufficient.

In Section 4.3 we generalize the Lemma 3.4.1 and Lemma 3.5.2 to GMVID and GMVD.

Feasibility Checking

Given a general dissimilarity graph G, let S be any subset of edges whose weights can be modified to convert G into a general metric graph. We now show that there is a linear program which determines how to set the weights of the edges in S, and thus in the remainder of this chapter we only need to focus on finding the set S. A similar linear program was described in Chapter 3, though here some slight modifications are needed as G is no longer assumed to be complete.

Let the vertices in $G = (V, E)$ be labeled $[n] = \{1, \ldots, n\}$. Then for all pairs $i, j \in [n]$, we define a (symmetric) variable $\alpha_{ij} = \alpha_{ji}$. To enforce that we get a general metric graph, we require that all triangle inequalities $\alpha_{ik} \leq \alpha_{ij} + \alpha_{jk}$ are satisfied. As only edges in S are allowed to change weights, we require for any $(i, j) \in E \setminus S$ that $\alpha_{ij} = w((i, j))$. We thus have the following.

$$\begin{aligned}
&\alpha_{ij} = \alpha_{ji} = w((i,j)) && \forall (i,j) \in E \setminus S \\
&\alpha_{ij} = \alpha_{ji} \geq 0 && \forall (i,j) \notin E \setminus S \\
&\alpha_{ik} \leq \alpha_{ij} + \alpha_{jk} && \forall \text{ distinct triples } \{i,j,k\}
\end{aligned}$$

We now argue the above LP is feasible if and only if weights of edges in S can be modified to convert G into a general metric graph. First, suppose we have such a set S, and that the edge weights in S have already been modified such that G is now a general metric graph. Then we argue setting $\alpha_{ij} = w((i,j))$ for all $(i,j) \in E$, and otherwise setting α_{ij} equal to the shortest path distance from i to j in G, gives a feasible LP solution. Specifically, by definition of general metric graphs, $\alpha_{ij} = w((i,j))$ is the shortest path distance between i and j in G, which is the same way we defined α_{ij} for all $(i,j) \notin E$. Thus as the triangle inequality holds for shortest path distances, $\alpha_{ik} = d(i,k) \leq d(i,j) + d(j,k) = \alpha_{ij} + \alpha_{jk}$, where $d(x,y)$ is the shortest path distance from x to y.

Now suppose that the LP has a feasible solution, and set the weight of each edge $(i,j) \in S$ equal to the $\alpha_{i,j}$ value from the LP. Consider any edge $(i,j) \in E$, and let $i = l_1, l_2, \ldots, l_k = j$ be the in order sequence of vertices in a shortest path from i to j in G. By definition, to argue that G (with the weights for edges in S set to their α_{ij} values) is a generalized metric graph, we need to show $w(l_1, l_k) \leq \sum_{x=1}^{k-1} w(l_x, l_{x+1})$. However, by repeated application of the triangle inequality, $w(l_1, l_k) = \alpha_{l_1,l_k} \leq \sum_{x=1}^{k-1} \alpha_{l_x,l_{x+1}} = \sum_{x=1}^{k-1} w(l_x, l_{x+1})$, thus proving the claim.

Later in the chapter we consider variants of GMVD where entries are only allowed to be increased. Note that the above linear program can trivially be modified to handle this case. Namely replace the second constraint with $\alpha_{ij} = \alpha_{ji} \geq w((i,j)) \;\; \forall (i,j) \in S$ and $\alpha_{ij} = \alpha_{ji} \geq 0 \;\; \forall (i,j) \notin E$.

4.3 Covering is Sufficient

Here we generalize Lemma 3.4.1 and Lemma 3.5.2 to the case of general dissimilarity graphs. These key structural results are needed to design our approximation algorithms. Also note our reduction from GMVID to GMVD in the next section depends on both Theorem 4.3.1 and Theorem 4.3.2 below.

Theorem 4.3.1. *If G is an instance of GMVID and S is a non-top cover of all unbalanced cycles, then G can be converted into a metric graph by only increasing weights of edges in S.*

Proof. For now assume all edge weights are integers and let L denote the largest edge weight. We describe a procedure which only modifies the weights of edges in S, producing a new instance G', such that (i) S remains a non-top cover of G', (ii) the weight of at least one edge strictly increases and none decrease, and (iii) no edge weight is ever increased above L. For any instance G (with at least one unbalanced cycle) and non-top cover S, if we prove such a procedure exists whenever not all the edges in S have weight L, then this will imply the lemma. Specifically, after applying the procedure at most $|S| \cdot L$ times, all edge weights will be equal to L, and hence no unbalanced cycle (and so no unsatisfied triangle inequality) can remain, since otherwise S would cover that unbalanced cycle with an edge of weight L, which is at least the weight of that cycle's top edge (i.e. the cycle is not actually unbalanced). Moreover, this procedure only increases weights of edges in S, as desired. Note also that as the number of steps in this existential argument was irrelevant, so long as it was finite, this procedure will imply the claim for rational input weights as well.

We now prove the above described procedure exists. Let G and S be as in the lemma statement, and fix any unbalanced cycle, C, which can be assumed to exist as otherwise the lemma is trivially true. Let t be the top edge of C, and let $C \setminus t$ denote the set of non-top edges of C. Let $F = (C \setminus t) \cap S$ be the set of non-top edges in C that are covered by S. Note that F is non-empty.

If there exists any edge $f \in F$ whose weight can be increased without creating any new unbalanced cycle which is not non-top covered, then we increase f by one, and move onto the next iteration of our procedure. Thus we can assume we have reached an iteration where no such edge exist. Thus for every edge $f \in F$, there must be some cycle $\mathbb{C}_f$, which is not non-top covered, and such that if we increased f at all then $\mathbb{C}_f$ would become unbalanced.

Note this implies $w(f) = \sum_{e \in (\mathbb{C}_f \setminus f)} w(e)$, and if increased f would be the top edge of this new unbalanced cycle.

Let $(t, e_1, e_2, \cdots, e_k)$ be a cyclic ordering of the edges in C. For any $e_i \in F$, let $\pi(\mathbb{C}_{e_i} \setminus e_i)$ denote the sub-path of cycle $\mathbb{C}_{e_i}$ starting at the common vertex of e_{i-1} and e_i, and ending at the common vertex of e_i and e_{i+1}. For any $e_i \in (C \setminus t)$, if $e_i \in F$ then define $\mathbb{P}_i = \pi(\mathbb{C}_{e_i} \setminus e_i)$, and otherwise define $\mathbb{P}_i = e_i$. Consider the closed walk $\sigma = \mathbb{P}_1 \circ \cdots \circ \mathbb{P}_k \circ t$. As σ is closed it must contain some cycle D which includes t. Since D is a cycle from σ, $w(D \setminus t) = w(D) - w(t) \leq w(\mathbb{P}_1) + \cdots + w(\mathbb{P}_k) = w(e_1) + \cdots + w(e_k) < w(t)$, where the last inequality follows since C is unbalanced. Thus by definition D is unbalanced with top edge t, and so by assumption must be non-top covered by S, which in turn implies some edge in $\sigma \setminus t$ lies in S. However, every edge in $\sigma \setminus t$ is either a non-top edge of C which is not in S, or a non-top edge of $\mathbb{C}_f$ for some $f \in F$, and we assumed all such cycles are not non-top covered. Thus in either case we get a contradiction.

Therefore, as long as there are unbalanced cycles, each step of the procedure strictly increases an edge from S and no edges get decreased. Since any newly created unbalanced cycle is still covered by S, conditions (i), (ii), and (iii) are satisfied. □

Theorem 4.3.2. *If G is an instance of GMVD and S is a regular cover of all unbalanced cycles, then G can be converted into a metric graph by only changing weights of edges in S.*

Proof. For now assume all edge weights are integers and let L denote the largest edge weight. First use Lemma 3.5.1 to partition S into two disjoint sets S^+ and S^-, such that every unbalanced cycle is either non-top covered by S^+ or top covered by S^-. We describe a procedure producing a new instance G', such that (i) every unbalanced cycle of G' is either non-top covered by S^+ or top covered by S^-, (ii) the weight of either one '+' edge strictly increases or one '−' edge strictly decreases (and no other edge weights are modified), and (iii) no edge weight is ever increased above L or below 0. Proving such a procedure exists

for any instance G and regular cover S, will imply the lemma. Specifically, after applying the procedure at most $|S| \cdot L$ times, all edge weights in S^+ will be equal to L, and all edge weights in S^- will be equal to 0. This implies there are no remaining unbalanced cycles, since otherwise the unbalanced cycle must either be non-top covered by an edge of weight L or top covered by an edge weight 0, in either case yielding the contradiction that the cycle was actually balanced. Moreover, this procedure correctly modifies edge weights from S according to their label. As the number of steps in this existential argument was irrelevant, so long as it was finite, this procedure will imply the claim for rational input weights as well.

We now prove the above described procedure exists. Let G and S be as in the lemma statement, and fix any unbalanced cycle, C, which can be assumed to exist as otherwise the lemma is trivially true. Let t be the top edge of C, and let $C \setminus t$ denote the set of non-top edges of C. Let $F = (C \setminus t) \cap S^+$ be the set of non-top edges in C that are covered by S^+. Note that F is non-empty or $t \in S^-$. If a cycle is not non-top covered by S^+ and not top covered by S^-, we say that cycle is *uncovered*.

If there exists any edge $f \in F$ whose weight can be increased without creating any new unbalanced cycle which is uncovered, then we increase f by one, and move onto the next iteration of our procedure. Similarly, we decrease t by one if $t \in S^-$ and decreasing t does not create any new unbalanced cycle which is uncovered. Thus we can assume we have reached an iteration where no such edge exist. Thus for every edge $f \in F$, there must be some cycle $\mathbb{C}_f$, which is uncovered, and such that if we increased f at all then $\mathbb{C}_f$ would become unbalanced. Note this implies $w(f) = \sum_{e \in (\mathbb{C}_f \setminus f)} w(e)$, and if increased f would be the top edge of this new unbalanced cycle. Similarly, if $t \in S^-$, then there must be some uncovered cycle $\mathbb{C}_t$ which would become unbalanced if t were decreased any further.

We now argue the existence of an unbalanced cycle C, in an iteration where weights in S^+ cannot be increased nor weights in S^- decreased as described above, will contradict the starting assumption that all unbalanced cycles are either non-top covered by S^+ or top

covered by S^-. We break the analysis into two cases. For both cases, let $(t, e_1, e_2, \cdots, e_k)$ be a cyclic ordering of the edges in C. For any $e_i \in F$, let $\pi(\mathbb{C}_{e_i} \setminus e_i)$ denote the sub-path of cycle $\mathbb{C}_{e_i}$ starting at the common vertex of e_{i-1} and e_i, and ending at the common vertex of e_i and e_{i+1}. For any $e_i \in (C \setminus t)$, if $e_i \in F$ then define $\mathbb{P}_i = \pi(\mathbb{C}_{e_i} \setminus e_i)$, and otherwise define $\mathbb{P}_i = e_i$.

Case 1: $t \notin S^-$. Consider the closed walk $\sigma = \mathbb{P}_1 \circ \cdots \circ \mathbb{P}_k \circ t$. As σ is closed it must contain some cycle D which includes t. Since D is a cycle from σ, $w(D \setminus t) = w(D) - w(t) \leq w(\mathbb{P}_1) + \cdots + w(\mathbb{P}_k) = w(e_1) + \cdots + w(e_k) < w(t)$, where the last inequality follows since C is unbalanced. Thus by definition D is unbalanced with top edge t, and so by assumption must either be top covered by S^- or non-top covered by S^+. Since $t \notin S^-$, this implies some edge in $\sigma \setminus t$ lies in S^+. However, every edge in $\sigma \setminus t$ is either a non-top edge of C which is not in S^+, or a non-top edge of $\mathbb{C}_f$ for some $f \in F$, and we assumed all such cycles are not non-top covered. Thus in either case we get a contradiction.

Case 2: $t \in S^-$. For $1 \leq i \leq k$, let $\mathbb{P}_i$ be as defined above, and consider the cycle $\mathbb{C}_t$ which by definition is uncovered and would become unbalanced, with top edge t', if t were decreased. Let $(t', g_1, g_2, \cdots, g_j, t, g_{j+1}, \cdots, g_l)$ be a cyclic ordering of the edges in $\mathbb{C}_t$. Consider the closed walk $\sigma = t' \circ g_1 \circ \cdots \circ g_j \circ \mathbb{P}_1 \circ \cdots \circ \mathbb{P}_k \circ g_{j+1} \circ \cdots \circ g_l$. As σ is closed it must contain some cycle D which includes t'. Since D is a cycle from σ, $w(D \setminus t') = w(D) - w(t') \leq w(g_1) + \cdots + w(g_j) + w(\mathbb{P}_1) + \cdots + w(\mathbb{P}_k) + w(g_{j+1}) + \cdots + w(g_l) = w(g_1) + \cdots + w(g_j) + w(e_1) + \cdots + w(e_k) + w(g_{j+1}) + \cdots + w(g_l) < w(g_1) + \cdots + w(g_j) + w(t) + w(g_{j+1}) + \cdots + w(g_l) = w(t')$, where the strict inequality follows since C is unbalanced. Thus by definition D is unbalanced with top edge t', and so by assumption must either be top covered by S^- or non-top covered by S^+. This implies either an edge in some $\mathbb{P}_i$ lies in S^+, some edge g_i lies in S^+, or $t' \in S^-$. However, by assumption, each $\mathbb{P}_i$ is either an uncovered non-top edge from C or is a path coming from an uncovered cycle $\mathbb{C}_i$, and hence no edge in $\mathbb{P}_i$ lies in S^+. Additionally, $\mathbb{C}_t$ is assumed to be uncovered and so no g_i lies in S^+, and t' is not in S^-. Thus we get a contradiction.

Thus as long as unbalanced cycles remain, at least one edge weight from S gets increased or decreased (and not above L or below 0) according to its label, and any newly created unbalanced cycle is already top covered by S^- or non-top covered by S^+. □

4.4 Hardness

In Chapter 3, we gave an approximation-preserving reduction from Vertex Cover to both MVD and MVID. Thus both are APX-complete, and in particular are hard to approximate within a factor of $2 - \varepsilon$ for any $\varepsilon > 0$, assuming the Unique Games Conjecture (UGC) [33]. In this section we give stronger hardness results for GMVID and GMVD by giving approximation-preserving reductions from MULTICUT and LB-CUT.

Problem 4.4.1 (MULTICUT)**.** *Given an undirected unweighted graph $G = (V, E)$ on $n = |V|$ vertices together with k pairs of vertices $\{s_i, t_i\}_{i=1}^k$, compute a minimum size subset of edges $M \subseteq E$ whose removal disconnects all the demand pairs, i.e., in the subgraph $(V, E \setminus M)$ every s_i is disconnected from its corresponding vertex t_i.*

[16] proved that if UGC is true, then it is NP-hard to approximate MULTICUT within any constant factor $L > 0$, and assuming a stronger version of the UGC, within $\Omega(\sqrt{\log \log n})$. (Note that the version of MULTICUT in [16] allows weights, but as the weights are polynomial, the authors remark that their hardness proofs extend to the unweighted case.)

Theorem 4.4.2. *There is an approximation-preserving, polynomial-time reduction from MULTICUT to GMVID.*

Proof. Let $G = (V, E)$ be an instance of MULTICUT with k pairs of vertices $\{s_i, t_i\}_{i=1}^k$. First, if $(s_i, t_i) \in E$ for any i, then that edge must be included in the solution M. Thus we can assume no such edges exists in the MULTICUT instance, as assuming this can only make it harder to approximate the optimum value of the MULTICUT instance. We now construct an

instance of GMVID, $G' = (V', E')$. Let $V' = V$ and $E' = E \cup \{s_i, t_i\}_{i=1}^{k}$ where the edges in E have weight 1 and the edges (s_i, t_i), for all $i \in [k]$, have weight $n = |V|$.

Observe that if a cycle in G' has exactly one edge of weight n then it must be unbalanced, since a cycle can have at most $n-1$ other edges (each of weight 1). The claim is that the converse is also true, that is any unbalanced cycle in G' must have exactly one edge of weight n (and note this is the top edge). Specifically, if the cycle has no n weight edges, then all edges have weight 1 and hence the cycle is balanced. If the cycle has more than one edge with weight n, then the cycle is also balanced, since for any edge, the total weight of the other edges in the cycle is $\geq n$, and no edge has weight larger than n.

Note that the edges from G are exactly the weight one edges in G', and thus the paths in G are in one-to-one correspondence with the paths in G' which consist of only weight one edges. Moreover, the weight n edges in G' are in one-to-correspondence with the (s_i, t_i) pairs from G. Thus the cycles in G' with exactly one weight n edge followed paths of all weight one edges connecting their endpoints, which by the above are exactly the set of unbalanced cycles, are in one-to-one correspondence with paths between (s_i, t_i) pairs from G. Therefore, a minimum cardinality subset of edges which non-top cover all unbalanced cycles, i.e. an optimal solution to GMVID, corresponds to a minimum cardinality subset of edges from E which cover all paths from s_i to t_i for all i, i.e. an optimal solution to MULTICUT. □

Problem 4.4.3 (LB-CUT)**.** *Given a value L and an undirected unweighted graph $G = (V, E)$ with source s and sink t, find a minimum size subset of edges $M \subseteq E$ such that no s-t-path of length less than or equal to L remains in the graph after removing the edges in M.*

An instance of LB-CUT with length bound L, is referred to as an instance of the L-LB-CUT problem. For any fixed L, Lee [21] showed that it is hard to approximate L-LB-CUT within a factor of $\Omega(\sqrt{L})$. The proof of the following theorem is similar to that of Theorem 4.4.2 (actually it is simpler as there is only one (s, t) pair).

Theorem 4.4.4. *For any fixed value L, there is an approximation-preserving, polynomial-time reduction from L-LB-CUT to GMVID.*

Proof. Let $G = (V, E)$ be an instance of L-LB-CUT with source s and sink t. First, if $(s, t) \in E$, then that edge must be included in the solution M. Thus we can assume that edge is not in the LB-CUT instance, as assuming this can only make it harder to approximate the optimum value of the LB-CUT instance. We now construct an instance of GMVID, $G' = (V', E')$. Let $V' = V$ and $E' = E \cup \{(s, t)\}$ where the edges in E have weight 1 and the edge (s, t) has weight $L + 1$.

First, observe that any cycle containing the edge (s, t) followed by $\leq L$ unit weight edges is unbalanced, as the sum of the unit weight edges will be $< L+1 = w((s, t))$. Conversely, any unbalanced cycle must contain the edge (s, t) followed by $\leq L$ unit weight edges. Specifically, if a cycle does not contain (s, t) then it is balanced since all edges would then have weight 1. Moreover, if a cycle contains (s, t) and $> L$ other edges, then the total sum of those unit edges will be $\geq L + 1 = w((s, t))$.

Note that the edges from G are exactly the weight one edges in G', and thus the paths in G are in one-to-one correspondence with the paths in G' which consist of only weight one edges. Moreover, the edge (s, t) in G' corresponds with the source and sink from G. Thus by the above, the unbalanced cycles in G' are in one-to-one correspondence with s-t-paths with length $\leq L$ in G. Therefore, a minimum cardinality subset of edges which non-top cover all unbalanced cycles, i.e. an optimal solution to GMVID, corresponds to a minimum cardinality subset of edges from E which cover all paths from s to t of length $\leq L$, i.e. an optimal solution to LB-CUT. □

We now reduce GMVID to GMVD, thus implying the above theorems also hold for GMVD.

Theorem 4.4.5. *There is an approximation-preserving, polynomial-time reduction from GMVID to GMVD.*

Proof. Let $G = (V, E)$ be an instance of GMVID. Find the set $T = \{(s_1, t_1), \ldots, (s_{|T|}, t_{|T|})\}$ of top edges of all unbalanced cycles by comparing the weight of each edge to the shortest path distance between its endpoints. We now construct an instance, $G' = (V', E')$, of GMVD. For all $1 \leq i \leq |T|$ and $1 \leq j \leq |E| + 1$, let $Q = \{v_{ij}\}_{i,j}$ be a set of vertices, and let $F_l = \{(s_i, v_{ij})\}_{i,j}$ and $F_r = \{(t_i, v_{ij})\}_{i,j}$ be sets of edges. Let $V' = V \cup Q$ and $E' = E \cup F_l \cup F_r$, where all (s_i, v_{ij}) edges in F_l have weight $L = 1 + \max_{e \in E} w(e)$, and for any i all (t_i, v_{ij}) edges in F_r have weight $L - w((s_i, t_i))$.

Let C be any unbalanced cycle in G with top edge (s_i, t_i) for some i. First, observe that the cycle $C' = (C \setminus (s_i, t_i)) \cup \{(s_i, v_{ij}), (t_i, v_{ij})\}$ is an unbalanced cycle with top edge (s_i, v_{ij}), for any j. To see this, note that $w((s_i, v_{ij})) = L = w((t_i, v_{ij})) + w((s_i, t_i))$. Thus since C is unbalanced,

$$w((s_i, v_{ij})) = w((t_i, v_{ij})) + w((s_i, t_i)) > w((t_i, v_{ij})) + w(C \setminus (s_i, t_i)),$$

and thus by definition C' is unbalanced with top edge (s_i, v_{ij}). Hence each unbalanced cycle C in G, with top edge (s_i, t_i), corresponds to $|E| + 2$ unbalanced cycles in G', namely, C itself and the cycles obtained by replacing (s_i, t_i) with a pair $(s_i, v_{ij}), (t_i, v_{ij})$, for any j.

We now show the converse, that any unbalanced cycle C' in G' is either also an unbalanced cycle C in G, or obtained by replacing (s_i, t_i) with $(s_i, v_{ij}), (t_i, v_{ij})$ for some j. First, observe that for any i, any cycle containing the edge (s_i, v_{ij}) must also contain the edge (t_i, v_{ij}), and moreover if a cycle containing such a pair is unbalanced, then its top edge must be (s_i, v_{ij}) as $w((s_i, v_{ij})) = L$. Similarly, any cycle containing more than one of these pairs of edges (over all i and j) must be balanced, since such cycles then would contain at least two edges with the maximum edge weight L. So let C' be any unbalanced cycle containing exactly one such (s_i, v_{ij}), (t_i, v_{ij}) pair. Note that C' cannot be the cycle $((s_i, v_{ij}), (t_i, v_{ij}), (s_i, t_i))$, as this cycle is balanced because $w((s_i, v_{ij})) = w((t_i, v_{ij})) + w((s_i, t_i))$. Otherwise, $C = C' \setminus \{(s_i, v_{ij}), (t_i, v_{ij})\} \cup \{(s_i, t_i)\}$ is also a cycle, and note that C' being unbalanced implies

C is unbalanced with top edge (s_i, t_i), implying the claim. This holds since

$$w(s_i, t_i) = w((s_i, v_{ij})) - w((t_i, v_{ij})) > w(C' \setminus (s_i, v_{ij})) - w((t_i, v_{ij})) = w(C \setminus (s_i, t_i)).$$

Now consider any optimal solution M to the GMVID instance G, which by Theorem 4.3.1 we know is a minimum cardinality non-top cover of G. By the above, we know that M is also a non-top cover of G', and hence is also a regular cover of G'. Thus by Theorem 4.3.2, M is a valid solution to the GMVD instance. Conversely, consider any optimal solution M' to the GMVD instance G', which by Theorem 4.3.2 is a minimum cardinality regular cover of G'. The claim is that M' is also a non-top cover of G, and hence is a valid solution to the GMVID instance. To see this, observe that since all unbalanced cycles in G are unbalanced cycles in G', M' must be a regular cover of all unbalanced cycles in G, and we now argue that it is in fact a non-top cover. Specifically, consider all the unbalanced cycles in G which have a common top edge (s_i, t_i). Suppose there is some cycle in this set, call it C, which is not non-top covered by M'. As M' is a regular cover for G', this implies that for any j, the unbalanced cycle described above determined by removing the edge (s_i, t_i) from C and adding edges (s_i, v_{ij}) and (t_i, v_{ij}), must be covered either with (s_i, v_{ij}) or (t_i, v_{ij}). However, as j ranges over $|E| + 1$ values, and these edge pairs have distinct edges for different values of j, this means M' has at least $|E| + 1$ edges. This is a clear contradiction with M' be a minimum sized cover, as any non-top cover of G is a regular cover of G', and G only has $|E|$ edges in total. □

Observe that in the above reduction, if the maximum edge weight in the GMVID instance was L then maximum edge weight in the corresponding GMVD instance is $L + 1$. Moreover, for the reduction in Theorem 4.4.4, for any instance of L-LB-CUT, the corresponding GMVID instance has maximum edge weight $L + 1$. Thus based on the above reductions, and previous known hardness results, we have the following.

Theorem 4.4.6. *GMVID and GMVD are APX-complete, and moreover assuming UGC neither can be approximated within any constant factor.*

For any fixed value L, consider the problem defined either by the restriction of GMVID or GMVD to the subset of instances with maximum edge weight L (and minimum edge weight 1), then assuming UGC this problem is hard to approximate within a factor of $\Omega(\sqrt{L})$.

Note our reduction in Theorem 4.4.4 actually implies a stronger second statement than what is given in the above theorem, as it reduces to an instance of GMVID with all unit weight edges except for a single $L + 1$ weight edge. However, we find the above statement more natural, and it is sufficient for our purposes.

4.5 Approximation Algorithm

In this section we present approximation algorithms for the GMVID and GMVD problems. As the algorithms are nearly identical for the two cases, we present the algorithm for GMVD first, and then remark on the minor change needed to apply it to GMVID.

By Theorem 4.3.2, we know that an optimal solution to GMVD is a minimum cardinality regular cover of all unbalanced cycles. This naturally defines a hitting set instance $(E, \mathcal{C})$, where the ground set E is the edges from G, and $\mathcal{C}$ is the collection of the subsets of edges determined by the unbalanced cycles. Thus if for any edge $e \in E$ we could compute the number of unbalanced cycles it participates in, then immediately we get an $O(\log n)$ approximation for GMVD by running the standard greedy algorithm for hitting set. Namely, while unbalanced cycles remain, we simply repeatedly remove the edge hitting the largest number of remaining unbalanced cycles (as removing the edge effectively removes the sets it hit from the hitting set instance). However, there may be an exponential number of unbalanced cycles. Note that in general just counting the number of simple paths in a graph is #P-Hard [35], though it is known how to count paths of length up to roughly $O(\log n)$

using the color-coding technique. (See for example [36] and references therein. Also see [37] for recent FPT algorithms.) Moreover, observe that our situation is more convoluted as we only wish to count paths corresponding to unbalanced cycles.

Despite the challenge of counting unbalanced cycles, we are able to get an approximation by making the key observation that a cycle with the largest deficit value must correspond to a shortest path, which in turn allows us to quickly get a count when restricting to such cycles. Thus our approach is to iteratively handle covering cycles by decreasing deficit value, ultimately breaking the problem into multiple hitting set instances.

For any pair of vertices $s, t \in V$, we write $d(s,t)$ to denote their shortest path distance in G, and $\mathsf{csp}(s,t)$ to denote the number of shortest paths from s to t.

Lemma 4.5.1. *Let G be a positively weighted graph, where for all pairs of vertices u, v one has constant time access to the value $d(u,v)$. Then for any pair of vertices s, t, the value $\mathsf{csp}(s,t)$ can be computed in $O(m + n \log n)$ time.*

Proof. Let $V = \{v_1, v_2, v_3, ..., v_n\}$, and let $N(v_i)$ denote the set of neighbors of v_i. Define $X_i = \{v_j \in N(v_i) \mid w(v_i, v_j) + d(v_j, t) = d(v_i, t)\}$, that is, X_i is the set of neighbors of v_i where there is a shortest path from t to v_i passing through that neighbor. Thus we have,

$$\mathsf{csp}(v_i, t) = \sum_{v_j \in X_i} \mathsf{csp}(v_j, t).$$

Note that any shortest path from v_i to t can only use vertices v_j which are closer to t than v_i. Thus assume we have sorted and labeled the vertices $t = v_1, v_2, v_3, ..., v_n$ according to increasing order of their distance $d(v_i, t)$ from t. Thus if we compute the $\mathsf{csp}(v_i, t)$ values in increasing order of the index i, then each $\mathsf{csp}(s, v_j)$ value can be computed in time proportional to the degree of v_i, and so the overall running time is $O(m + n \log n)$. □

Recall that for an unbalanced cycle C with top edge t, the deficit of C is $\delta(C) = w(t) - \sum_{e \in (C \setminus t)} w(e)$. Moreover, $\delta(G)$ is used to denote the maximum deficit over all cycles in

G. For any edge e, define $N_T(e, \alpha)$ to be the number of distinct unbalanced cycles of deficit α whose top edge is e. Similarly, let $N_U(e, \alpha)$ denote the number of distinct unbalanced cycles with deficit α which contain the edge e, but where e is not the top edge.

Lemma 4.5.2. *For any edge $e = (s, t)$, if $w(e) = d(s, t) + \delta(G)$ then $N_T(e, \delta(G)) = \mathsf{csp}(s, t)$, and otherwise $N_T(e, \delta(G)) = 0$.*

Proof. If $w(e) \neq d(s, t) + \delta(G)$, then as $\delta(G)$ is the maximum deficit over all cycles, it must be that $w(e) < d(s, t) + \delta(G)$, which in turn implies any unbalance cycle with top edge e has deficit strictly less than $\delta(G)$. Now suppose $w(e) = d(s, t) + \delta(G)$, and consider any path $p_{s,t}$ from s to t such that e together with $p_{s,t}$ creates an unbalanced cycle with top edge e. If $p_{s,t}$ is a shortest path then $w(e) - w(p_{s,t}) = w(e) - d(s, t) = \delta(G)$, and otherwise $w(p_{s,t}) > d(s, t)$ and so $w(e) - w(p_{s,t}) < w(e) - d(s, t) = \delta(G)$. Thus $N_T(e, \delta(G)) = \mathsf{csp}(s, t)$ as claimed. □

As G is undirected, every edge $e \in E$ corresponds to some unordered pair $\{a, b\}$. However, often we write $e = (a, b)$ as an ordered pair, according to some fixed arbitrary total ordering of all the vertices. We point this out to clarify the following statement.

Lemma 4.5.3. *Fix any edge $e = (s, t)$, and let $X = \{f = (\alpha, \beta) \mid w(f) = d(\alpha, s) + w(e) + d(t, \beta) + \delta(G)\}$, and $Y = \{f = (\alpha, \beta) \mid w(f) = d(\beta, s) + w(e) + d(t, \alpha) + \delta(G)\}$. Then it holds that*

$$N_U(e, \delta(G)) = \left(\sum_{(\alpha,\beta) \in X} \mathsf{csp}(\alpha, s) \cdot \mathsf{csp}(t, \beta) \right) + \left(\sum_{(\alpha,\beta) \in Y} \mathsf{csp}(\beta, s) \cdot \mathsf{csp}(t, \alpha) \right).$$

Proof. Consider any unbalanced cycle C containing $e = (s, t)$, with top edge $f = (\alpha, \beta)$ and where $\delta(C) = \delta(G)$. Such a cycle must contain a shortest path between α and β, as otherwise it would imply $\delta(G) > \delta(C)$. Now if we order the vertices cyclically, then the subset of C's vertices $\{\alpha, \beta, s, t\}$, must appear either in the order α, s, t, β or β, s, t, α. In the former case, as the cycle must use shortest paths, $w(f) = d(\alpha, s) + w(e) + d(t, \beta) + \delta(G)$,

and the number of cycles satisfying this is $\mathsf{csp}(\alpha, s) \cdot \mathsf{csp}(t, \beta)$. In the latter case, $w(f) = d(\beta, s) + w(e) + d(t, \alpha) + \delta(G)$, and the number of cycles satisfying this is $\mathsf{csp}(\beta, s) \cdot \mathsf{csp}(t, \alpha)$. Note also that the set X from the lemma statement is the set of all $f = (\alpha, \beta)$ satisfying the equation in the former direction, and Y is the set of all $f = (\alpha, \beta)$ satisfying the equation in the later direction. Thus summing over each relevant top edge in X and Y, of the number of unbalanced cycles of deficit $\delta(G)$ which involve that top edge and e, yields the total number of unbalanced cycles with deficit $\delta(G)$ containing e as a non-top edge. □

Corollary 4.5.4. *Given constant time access to $d(u, v)$ and $\mathsf{csp}(u, v)$ for any pair of vertices u and v, $N_T(e, \delta(G))$ can be computed in $O(1)$ time and $N_U(e, \delta(G))$ can be computed in $O(m)$ time.*

Proof. By Lemma 4.5.2, in constant time we can check whether $w(e) = d(s, t) + \delta(G)$, in which case we set $N_T(e, \delta(G)) = \mathsf{csp}(s, t)$, and otherwise we set $N_T(e, \delta(G)) = 0$. By Lemma 4.5.3, we can compute $N_U(e, \delta(G))$ with a linear scan over the edges, where for each edge f in constant time we can compute whether $w(f) = d(\alpha, s) + w(e) + d(t, \beta) + \delta(G)$ and if so add $\mathsf{csp}(\alpha, s) \cdot \mathsf{csp}(t, \beta)$ to the sum over X, and similarly if $w(f) = d(\beta, s) + w(e) + d(t, \alpha) + \delta(G)$ then add $\mathsf{csp}(\beta, s) \cdot \mathsf{csp}(t, \alpha)$ to the sum over Y. □

Theorem 4.5.5. *For any positive integer c, consider the set of GMVD instances where the number of distict deficit values is at most c, i.e. $|\{\delta(C) \mid C \text{ is a cycle in } G\}| \leq c$. Then Algorithm 3 gives an $O((n^3 + m^2) \cdot OPT \cdot c \log n)$ time $O(c \log n)$-approximation, where OPT is the size of the optimal solution.*

Proof. Observe that the algorithm terminates only when $\delta(G) = 0$, i.e. only once there are no unbalanced cycles left. As no new edges are added, and weights are never modified, this implies that when the algorithm terminates it outputs a valid regular cover S. (The algorithm must terminate as every round removes an edge.) Therefore, by Theorem 4.3.2, S is a valid GMVD solution, and so we only need to bound its size.

Input : An instance $G = (V, E)$ of GMVD
Output : A valid solution S to the given instance.

1 Let $S = \emptyset$
2 **while** *True* **do**
3 For every pair $s, t \in V$ compute $d(s, t)$
4 Compute $\delta(G) = \max_{e=(s,t)\in E}\ w(e) - d(s, t)$
5 **if** $\delta(G) = 0$ **then**
6 **return** S
7 **end**
8 For every edge $(s, t) \in E$ compute $\mathsf{csp}(s, t)$
9 For every $e \in E$ compute $count(e) = N_T(e, \delta(G)) + N_U(e, \delta(G))$
10 Set $f = \arg\max_{e\in E} count(e)$
11 Update $S = S \cup \{f\}$ and $G = G \setminus f$
12 **end**

Algorithm 3: Finds a valid solution for GMVD.

Let the edges in $S = \{s_1, \ldots, s_k\}$ be indexed in increasing order of the loop iteration in which they were selected. Let $G_1, \ldots, G_{k+1}$ be the corresponding sequence of graphs produced by the algorithm, where $G_i = G \setminus \{s_1, \ldots, s_{i-1}\}$. Note that for all i, $G_i = (V, E_i)$ induces a corresponding instance of hitting set, $(E_i, \mathcal{C}_i)$, where the ground set is the set of edges from the GMVD instance G_i, and $\mathcal{C}_i = \{E_i(C) \mid C \text{ is an unbalanced cycle in } G_i\}$ (where $E_i(C)$ is the set of edges in C).

Let $D = \{\delta(C) \mid C \text{ is a cycle in } G\}$, where by assumption $|D| \leq c$. Note that any cycle C in any graph G_i, is also a cycle in G. Thus as we never modify edge weights, $\delta(G_1), \ldots, \delta(G_{k+1})$ is a decreasing sequence. Moreover $X = \{\delta(G_i)\}_i \subseteq D$, and in particular $|X| \leq c$. For a given value $\delta \in X$, let $G_\alpha, G_{\alpha+1}, \ldots, G_\beta$ be the subsequence of graphs with deficit δ (which is consecutive as the deficit values are decreasing). Observe that for all $\alpha \leq i \leq \beta$, the edge s_i is an edge from a cycle with deficit $\delta = \delta(G_i)$. So for each $\alpha \leq i \leq \beta$, define a sub-instance of hitting set $(E'_i, \mathcal{C}'_i)$, where E'_i is the set of edges in cycles of deficit δ from G_i, and $\mathcal{C}'_i$ are the subsets of edges from cycles of deficit δ in G_i.

The claim is that for the hitting set instance $(E'_\alpha, \mathcal{C}'_\alpha)$, that $\{s_\alpha, \ldots, s_\beta\}$ is an $O(\log n)$ approximation to the optimal solution. To see this, observe that for any $\alpha \leq i \leq \beta$ in line 9,

count(e) is the number of times e is contained in an unbalanced cycle with deficit $\delta = \delta(G_i)$, as by definition $N_T(e, \delta(G_i))$ and $N_U(e, \delta(G_i))$ count the occurrences of e in such cycles as a top edge or non-top edge, respectively. Thus s_i is the edge in E'_i which hits the largest number of sets in $\mathcal{C}'_i$, and moreover, $(E'_{i+1}, \mathcal{C}'_{i+1})$ is the corresponding hitting set induced by removing s_i and the sets it hit from $(E'_i, \mathcal{C}'_i)$. Thus $\{s_\alpha, \ldots, s_\beta\}$ is the resulting output of running the standard greedy hitting set algorithm on $(E'_\alpha, \mathcal{C}'_\alpha)$ (that repeatedly removes the element hitting the largest number of sets), and it is well known this greedy algorithm produces an $O(\log n)$ approximation.

The bound on the size of S now easily follows. Specifically, let $I = \{i_1, i_2, \ldots, i_{|X|}\}$ be the collection of indices, where i_j was the first graph considered with deficit $\delta(G_{i_j})$. By the above, S is the union of the $O(\log n)$-approximations to the sequence of hitting set instance $(E'_{i_1}, \mathcal{C}'_{i_1}), \ldots, (E'_{i_{|X|}}, \mathcal{C}'_{i_{|X|}})$. In particular, note that for all i_j, $(E'_{i_j}, \mathcal{C}'_{i_j})$ is a hitting set instance induced from the removal of a subset of edges from the initial hitting set instance $(E_1, \mathcal{C}_1)$, and then further restricted to sets from cycles with a given deficit value. Thus the size of the optimal solution on each of these instances can only be smaller than on $(E_1, \mathcal{C}_1)$. This implies that the total size of the returned set S is $O(OPT \cdot |X| \log n) = O(OPT \cdot c \log n)$.

As for the running time, first observe that by the above, there are $O(OPT \cdot c \log n)$ while loop iterations. For a given loop iteration, computing all pairwise distance in line 3 takes $O(n^3)$ time using the standard Floyd-Warshall algorithm. Computing the graph deficit in line 4 can then be computed in $O(m)$ time. For any given vertex pair s, t, computing $\mathsf{csp}(s, t)$ takes $O(m + n \log n)$ time by Lemma 4.5.1. Thus computing the number of shortest paths over all edges in line 8 takes $O(m^2 + mn \log n)$ time. For each edge e, by Corollary 4.5.4, *count*$(e) = N_T(e, \delta(G)) + N_U(e, \delta(G))$ can be computed in $O(m)$ time, and thus computing all counts in line 9 takes $O(m^2)$ time. As the remaining steps can be computed in linear time, each while loop iteration in total takes $O(n^3 + mn \log n + m^2) = O(n^3 + m^2)$ time, thus implying the running time bound over all iterations in the theorem statement. □

Remark 4.5.6. *In the above our goal was to present the algorithm and analysis in simple and practical terms. However, it should be noted that the running time can be improved, though potentially at the cost of added complication. In particular, rather than computing the $d(u, v)$ values from scratch in each iteration, we can use a dynamic data structure. This would slightly improve the above running time to $O(n^3 + (n^{2+\alpha} + m^2) \cdot OPT \cdot c \log n)$, where $0 \leq \alpha$ is a constant depending on the query and update time of the dynamic data structure. (Ignoring* log *factors, $\alpha = 3/4$ is known. See for example the recent paper [38] and references therein). However, similarly improving the m^2 term in the running time seems more challenging as the $N_U(e, \delta(G))$ values depend in a non-trivial way on collections of $d(u, v)$ values, each of which may or may not have changed.*

Throughout this section we considered the GMVD problem. A similar result holds for GMVID, and there are a couple ways to achieve it. First, recall that Theorem 4.4.5 gave a polynomial-time approximation-preserving reduction from GMVID to GMVD, and hence this reduction and the above algorithm could be combined to yield an algorithm for GMVID. One issue with this approach is that the reduction in Theorem 4.4.5 increases the graph size by a linear factor, resulting in a slower running time. An alternative and simpler approach is to observe that the above proof and algorithm will work nearly identically for GMVID, except that by Theorem 4.3.1 the sets in the corresponding hitting set instance should not include the top edge of each cycle. We thus have the following.

Theorem 4.5.7. *For any positive integer c, consider the set of GMVID instances where the number of distict deficit values is at most c, i.e. $|\{\delta(C) \mid C \text{ is a cycle in } G\}| \leq c$. Then Algorithm 3, where line 9 instead sets $count(e) = N_U(e, \delta(G))$, gives an $O((n^3 + m^2) \cdot OPT \cdot c \log n)$ time $O(c \log n)$-approximation.*

Note that the GMVD and GMVID problems are phrased in terms of finding minimum sized edge sets whose weights can be modified. To determine how to modify the weights of the output edges, recall that this can be done with the LP in Section 4.2.

CHAPTER 5

COMPUTING THE FRÉCHET GAP DISTANCE

5.1 Introduction

Polygonal curves arise naturally in the modeling of a number of computational problems, and for such problems assessing the similarity of two curves is one of the most fundamental tasks. There are several competing measures for defining curve similarity. Among these, there has been strong interest in the Fréchet distance, particularly from the computational geometry community. In this chapter we consider a variant called the *Fréchet gap distance*, originally introduced by Filtser and Katz in the context of the discrete Fréchet distance [39]. Recall the analogy for the standard Frechet distance, where the goal is to use the minimum leash length possible for a man on one curve to walk a dog on the other curve, from start to end of their respective curves. In the gap variant the goal is instead to minimize the difference of the lengths of the longest and shortest leashes used over the entire walk. As discussed in [39], since this measure considers both the closest and farthest relative positions of the man and dog, in many cases it is closer to our intuitive notion of curve similarity. Notably, two translated copies of the *same* curve have Fréchet gap distance zero, as opposed to the magnitude of the translation under the standard Fréchet distance. Though this is not to say that it is the same as minimizing the standard Fréchet distance under translation. For instance, fix any two points on a rigid body in two or three dimensions. The pair of curves traced out by these points as we arbitrarily rotate and translate the rigid body will always have Fréchet gap distance zero (see Figure 5.1).

A natural scenario for the gap distance is planning the movement of military units, where one wants them to be sufficiently close to support each other in case of need, but sufficiently far from each other to avoid unintended interaction (i.e., friendly fire). Such units might move on two major roads that are roughly parallel to each other, thus matching our setup.

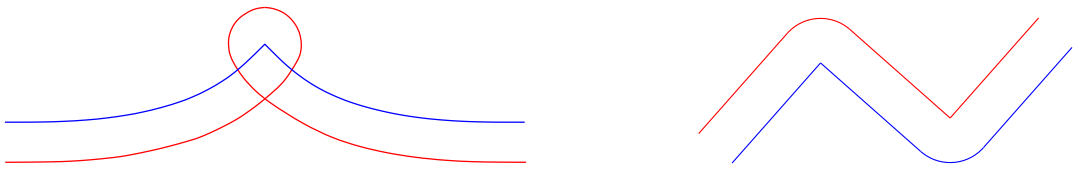


Figure 5.1: Left: A 2D "airplane roll". Right: Turning in 2D by pivoting on one side at a time.

Previous Work. Alt and Godau [11] presented an $O(n^2 \log(n))$ time algorithm to compute the standard Fréchet distance. More recently Buchin et al. [40] improved the logarithmic factor in the running time (building on [41]), however Bringmann [42] showed that assuming the Strong Exponential Time Hypothesis (SETH), no strongly subquadratic time algorithm is possible. Moreover, Bringmann showed that assuming SETH there is no strongly subquadratic 1.001-approximation algorithm, thus ruling out the possibility of a strongly subquadratic PTAS for general curves. On the other hand, there are fast approximation algorithms for several families of nicely behaved curves, for example Driemel et al. [43] gave an $O(cn/\varepsilon + cn \log n)$ time algorithm for the case of c-packed curves. The Fréchet distance and related measures have been used for a variety of applications [44, 45, 46, 47, 48].

Many variants of the Fréchet distance between polygonal curves have been considered. Daescu [49] considered the Fréchet distance in weighted regions, where the distance between two points is the length of the shortest path between the points. Alt and Godau [11] gave a quadratic time algorithm for the weak Fréchet distance, where backtracking on the curves is allowed. Driemel and Har-Peled [50] considered allowing shortcuts between vertices, and for this more challenging variant, they give a near linear time 3-approximation for c-packed curves. Later Buchin et al. [51] proved the general version, where shortcutting is also allowed on edge interiors, is NP-hard (and gave an approximation for the general case and an exact algorithm for the vertex case). The *discrete* Fréchet distance only considers distances at the vertices of the polygonal curves, i.e. rather than a continuously walking man and dog, there is a pair of frogs hopping along the vertices. This somewhat simpler variant can be

solved in $O(n^2)$ time using dynamic programming [52]. Interestingly, Agarwal et al. [41] showed the discrete variant can be solved in weakly subquadratic $O(n^2 \log\log n/\log n)$ time, however the above results of Bringmann [42] also imply there is no strongly subquadratic algorithm for the discrete case, assuming SETH. Avraham et al. [53] considered shortcuts in the discrete case, providing a strongly subquadratic running time, showing shortcuts make it more tractable, which was the reverse for the continuous case.

Minimizing Fréchet distance under translation (and other transformations) was previously considered, though running times are typically large. For example, Alt et al. [54] gave a roughly $O(n^8)$ time algorithm, though they also gave a $O(n^2/\varepsilon^2)$ time $(1+\varepsilon)$-approximation. Avraham et al. [55] consider the discrete case, and provide a nice summary of other previous work. The Fréchet distance has also been extended to more general inputs, such as graphs [56], piecewise smooth curves [57], simple polygons [58], surfaces [59], and complexes [60]. In general there are too many Fréchet distance results to cover, and the above is just a sampling.

The most relevant previous work is that of Filtser and Katz [39], who first proposed the Fréchet gap distance. The technical content of the two papers differs significantly however, as [39] considers the discrete case, avoiding many of the difficulties faced in our continuous setting. In particular, a solution to the gap problem is a distance interval. In the continuous case the challenge is bounding the number of possible intervals, while in the discrete case a bound of $O(n^4)$ holds, as each interval endpoint is a vertex to vertex distance. Using a result of Avraham et al. [55], Filtser and Katz improve this to an $O(n^3)$ time algorithm to compute the minimum discrete Fréchet gap. They also provide $O(n^2 \log^2 n)$ time algorithms for one-sided discrete Fréchet gap with shortcuts and the weak discrete Fréchet gap distance.

Contributions and Overview. Here we consider the continuous Fréchet gap distance problem (defined informally above, and formally below). This is the first to consider the more challenging continuous version of this problem. For this problem we provide an $O(n^5 \log n)$

time exact algorithm and a more efficient $O(n^2 \log n + \frac{n^2}{\varepsilon} \log \frac{1}{\varepsilon})$ time $(1+\varepsilon)$-approximation algorithm, and we now outline our approach and main contributions.

The standard approach for computing the Fréchet distance starts by solving the decision version for a given query distance $\delta \geq 0$, by using the free space diagram, which describes the pairs of points (one from each curve) which are within distance δ. The convexity of the free space cells allows one to efficiently propagate reachibility information, leading to a quadratic time proceedure overall. For the Fréchet gap problem the free space cells are no longer convex, but despite this we show that they have sufficient structure to allow efficient reachability propagation, again leading to a quadratic time decider, which in our case determines whether a given query interval $[s, t]$ is feasible.

The next step in computing the Fréchet distance is to find a polynomially sized set of critical events, determined by the input curves, to search over. For the standard Fréchet distance this set has $O(n^3)$ size. For the Fréchet gap case however the number of critical events can be much larger as they are determined by two rather than one distance value. As mentioned above, for the discrete case only pairs of vertex distances are relevant and so there are $O(n^4)$ events. On the other hand, for the continuous case there can now be "floating" monotonicity events where increasing (or decreasing) the gap interval endpoint values simultaneously may lead to an entire continuum of optimum intervals. Despite this we show there is an $O(n^6)$ sized set of canonical intervals containing an optimum solution.

The last step is efficiently searching over the critical events. For the standard Fréchet distance this can be done via parametric search [11] or sampling [60], yielding an $O(n^2 \log n)$ running time. Searching in the gap case however is more challenging, as there is no longer a natural linear ordering of events. Specifically, the set of feasible intervals may not appear contiguously when ordering candidate intervals by width. Despite this, we similarly get a near linear factor speed up, by using a more advanced version of the basic approach in [60].

Our approximation uses the observation that all feasible intervals share a common value. Roughly speaking, at the cost of a 2-approximation, this allows us to consider the radius of

intervals centered at this common value, rather than two independent interval endpoints, reducing the number of critical events. This is improved to a $(1+\varepsilon)$-approximation, and finally the running time is reduced by a linear factor, again using a modified version of [60].

5.2 Preliminaries

Throughout, given points $p, q \in \mathbb{R}^d$, $||p-q||$ denotes their Euclidean distance. Moreover, given two (closed) sets $P, Q \subseteq \mathbb{R}^d$, $dist(P,Q) = \min_{p\in P, q\in Q} ||p-q||$ denotes their distance.

5.2.1 Fréchet Distance and Fréchet Gap Distance

A *polygonal curve* π of length n is a continuous mapping from $[0,n]$ to $\mathbb{R}^d$, such that for any integer $0 \leq i < n$, the restriction of π to the interval $[i, i+1]$ is defined by $\pi(i+\alpha) = (1-\alpha)\pi(i) + \alpha\pi(i+1)$ for any $\alpha \in [0,1]$, i.e. a straight line segment. When it is clear from the context, we often use π to denote the image $\pi([0,n])$. The set of vertices of π is defined as $V(\pi) = \{\pi_0, \pi_1 \ldots, \pi_n\}$, where $\pi_i = \pi(i)$, and the set of edges is $E(\pi) = \{\pi_0\pi_1, \ldots, \pi_{n-1}\pi_n\}$, where $\pi_i\pi_{i+1}$ is the line segment connecting π_i and π_{i+1}.

A reparameterization for a curve π of length n is a continuous non-decreasing bijection $f : [0,1] \to [0,n]$ such that $f(0) = 0, f(1) = n$. Given reparameterizations f, g of an n length curve π and an m length curve σ, respectively, the *width* between f and g is defined as

$$width_{f,g}(\pi,\sigma) = \max_{\alpha\in[0,1]} ||\pi(f(\alpha)) - \sigma(g(\alpha))||$$

The (standard) *Fréchet distance* between π and σ is then defined as

$$d_{\mathcal{F}}(\pi,\sigma) = \min_{f,g} width_{f,g}(\pi,\sigma)$$

where f, g range over all possible reparameterizations of π and σ.

A *gap* is an interval $[s,t]$ where $0 \leq s \leq t$ are real numbers, and the *gap width* is $t-s$. Similarly, given reparameterizations f, g for curves π, σ, define their gap and gap width as

$$gap_{f,g}(\pi,\sigma) = \left[\min_{\alpha\in[0,1]} ||\pi(f(\alpha)) - \sigma(g(\alpha))||,\ \max_{\alpha\in[0,1]} ||\pi(f(\alpha)) - \sigma(g(\alpha))||\right]$$

$$gapwidth_{f,g}(\pi,\sigma) = \max_{\alpha\in[0,1]} ||\pi(f(\alpha)) - \sigma(g(\alpha))|| - \min_{\alpha\in[0,1]} ||\pi(f(\alpha)) - \sigma(g(\alpha))||$$

The *Fréchet gap distance* between two curves π and σ is then defined as

$$d_{\mathcal{G}}(\pi,\sigma) = \min_{f,g} gapwidth_{f,g}(\pi,\sigma)$$

where f, g range over all possible reparameterizations of π and σ.

If there exist reparameterizations f and g for curves π and σ satisfying the inequalities,

$$\max_{\alpha\in[0,1]} ||\pi(f(\alpha)) - \sigma(g(\alpha))|| \leq t \qquad\qquad \min_{\alpha\in[0,1]} ||\pi(f(\alpha)) - \sigma(g(\alpha))|| \geq s,$$

we say $[s,t]$ is a *feasible gap* between curves π and σ. Throughout the chapter $[s^*,t^*]$ denotes an arbitrary optimal gap, that is $t^* - s^* = d_{\mathcal{G}}(\pi,\sigma)$. (Note there may be more than one such optimal gap, and moreover a feasible gap does not necessarily contain an optimal gap.)

Note that in the later sections of this chapter we refer to gaps or intervals $[s,t]$ instead as parametric points or pairs (s,t), in which case feasibility is defined analogously.

5.2.2 Free Space

To compute the standard Fréchet distance one normally looks at the so called *free space*. The t free space between curves π and σ, with n and m edges respectively, is defined as

$$F_t = \{(\alpha,\beta) \in [0,n]\times[0,m] \;\mid\; ||\pi(\alpha) - \sigma(\beta)|| \leq t\}.$$

Similarly define $F_t^< = \{(\alpha,\beta) \in [0,n]\times[0,m] \;\mid\; ||\pi(\alpha) - \sigma(\beta)|| < t\}$ to be F_t without its boundary. $C(i,j) = [i,i+1]\times[j,j+1]$ is referred to as the cell of the free space diagram determined by edges $\pi_i\pi_{i+1}$ and $\sigma_j\sigma_{j+1}$, and the free space within this cell is

$$F_t(i,j) = \{(\alpha,\beta) \in [i,i+1]\times[j,j+1] \;\mid\; ||\pi(\alpha) - \sigma(\beta)|| \leq t\}.$$

Alt and Godau [11] showed that the free space within a cell is always a convex set (specifically, the clipping of an affine transformation of a disk to the cell). Moreover,

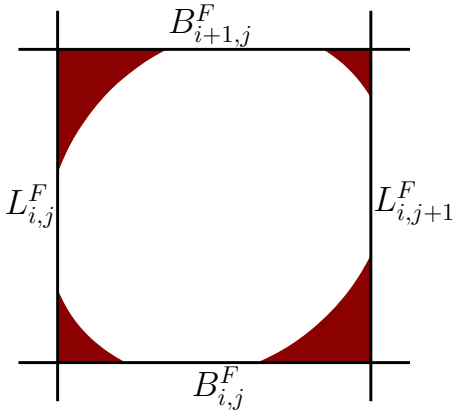


Figure 5.2: Free space cell

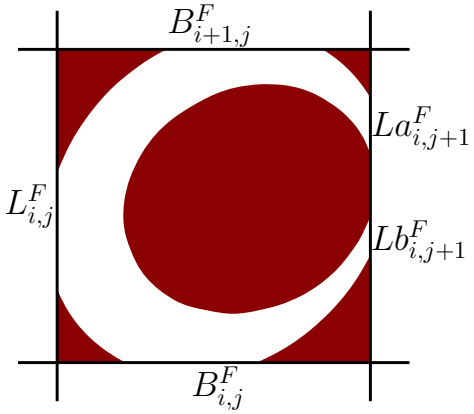


Figure 5.3: Relative free space cell

any x, y monotone path in the free space from $(0,0)$ to (n,m) corresponds to a pair of reparameterizations f, g of π, σ such that $width_{f,g}(\pi,\sigma) \leq t$. The converse also holds and hence $d_{\mathcal{F}}(\pi,\sigma) \leq t$ if and only if such a monotone path exists. These two statements together imply that in order to determine if $d_{\mathcal{F}}(\pi,\sigma) \leq t$, it suffices to restrict attention to the free space intervals on the boundaries of the cells. Specifically, let $L^F_{i,j}$ (resp. $B^F_{i,j}$) denote the left (resp. bottom) free space interval of $C(i,j)$, i.e. $L^F_{i,j} = F_t(i,j) \cap (\{i\} \times [j, j+1])$ (resp. $B^F_{i,j} = F_t(i,j) \cap ([i, i+1] \times \{j\})$). See Figure 5.2.

5.2.3 Relative Free Space

We extend the standard free space definitions of the previous section to the Fréchet gap distance problem. First we define the s,t relative free space between π and σ to be

$$F_{[s,t]} = \{(\alpha,\beta) \in [0,n] \times [0,m] || \ \mid \ s \leq ||\pi(\alpha) - \sigma(\beta)|| \leq t\} = F_t \setminus F_s^{<},$$

describing all pairs of points, one on π and one on σ, whose distance is contained in $[s,t]$. For a point (α,β) in a cell of $F_{[s,t]}$ or F_t, throughout we use the colloquial terms higher or lower (resp. right or left) to refer larger or smaller values of α (resp. β).

Again we seek an x, y monotone path in the relative free space from $(0,0)$ to (n,m), since such a path corresponds to a pair of reparameterizations f, g of π, σ such that

$gapwidth_{f,g}(\pi,\sigma) \leq t-s$, and hence $d_{\mathcal{G}}(\pi,\sigma) \leq t-s$. Conversely, if no such path exists then $[s,t]$ is not a feasible gap for π and σ, implying that $[s^*,t^*] \not\subseteq [s,t]$, but note however that unlike the standard Fréchet distance, it may still hold that $t^*-s^* \leq t-s$.

The relative free space in the cell $C(i,j)$ determined by edges $\pi_i\pi_{i+1}$ and $\sigma_j\sigma_{j+1}$ is,

$$F_{[s,t]}(i,j) = \{(\alpha,\beta) \in [i,i+1]\times[j,j+1] \;\mid\; s \leq ||\pi(\alpha)-\sigma(\beta)|| \leq t\} = F_t(i,j) \setminus F_s^{<}(i,j).$$

Another technical challenge with the Fréchet gap problem arises from the fact that the relative free space in a cell may not be convex (see Figure 5.3). However, there is some structure. Observe that $F_{[s,t]}(i,j) = F_t(i,j) \setminus F_s^{<}(i,j)$, and hence is the set difference of two convex sets, where one is contained in the other. In other words, it looks like a standard free space cell with a hole removed. In particular, we can again look at the free space intervals on the cell boundaries. As $F_t(i,j)$ is convex, it still determines a single interval on each cell boundary, however, this interval may be broken into two subintervals by the removal of $F_s(i,j)$ (whose convexity implies it is at most two subintervals). Let $L^F_{i,j} = Lb^F_{i,j} \cup La^F_{i,j}$ denote the relative free space on the left boundary of $C(i,j)$, where $Lb^F_{i,j}$ denotes the bottom and $La^F_{i,j}$ the top interval (note if $F_s(i,j)$ does not intersect the boundary then $Lb^F_{i,j} = La^F_{i,j} = L^F_{i,j}$). Similarly, let $B^F_{i,j} = Bl^F_{i,j} \cup Br^F_{i,j}$ denote the relative free space on the bottom boundary of $C(i,j)$, where $Bl^F_{i,j}$ denotes the left and $Br^F_{i,j}$ the right interval.

5.3 The Fréchet Gap Decision Problem

The Fréchet gap decision problem is defined as follows.

Problem 5.3.1. *Given polygonal curves π,σ, is a given interval $[s,t]$ a feasible gap for π,σ?*

As discussed in Section 5.2.3, $[s,t]$ is a feasible gap for π and σ if and only if there exists an x,y monotone path from $(0,0)$ to (n,m) in the $[s,t]$ relative free space $F_{[s,t]}$. This motivates the definition of the reachable relative free space,

$$RF_{[s,t]} = \{(\alpha,\beta) \in [0,n]\times[0,m] \;\mid\; \text{there exists an } x,y \text{ monotone path from} (0,0) \text{ to } (\alpha,\beta)\}.$$

Hence the answer to Problem 5.3.1 is 'yes' if and only if $(n, m) \in RF_{[s,t]}$. As was the case with the relative free space, the relevant information for the reachable relative free space is contained on the cell boundaries. We now describe how to propagate the reachable information from the left and bottom boundary to the right and top boundary of a cell, which ultimately will allow us to propagate the reachable information from $(0, 0)$ to (n, m). (Note this is the typical approach to solving the standard Fréchet distance decision problem.)

Let $L^R_{i,j}$ and $B^R_{i,j}$ denote the reachable subsets of the left and bottom boundaries of $C(i, j)$. First we argue that like $L^F_{i,j}$, $L^R_{i,j}$ is composed of at most two disjoint intervals. Let $Lx^F_{i,j}$ be either $La^F_{i,j}$ or $Lb^F_{i,j}$. The reachable subset of $Lx^F_{i,j}$ is a single connected interval. To see this, observe that wherever the lowest reachable point in $Lx^F_{i,j}$ lies, all points above it in $Lx^F_{i,j}$ are reachable by a monotone path. As $L^R_{i,j}$ is a subset of $L^F_{i,j}$, this implies it is composed of at most two intervals denoted $La^R_{i,j}$ and $Lb^R_{i,j}$ (if $L^F_{i,j}$ is single interval then $L^R_{i,j} = La^R_{i,j} = Lb^R_{i,j}$). $Bl^R_{i,j}$ and $Br^R_{i,j}$ are defined similarly.

Propagating in a cell: Given $L^R_{i,j}$ and $B^R_{i,j}$, we now describe how to compute $L^R_{i,j+1}$ ($B^R_{i+1,j}$ is handled similarly). There are four cases, determined by whether we are propagating $L^R_{i,j}$ or $B^R_{i,j}$, and whether we are going above or below the hole $F_s(i, j)$. First, some notation.

Definition 5.3.2. *Label the leftmost and rightmost vertical lines tangent to the hole $F_s(i, j)$ as $v\ell^l_{i,j}$ and $v\ell^r_{i,j}$, and label the topmost and bottommost horizontal tangent lines as $h\ell^a_{i,j}$ and $h\ell^b_{i,j}$ (see Figure 5.4). Similarly define the leftmost point $\mathcal{H}^l_{i,j}$, the rightmost point $\mathcal{H}^r_{i,j}$, the topmost point $\mathcal{H}^a_{i,j}$, and the bottommost point $\mathcal{H}^b_{i,j}$, of $F_s(i, j)$ (Note any one of these points may be undefined if $F_s(i, j)$ intersects the boundary in more than a single point, as is the case for $\mathcal{H}^r_{i,j}$ in Figure 5.4.) Finally, let $\mathcal{I}^a_{i,j}$ be the highest and $\mathcal{I}^b_{i,j}$ the lowest point of $L^F_{i,j+1}$. When i, j is fixed, the subscript is often dropped.*

The above notation will be used throughout this chapter as it defines the relevant extent measures of the relative free space. Here we also define the point w_a to be the intersection

point of $La^F_{i,j+1}$ with $h\ell^a$, or more generally if they do no intersect w_a is the lowest point of $La^F_{i,j+1}$ that is above $h\ell^a$. (Note w_a is the lowest reachable point when passing over the hole $F_s(i,j)$, and may possibly be undefined.) For the four cases below we consider four points p_r, p_l, p_b, and p_a. We assume these points are defined, though they may not be depending on the structure of $L^R_{i,j}$ and $B^R_{i,j}$, in which case there is nothing to propagate.

1) Propagating $B^R_{i,j}$

   a) Below $F_s(i,j)$: Let p_r be the rightmost point of $Br^R_{i,j}$ (note we may have $Br^F_{i,j} = B^F_{i,j}$). In this case there is a monotone path along the boundary of $F_t(i,j)$ from p_r to $\mathcal{I}^b$, and hence all of $Lb^F_{i,j+1}$ is reachable, i.e. $Lb^R_{i,j+1} = Lb^F_{i,j+1}$. See green path in Figure 5.5.

   b) Above $F_s(i,j)$: Let p_l be the intersection point of $Bl^R_{i,j}$ and the line $v\ell^l$, and let w_a be as described above. If either p_l or w_a is undefined there is nothing to propagate. Otherwise there is a monotone path from p_l to w_a, specifically follow the line of $v\ell^l$ from p_l to $\mathcal{H}^l$, then continue along the boundary of $F_s(i,j)$ to $\mathcal{H}^a$, and then follow $h\ell^a$ to w_a. Hence all the points in $La^F_{i,j+1}$ that are at least as high as w_a are reachable from p_l, see the blue path in Figure 5.5.

2) Propagating $L^R_{i,j}$

   a) Below $F_s(i,j)$: Let p_b be the lowest point of $Lb^R_{i,j}$ (note we may have $Lb^F_{i,j} = L^R_{i,j}$). If p_b lies above $h\ell^b$, then there is nothing to propagate. Otherwise, the reachable points on $Lb^F_{i,j+1}$ coming from monotone paths from p_b (that pass below $F_s(i,j)$) can be found by walking as low as possible through the cell. Specifically, if there is a point $Lb^F_{i,j+1}$ at the same height as p_b then we can walk horizontally directly to it, otherwise when we walk horizontally we bump into the boundary of $F_t(i,j)$ and follow it up to $\mathcal{I}^b$ (green path in Figure 5.6). In either case all higher points on $Lb^F_{i,j+1}$ are reachable.

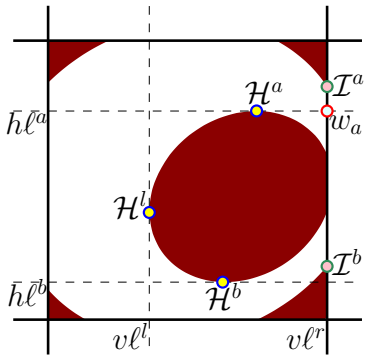


Figure 5.4: Free space cell

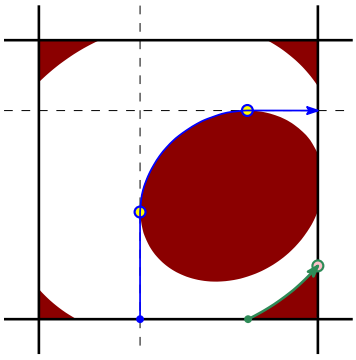


Figure 5.5: $B^R_{i,j}$ to $L^R_{i,j+1}$

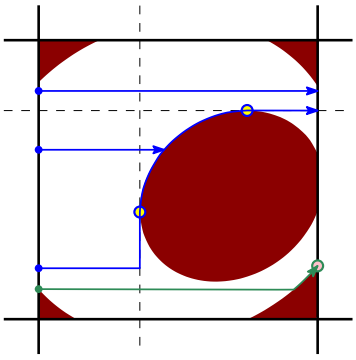


Figure 5.6: $L^R_{i,j}$ to $L^R_{i,j+1}$

b) Above $F_s(i,j)$: Let p_a be the lowest point of $La^R_{i,j}$. If p_a lies above $h\ell^a$ then by walking horizontally to the right boundary of the cell (top blue path in Figure 5.6), we can reach all points of $La^F_{i,j+1}$ that are at least as high as p_a (note there may be no such points). Otherwise, there is a monotone path from p_a to w_a (if w_a is defined). There are two cases based on the relative heights of p_a and the point $\mathcal{H}^l$. If p_a lies below $\mathcal{H}^l$, then the monotone path walks horizontally from p_a to $v\ell^l$, then vertically on $v\ell^l$ to $\mathcal{H}^l$, then continues along the boundary of $F_s(i,j)$ to $\mathcal{H}^a$, and then horizontally to w_a. If p_a lies above $\mathcal{H}^l$, then the monotone path walks horizontally from p_a to the boundary of $F_s(i,j)$, then continues along the boundary of $F_s(i,j)$ to $\mathcal{H}^a$, and then horizontally to w_a. In either case all the points in $La^F_{i,j+1}$ that are at least as high as w_a are reachable.

Theorem 5.3.3. *Given polygonal curves π of length n, σ of length m, and an interval $[s,t]$, the Fréchet gap decision problem, Problem 5.3.1, can be solved in $O(nm)$ time.*

Proof. First compute $L^F_{i,j}$ and $B^F_{i,j}$ for all $1 \le i \le n$, $1 \le j \le m$. Next initialize the reachable subset of left boundary of the entire relative free space diagram, i.e. $\cup_i L^R_{i,1}$. To do so, consider the entire free space of the left boundary, $\cup_i L^F_{i,1}$, and mark all points in this set that are reachable (by paths restricted to $\cup_i L^F_{i,1}$) from $(0,0)$. Note if $(0,0)$ is not in the free space, we return 'no' as the answer to the decision problem. Similarly initialize the reachable subset of

bottom boundary of the entire relative free space diagram. Now propagate the reachable sets using any topological ordering of the cells (e.g. go in increasing column order, and for each column go by increasing row order). Specifically, for each cell $C(i,j)$ we use $L^R_{i,j}$ and $B^R_{i,j}$ (and $L^F_{i,j+1}$ and $B^F_{i+1,j}$) to compute $L^R_{i,j+1}$ and $B^R_{i+1,j}$, as described above. We then return 'yes' if (n,m) is in the reachable space and 'no' otherwise.

As for the running time, $L^F_{i,j}$ and $B^F_{i,j}$ take $O(1)$ time to compute per cell, and there are $O(nm)$ cells. Initializing the reachable sets then takes $O(n+m)$ time. As argued above, for any i,j, $L^R_{i,j}$ and $B^R_{i,j}$ are composed of at most two disjoint intervals hence propagating the reachable information to $L^R_{i,j+1}$ and $B^R_{i+1,j}$ takes $O(1)$ time per cell, and again there are $O(nm)$ cells, so this is the total running time. □

5.4 Finding the Relative Free Space Critical Events

In this section we describe the relative free space critical events, that is a polynomially sized subset of possible intervals, which must contain an optimal interval $[s^*,t^*]$. The relative free space events are significantly more complicated than the free space events for the standard Fréchet distance. The following definitions will be used throughout this section.

Definition 5.4.1. *Two free space cells $C(i,j)$ and $C(k,l)$ are* adjacent *if they share a horizontal or vertical boundary, i.e. $k=i$ and $|l-j|=1$, or $l=j$ and $|k-i|=1$. Call any monotone path from $(0,0)$ to (n,m) in the relative free space a* valid *path. Given any valid path p, the* cell sequence *of p, denoted $\mathsf{cp}(p)=(C_1,\ldots,C_{n+m-1})$, is the ordered sequence of cells p intersects (so $C_1=C(0,0)$, $C_{n+m-1}=C(n-1,m-1)$).*

We now define a number of other sequences determined by p. Let the entry point e_i be the point where p first intersects the cell C_i, and define the entry sequence of p as $\mathsf{entry}(p)=(e_1,\ldots,e_{n+m})$, where $e_1=(0,0)$ and e_{n+m} is defined as (n,m). Let $\mathsf{int}(p)=(I_2,I_3,\ldots,I_{n+m-1})$ denote the sequence of boundary free space intervals passed by p, i.e. e_i

lies on I_i. For horizontally adjacent cells $C(i,j)$ and $C(i,j+1)$ in the cell sequence, p either passes above or below $F_s(i,j)$, specifically if p intersects the vertical segment connecting $\mathcal{H}^a$ to the top boundary of $C(i,j)$ then p passes above *$F_s(i,j)$, and otherwise p* passes below. *(Similarly define passing left or right for vertically adjacent cells.) This defines the* passing sequence *of p, denoted $\mathsf{pass}(p) = (h_1, \ldots, h_{n+m-1})$, where $h_i \in \{$above, below, left, right$\}$.*

For the standard Fréchet distance, Alt and Godau [11] specified the following set of distance values, called the critical events, which must contain the optimal Fréchet distance.

- *Initialization* event: The minimum value ε such that $(0,0) \in F_\varepsilon$ and $(n,m) \in F_\varepsilon$.
- *Connectivity* events: For any cell C_i, the minimum ε such that L_i^F or B_i^F is non-empty, corresponding to the distance between a vertex of one curve and an edge of the other.
- *Monotonicity* events: Let I_j and I_k be two non-empty vertical free space boundary intervals in the same row with I_j left of I_k (or horizontal intervals in the same column). The minimum ε such that $\mathcal{I}_j^b \leq \mathcal{I}_k^a$, that is there is a monotone path between I_j and I_k.

Since any valid path can be decomposed into a set of row and column subpaths, proving that $d_{\mathcal{F}}(\pi, \sigma)$ is one the above defined critical events is a straightforward task.

For the Fréchet gap distance, the critical events will be a super-set of the standard Fréchet events. As an optimal gap is defined by an interval $[s,t]$, the events below can either be a value of s or a value of t. A *critical interval* is then any valid $s \leq t$ pair from the first three critical event types defined below. Additionally, there is now a fourth type called a floating monotonicity event. These events directly specify the s,t pair (i.e. these "events" are also "critical intervals"), and there are potentially an infinite number of such events.

1) *Initialization* events: The values $s = \min\{||\pi_0 - \sigma_0||, ||\pi_n - \sigma_m||\}$ and $t = \max\{||\pi_0 - \sigma_0||, ||\pi_n - \sigma_m||\}$. That is, the supremum of values for s such that $(0,0) \notin F_s$ and $(n,m) \notin F_s$, and the minimum value of t such that $(0,0) \in F_t$ and $(n,m) \in F_t$.
2) *Connectivity* events: For any row i and column j, the values $dist(\pi_i, \sigma_j\sigma_{j+1})$, $dist(\pi_{i+1}, \sigma_j\sigma_{j+1})$, $dist(\pi_i\pi_{i+1}, \sigma_j), dist(\pi_i\pi_{i+1}, \sigma_{j+1})$, for either s or t. In other words for cell $C_{i,j}$,

the maximum value s such that $\mathcal{H}^a$, $\mathcal{H}^b$, $\mathcal{H}^l$, or $\mathcal{H}^r$ are defined, or minimum value t such that $\mathcal{I}^a, \mathcal{I}^b$ (or similarly any of the other three cell boundary intervals) are defined. Note $\mathcal{I}^a, \mathcal{I}^b$ are first defined at the same location/value where $\mathcal{H}^r$ is last defined, yet we still regard these as separate events, one for s and the other for t. (For s this is when the free space intervals may break into two, and for t it is when the interval is first non-empty.)

3) *Standard Monotonicity* events: For any cells C_j, C_k in the same row with C_j left of C_k:
   (a) The value t such that $height(\mathcal{I}_j^b) = height(\mathcal{I}_k^a)$.
   (b) The value s such that $height(\mathcal{H}_j^a) = height(\mathcal{H}_k^b)$.
4) *Floating Monotonicity* events: For any cells C_j, C_k in the same row with C_j left of C_k:
   (a) Any pair s, t such that $height(\mathcal{I}_j^b) = height(\mathcal{H}_k^b)$.
   (b) Any pair s, t such that $height(\mathcal{H}_j^a) = height(\mathcal{I}_k^a)$.

Here $height()$ denotes the vertical coordinate of a point in the relative free space. Analogous definitions apply to the case when cells are in the same column. Note that depending on the geometry such events may not be defined.

Let S_s and S_t denote the set of values for s and t, respectively, determined by the initialization, connectivity and standard monotonicity critical events, and let $S_s \times S_t$ denote the corresponding set of valid critical intervals determined by these values. Let S_F be the set of s, t intervals determined by floating monotonicity events. The set of all critical intervals is then $S_I = S_F \cup (S_s \times S_t)$.

Lemma 5.4.2. *S_I contains any optimal Fréchet gap interval $[s^*, t^*]$.*

Proof. For the sake of contradiction, suppose $[s^*, t^*] \notin S_I$. As $[s^*, t^*]$ is a feasible Fréchet gap, there must exist a valid path p in $F_{[s^*,t^*]}$. Let the cell, passing, entry, and interval sequences of p be $\mathsf{cp}(p) = (C_1, \ldots, C_{n+m-1})$, $\mathsf{pass}(p) = (h_1, \ldots, h_{n+m-1})$, $\mathsf{entry}(p) = (e_1, \ldots, e_{n+m})$, and $\mathsf{int}(p) = (I_2, I_3, \ldots, I_{n+m-1})$ (see Definition 5.4.1). In this proof we will show that if $[s^*, t^*] \notin S_I$ then the canonical form of p which, subject to having the same cell and passing

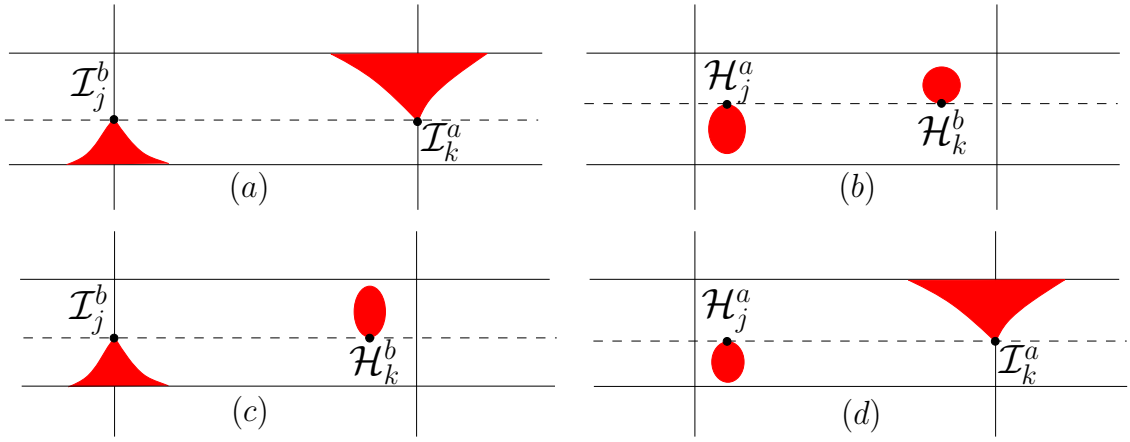


Figure 5.7: Opening of a horizontal passage.

sequences as p, locally remains as low and left as possible (i.e. follows the reachable free space propagation rules of Section 5.3), will also define a valid path in the free space after either increasing s^* or decreasing t^*.

Let $S_F^* = \{t \mid (s^*, t) \in S_F\}$ (which may possibly be empty). Since $[s^*, t^*] \notin S_I$ it must be that $t^* \notin S_F^*$. Also since $[s^*, t^*] \notin S_I$, either $s^* \notin S_s$ or $t^* \notin S_t$, and we will assume it is the $t^* \notin S_t$ case (the $s^* \notin S_s$ case is argued similarly). Let $t_{init} = \max\{||\pi_0 - \sigma_0||, ||\pi_n - \sigma_m||\}$ be the corresponding t value of the initialization event, and hence $t_{init} \in S_t$. Note that because $[s^*, t^*]$ is feasible, $t_{init} \in [s^*, t^*]$ (as any valid path contains (π_0, σ_0) and (π_n, σ_m)). Since we assumed $t^* \notin S_t$, this implies $t_{init} \in [s^*, t^*)$, and so $[s^*, t^*) \cap S_t \neq \emptyset$ (moreover, $s^* \neq t^*$). So let $T = S_t \cup S_F^*$, and let t^- be the largest value in T which is $\leq t^*$, which we just argued must exist and $t^- \in [s^*, t^*)$ (in particular we later use that $(t^-, t^*] \cap T = \emptyset$).

We now show there is a valid path in the free space $F_{[s^*, t^-]}$, which is a contradiction as $t^- < t^*$, but $[s^*, t^*]$ was optimal. Specifically, we argue there is a valid canonical path p' in $F_{[s^*, t^-]}$ with the same cell and passing sequences as p. To this end, define $\mathsf{entry}(p') = (e'_1, \ldots, e'_{n+m})$ such that (i) $e'_1 = e_1 = (0, 0)$, (ii) $e'_{n+m} = e_{n+m} = (n, m)$, and (iii) for $1 < i < n + m$, if I_i is vertical (resp. horizontal) then e'_i is the lowest (resp. leftmost) point in $F_{[s^*, t^-]} \cap I_i$ that is above (resp. right of) e'_{i-1}, and also above $\mathcal{H}^a_{i-1}$, (resp. right of $\mathcal{H}^r_{i-1}$) if h_{i-1} equals *above*

(resp. *right*). We now argue that the points in this entry sequence are well defined, and hence p' is a valid path through $F_{[s^*,t^-]}$.

First observe that as $(t^-, t^*] \cap T = \emptyset$, initialization events cannot lie in this interval, and so e'_1 and e'_{n+m} must be in $F_{[s^*,t^-]}$. Now we inductively argue that for all other $1 < i < n+m$, e'_i is well defined. Again since $(t^-, t^*] \cap T = \emptyset$, $F_{[s^*,t^-]} \cap I_i$ is not empty for all $1 < i < n+m$. So fix an index β, and assume $e'_{\beta-1}$ is well defined. Without loss of generality assume I_β is a vertical edge (the horizontal case is handled similarly). If $I_{\beta-1}$ is on a horizontal edge then clearly, if $h_{\beta-1}$ equals *below*, $e'_\beta = \mathcal{I}^b_{\beta-1}$, which is well defined as s^* is fixed and $(t^-, t^*] \cap T = \emptyset$. If $h_{\beta-1}$ equals *above*, then in $F_{[s^*,t^-]}$ it must be that $e'_{\beta-1}$ is on the left of $\mathcal{H}^l_{\beta-1}$ and $height(\mathcal{H}^a_{\beta-1}) \leq height(\mathcal{I}^a_{\beta-1})$, as this was true in $F_{[s^*,t^*]}$ and we assumed there were no monotonicity events in $(t^-, t^*] \cap T$. In other words we can pass to the left and above the hole, and thus e'_β is well defined, Also note in this case that either $height(e'_\beta) = height(\mathcal{H}^a_{\beta-1})$ or $height(e'_\beta) = height(\mathcal{I}^b_{\beta-1})$.

Now suppose $I_{\beta-1}$ is a vertical edge and moreover, let $I_\alpha, I_{\alpha+1}, \ldots, I_\beta$ be the earlier boundary intervals in this row, i.e. it is the maximal length contiguous subsequence of vertical boundary intervals ending at I_β. As $e'_{\beta-1}$ is well defined, we must have either $height(e'_{\beta-1}) = height(\mathcal{H}^a_j)$ for $j < \beta' - 1$ or $height(e'_{\beta-1}) = height(\mathcal{I}^b_j)$ for $j \leq \beta - 1$, as p' locally remained as low as possible (i.e. there must be a free space object responsible for pushing $e'_{\beta-1}$ to that height). Therefore if the point e'_β is not well defined then it must be that either $height(\mathcal{I}^a_{\beta-1}) < height(\mathcal{H}^a_j)$ or $height(\mathcal{I}^a_{\beta-1}) < height(\mathcal{I}^b_j)$ for $j \leq \beta - 1$. The latter case is not possible, as it implies there must have been a standard monotonicity event in $(t^-, t^*]$ (as clearly for $F_{[s^*,t^*]}$ there was a monotone path and so $height(\mathcal{I}^a_{\beta-1}) \geq height(\mathcal{I}^b_j)$), but we assumed $(t^-, t^*] \cap S_t = \emptyset$. For the former case, observe that $height(\mathcal{H}^a_j)$ is unchanged from $F_{[s^*,t^*]}$ to $F_{[s^*,t^-]}$, and so it would imply there was a floating monotonicity event in $(t^-, t^*] \cap S^*_F$, but we assumed this intersection was empty. □

5.4.1 Bounding the number of critical intervals

We now bound the number of critical intervals, i.e. $|S_I|$. An interval $[s,t] \in S_I$, is either in $S_s \times S_t$ or S_F. Now S_s (resp. S_t) has size[1] $O(n^3)$ as it contains one initialization event, $O(n^2)$ connectivity events, and $O(n^3)$ monotonicity events (just like the standard Fréchet case). As we consider all valid pairs from S_s and S_t, this gives an $O(n^6)$ bound on $|S_s \times S_t|$.

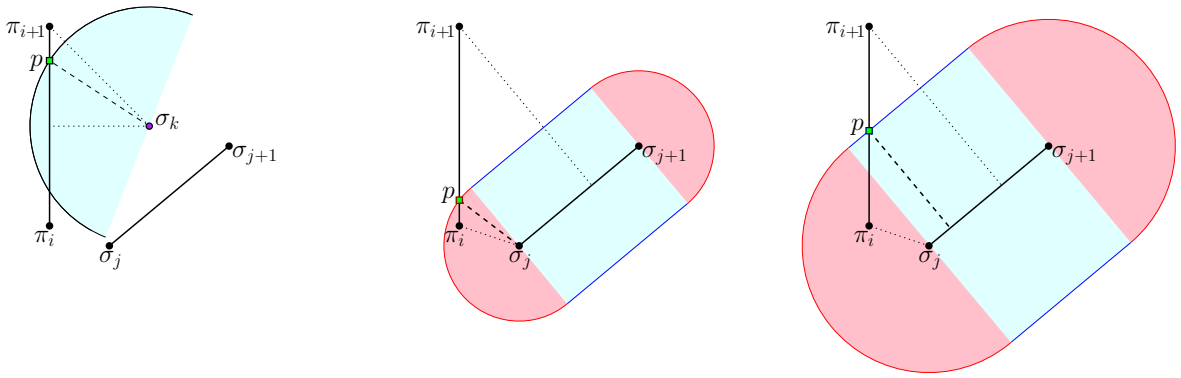
Bounding the size of S_F is significantly more complicated. In particular, the floating monotonicity events may give rise to an entire continuum of critical intervals. For example, consider the second type of floating monotonicity event (4b), shown in Figure 5.7. The value of $height(\mathcal{H}_j^a)$ is governed only by a function of s and the value of $height(\mathcal{I}_k^a)$ only by a function of t. These functions might be such that if we increase or decrease s, but keep $t-s$ constant (i.e. the gap value we are optimizing), $height(\mathcal{H}_j^a) = height(\mathcal{I}_k^a)$ remains an invariant. (Hence the term "floating" events.)

In this section we describe the functions which govern how s and t can vary such that $height(\mathcal{H}_j^a) = height(\mathcal{I}_k^a)$ remains an invariant. Ultimately our understanding of these function will yield a polynomially sized set of canonical critical intervals (determined by vertices of the arrangement of these functions), which must contain an optimum gap interval.

Function Description of Floating Monotonicity Events

Consider the floating monotonicity event type (4b) (similar statements will hold for type (4a)). Such an event is specified by a triple of indices, i,j,k, where i specifies an edge $\pi_i\pi_{i+1}$ (i.e. a row of the relative free space), j specifies an edge $\sigma_j\sigma_{j+1}$ (i.e. a column), and $k \geq j$ specifies a vertex σ_k (i.e. the right boundary of a column). The event occurs when $height(\mathcal{H}_j^a) = height(\mathcal{I}_k^a) = h$.

[1]For simplicity, from this point onwards we assume without loss of generality that $m \leq n$ and only write sizes and running times with respect to n.



(a) Distance from σ_k (b) Distance from σ_j (c) Distance from $\sigma_j\sigma_{j+1}$

Figure 5.8: How point p determines s and t. In general segments may not lie in a single plane.

Geometrically, a fixed height h corresponds to a point p on $\pi_i\pi_{i+1}$. The point $\mathcal{H}_j^a$ is determined by s, and $\mathcal{I}_k^a$ by t. First lets understand $\mathcal{I}_k^a$. In order to have $h = height(\mathcal{I}_k^a)$, t must be such that $t = ||\sigma_k - p||$, and moreover p must be the higher (i.e. closer to π_{i+1}) of the possibly two points on $\pi_i\pi_{i+1}$ satisfying this condition (the other point determining $\mathcal{I}_k^b$). Consider the plane determine by π_i, π_{i+1}, and σ_k, and let $\pi_i = (0, 0)$, $p = (0, h)$, and $\sigma_k = (\chi, \gamma)$ (see Figure 5.8a). Then as a function of h, t is described by the equation $t = \sqrt{\chi^2 + (\gamma - h)^2}$. Note that $\mathcal{I}_k^a$ is only defined when $t \in [t_1, t_2]$, where $t_1 = dist(\sigma_k, \pi_i\pi_{i+1})$ and $t_2 = ||\sigma_k - \pi_{i+1}||$, and hence this equation is only relevant in this interval.

$height(\mathcal{H}_j^a)$ on the other hand is determined by s, however the relationship is a bit more complicated. Observe that $\mathcal{H}_j^a$ is the only point on the horizontal line $h\ell_j^a$ that is in the set $F_s(i, j)$, meaning the point on $\sigma_j\sigma_{j+1}$ that $\mathcal{H}_j^a$ corresponds to must be the closest point on $\sigma_j\sigma_{j+1}$ to p (see Figure 5.8b and Figure 5.8c). If this closest point is either σ_j or σ_{j+1}, then the form of the equation for s in terms of h is the same as it was t, namely $s = \sqrt{\alpha^2 + (\beta - h)^2}$ (where α, β are now the coordinates of either σ_j or σ_{j+1}). Otherwise this closest point is in the interior of $\sigma_j\sigma_{j+1}$ in which case the equation is of the form $s = c \cdot h + d$, for some constants c and d (since as one walks along a line, the distance to another fixed line is given by a linear

equation). Similar to $\mathcal{I}_k^a$, $\mathcal{H}_j^a$ is only defined when $s \in [s_1, s_2]$, where $s_1 = dist(\sigma_j\sigma_{j+1}, \pi_i\pi_{i+1})$ and $s_2 = dist(\sigma_j\sigma_{j+1}, \pi_{i+1})$, and hence this equation is only relevant in this interval.

Now that we have a description of $height(\mathcal{H}_j^a)$ in terms of s and $height(\mathcal{I}_k^a)$ in terms of t, we can describe the function for t in terms of s, denoted $f_{i,j,k}(s)$, which describes when $height(\mathcal{H}_j^a) = height(\mathcal{I}_k^a) = h$. There are two cases based on the form of the function describing s.

Interior of $\sigma_j\sigma_{j+1}$ case:

$$s = c \cdot h + d, \quad t = \sqrt{\chi^2 + (\gamma - h)^2} \quad \Rightarrow \quad f_{i,j,k}(s) = \sqrt{\left(\frac{s-d}{c} - \gamma\right)^2 + \chi^2}$$

Endpoint of $\sigma_j\sigma_{j+1}$ case:

$$s = \sqrt{\alpha^2 + (\beta - h)^2}, \quad t = \sqrt{\chi^2 + (\gamma - h)^2} \quad \Rightarrow \quad f_{i,j,k}(s) = \sqrt{(\sqrt{s^2 - \alpha^2} + (\beta - \gamma))^2 + \chi^2}$$

To summarize, $f_{i,j,k}(s)$ is composed of at most three hyperbola[2] pieces, and is only (possibly) defined within the region $s \in [s_1, s_2]$ and $t \in [t_1, t_2]$, see Figure 5.9. Also, the geometry of the problem implies that when $f_{i,j,k}(s)$ is defined it is a monotone increasing function. Hence the intersection of $f_{i,j,k}$ with the bounding box $[s_1, s_2] \times [t_1, t_2]$ is connected, and so rather than using this box to define $f_{i,j,k}$, we instead say $f_{i,j,k}$ is either completely undefined or is defined only in the interval $[s^l, s^r]$ where s^l and s^r are the s coordinate where $f_{i,j,k}$ respectively enters and leaves the bounding box. Note that one can argue if $f_{i,j,k}$ is defined then $s^r = s_2$, however, it may be that $s^l > s_1$ (if the closest point to $\sigma_j\sigma_{j+1}$ is lower on $\pi_i\pi_{i+1}$ than the closest point to σ_k).

The exact form of the equation $f_{i,j,k}(s)$ is not needed in our analysis, however, the above discussion implies the following simple observation which will be used later.

Observation 5.4.3. *In the s, t parametric space $f_{i,j,k}$ is either undefined or defines a constant complexity monotonically increasing curve piece, with endpoints at values $s^l_{i,j,k} \leq s^r_{i,j,k}$. In*

[2]Technically, the endpoint case is not a hyperbola, though it is similar.

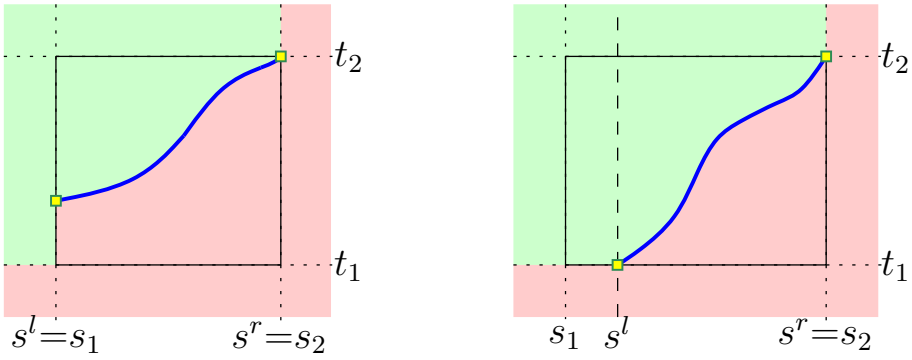


Figure 5.9: Two cases for curve piece $f_{i,j,k}$, and shaded satisfying points in s, t parametric space.

particular, $f_{i,j,k}$ has only a constant number of local minima and maxima (i.e. points of tangency) with respect to translations of the line $t = s$.

Note that for (4a), i.e. when $height(\mathcal{I}_j^b) = height(\mathcal{H}_k^b)$, $f_{i,j,k}$ can be defined similarly, and the above observation again holds. One must also define functions for the analogous events in the free space columns. Such functions are again determined by triples i, j, k, however now i, j refer to rows and k to the column. Below we will denote these functions by $g_{i,j,k}$.

Events minimizing the gap

As discussed above each $f_{i,j,k}$, if defined, gives an entire continuum of critical intervals. However, ultimately we are only interested in feasible intervals which minimize the gap, and this will allow us to reduce this continuum to a polynomial number of canonical intervals. This polynomially sized set is determined not only by the $f_{i,j,k}$, but also by the other types of critical events. Note that initialization (1), connectivity (2), and standard monotonicity events (3) only define constraints on either just s or t, whereas the $f_{i,j,k}$ and $g_{i,j,k}$ define a continuum of $[s, t]$ intervals. Hence to put them on equal footing we think of all of them as defining constraints in the two dimensional s, t parametric space.

First observe that in the parametric space, for any point (s, t) of interest, $0 \leq s \leq t$, and so we only consider points in the first quadrant that are above the line $t = s$. Initialization,

connectivity, and standard monotonicity events are simply defined by horizontal or vertical lines. Specifically, for each such event the points satisfying the corresponding constraint are those above (resp. left of) the corresponding horizontal (resp. vertical) line:

1) Initialization events: $s \leq \alpha_0, \quad t \geq \beta_0$

   Where $\alpha_0 = \min\{||\pi_0 - \sigma_0||, ||\pi_n - \sigma_m||\}$ and $\beta_0 = \max\{||\pi_0 - \sigma_0||, ||\pi_n - \sigma_m||\}$.

2) Connectivity events: $s \leq \alpha^l_{i,j}$ or $s \leq \alpha^b_{i,j}, \quad t \geq \beta^l_{i,j}$ or $t \geq \beta^b_{i,j}$

   Where the $\alpha_{i,j}$ and $\beta_{i,j}$ are *vertex-edge* distances, that is $\alpha^l_{i,j} = \beta^l_{i,j} = dist(\pi_i\pi_{i+1}, \sigma_j)$ or $\alpha^b_{i,j} = \beta^b_{i,j} = dist(\pi_i, \sigma_j\sigma_{j+1})$. Note defining both $\alpha_{i,j}$ and $\beta_{i,j}$ is not necessary but useful to distinguish constraints on s from those on t.

3) Standard Monotonicity events: $s \leq \alpha_{i,(j,k)}$ or $s \leq \alpha_{(i,j),k}, \quad t \geq \beta_{i,(j,k)}$ or $t \geq \beta_{(i,j),k}$

   Which happens when the free space is such that $\alpha_{i,(j,k)} = height(\mathcal{H}^a_{i,j}) = height(\mathcal{H}^b_{i,k})$ or $\alpha_{(i,j),k} = height(\mathcal{H}^a_{i,k}) = height(\mathcal{H}^b_{j,k})$, and when $\beta_{i,(j,k)} = height(\mathcal{I}^b_{i,j}) = height(\mathcal{I}^a_{i,k})$ or $\beta_{(i,j),k} = height(\mathcal{I}^b_{i,k}) = height(\mathcal{I}^a_{j,k})$.

4) Floating Monotonicity events: $t \geq f_{i,j,k}(s)$ for $s \in [s^{lf}_{i,j,k}, s^{rf}_{i,j,k}]$, or $t \geq g_{i,j,k}(s)$ for $s \in [s^{lg}_{i,j,k}, s^{rg}_{i,j,k}]$. Note depending on the geometry such constraints may not be defined.

Note that the first three event types each partition the entire parametric space into two connected sets, those which either satisfy or do not satisfy the constraint. The $f_{i,j,k}$ (and $g_{i,j,k}$) can also be thought of in this way, see the shaded regions in Figure 5.9. Specifically, (s,t) satisfies the constraint if $t \geq t_1$, $s \leq s_2$, and if $s \in [s^l, s^r]$ then (s,t) must lie above the curve $f_{i,j,k}$. Otherwise (s,t) does not satisfy the constraint.

Any valid path in the relative free space must have a well defined cell sequence $(C_1, \ldots, C_{n+m-1})$ and passing sequence $\mathsf{pass}(p) = (h_1, \ldots, h_{n+m-1})$ (see Definition 5.4.1). Moreover, such a pair of sequences precisely determine a subset of the constraints defined above, such that there is a valid path with this cell and passing sequence if and only if all constraints in the subset are satisfied (this is implied by Lemma 5.4.2). In other words, for a given cell and passing sequence we want to solve the optimization problem:

$$
\begin{aligned}
\min \quad & t - s \\
\text{subject to} \quad & \\
& t \geq s \geq 0, \\
& s \leq \alpha, \text{ where } \alpha = \min\{\alpha_0, \alpha^l_{i_1,j_1}, \alpha^b_{i_2,j_2}, \alpha_{i_3,(j_3,k_3)}, \alpha_{(i_4,j_4),k_4}\} \text{ over all} \\
& \qquad (i_1,j_1) \in Z_1, (i_2,j_2) \in Z_2, (i_3,j_3,k_3) \in Z_3, (i_4,j_4,k_4) \in Z_4 \\
& t \geq \beta, \text{ where } \beta = \max\{\beta_0, \beta^l_{i_5,j_5}, \beta^b_{i_6,j_6}, \beta_{i_7,(j_7,k_7)}, \beta_{(i_8,j_8),k_8}\} \text{ over all} \\
& \qquad (i_5,j_5) \in Z_5, (i_6,j_6) \in Z_6, (i_7,j_7,k_7) \in Z_7, (i_8,j_8,k_8) \in Z_8 \\
& t \geq f_{i,j,k}(s), \text{ for } s \in [s^{lf}_{i,j,k}, s^{rf}_{i,j,k}] \quad \forall (i,j,k) \in Z_9 \\
& t \geq g_{i,j,k}(s), \text{ for } s \in [s^{lg}_{i,j,k}, s^{rg}_{i,j,k}] \quad \forall (i,j,k) \in Z_{10}
\end{aligned}
$$

where the sets $Z_1, \ldots, Z_{10}$ of index tuples are determined naturally by the cell and passing sequences. (We skip the tedious and unenlightening description of these sets.)

Clearly the optimal value of this optimization problem must lie on the boundary of at least one constraint. In particular, the optimum lies either at the intersection point of the boundaries of two constraints, or at a local minimum of one of the boundary constraints, with respect to the objective of minimizing $t - s$. By Observation 5.4.3, each $f_{i,j,k}$ or $g_{i,j,k}$ has at most a constant number of local minima, and as the boundaries of all other constraints are straight lines, this is also true for every boundary function. Thus we have now determined the set of canonical critical intervals discussed earlier in this section.

Lemma 5.4.4. *The above defined constraints, determined by all types of critical events, determine an $O(n^6)$ sized set of canonical critical intervals, i.e. (s,t) pairs, that must contain an optimal gap $[s^*, t^*]$.*

Proof. Any optimal gap determines a cell and passing sequence of some valid path in the corresponding relative free space. Above it was discussed how such sequences determine a

subset of constraints, where the optimum gap width is determined either at an intersection of the boundaries of two constraints or at a local minimum of an $f_{i,j,k}$ or $g_{i,j,k}$. Now a priori we do not know the cell and passing sequence of a path determining an optimal gap, hence we will consider them all. So consider the arrangement of all planar curves defined by the boundaries of any of the possible constraints defined above. There are a constant number of initialization constraints, $O(n^2)$ possible connectivity constraints, and $O(n^3)$ possible standard or floating monotonicity constraints. Due to the particularly nice form of these curves, each pair intersect at most a constant number of times, and hence there are $O(n^6)$ intersections overall. Moreover, as discussed above, each curve has only a constant number of local minima with respect the the objective of minimizing $t - s$. Hence this arrangement determines as set of $O(n^6)$ points, at least one of which realizes the minimum gap width. □

The above discussion also implies the following observation.

Observation 5.4.5. *Whether or not a given (s,t)-pair is feasible for the Fréchet gap problem is solely determined by which constraints the point satisfies or does not satisfy. So consider the arrangement of curves determined by the boundaries of all the constraint types discussed above. Then within the interior of a given cell of the arrangement all (s,t)-pairs are thus either all feasible or all infeasible.*

5.5 Exact Computation of the Fréchet Gap Distance

The $O(n^6)$ critical intervals given by Lemma 5.4.4 together with the $O(n^2)$ decider of Theorem 5.3.3, naively give only an $O(n^8)$ algorithm for computing the Fréchet gap distance, as there is no immediate linear ordering to search over the events. However, here we give a much faster $O(n^5 \log n)$ time algorithm to compute the Fréchet gap distance exactly.

The standard Fréchet distance is computed in $O(n^2 \log n)$ time by searching over the $O(n^3)$ critical events with an $O(n^2)$ time decision procedure. This searching originally was done

with parametric search [11], though for our purposes the simpler sampling based approach of [60] is more relevant.

Searching is a far more challenging task in the Fréchet gap setting. Specifically, in the standard Fréchet case there is a linear ordering of the critical events, and in this ordering all events are infeasible up until the true Fréchet distance, and then feasible afterwards. However, in our two dimensional parametric space there is no such natural linear ordering. Moreover, recall that even if an interval $[s, t]$ is feasible, it does *not* imply $[s, t]$ contains an optimal gap as a subinterval. Crucially however, it still holds that the feasible points form a connected set in the parametric space.

Lemma 5.5.1. *In the parametric space, the set of feasible (s, t) pairs is a connected set.*

Proof. Given any two feasible points (s_1, t_1) and (s_2, t_2), we describe a path between them consisting of only feasible points (thus implying the feasible set is connected). Observe that if a point (s, t) is feasible, then any point (s', t') such that $s' \leq s$ and $t' \geq t$ is also feasible. This implies the line segments $(0, t_1)(s_1, t_1)$ and $(0, t_2)(s_2, t_2)$ consist solely of feasible points. Again applying this observation, the segment $(0, t_1)(0, t_2)$ consists solely of feasible points. Thus the path $((s_1, t_1), (0, t_1), (0, t_2), (s_2, t_2))$, consists only of feasible points. □

The algorithm for exactly computing the Fréchet gap distance uses the following subroutines:

- **deciderPoint**(s, t): Decides whether or not the pair (s, t) is feasible, in $O(n^2)$ time.
- **deciderLine**(c): Given a positive number c, returns "below" if there is any feasible (s, t)-pair with $t - s \leq c$, and returns "above" otherwise. The running time is $O(n^5)$.
- **sample**(r): Samples r (s, t)-pairs, independently and uniformly at random, from the set of $O(n^6)$ canonical critical pairs of Lemma 5.4.4. The running time is $O(r)$.
- **sweep**(c_1, c_2): Returns the set of all canonical critical (s, t)-pairs of Lemma 5.4.4 such that $c_1 \leq t - s \leq c_2$, in $O((n^3 + k) \log n)$ time, where k is the number of such critical pairs.

First observe the subroutine **deciderPoint**(s,t) is given by Theorem 5.3.3. **deciderLine**(c) is computed as follows. First compute the intersection points of the line, $t-s=c$, with the $O(n^3)$ boundaries of all the constraints discussed in Section 5.4.1. Since these constraints are horizontal/vertical lines or $f_{i,j,k}/g_{i,j,k}$, by Observation 5.4.3, there are $O(n^3)$ intersection points. Thus calling **deciderPoint** on each of these intersection points, takes $O(n^5)$ time as **deciderPoint** takes $O(n^2)$ time. By Observation 5.4.5, if all these point queries return infeasible, then all points on the line $t-s=c$ are infeasible. In this case, since by Lemma 5.5.1 the feasible region is connected, any optimal gap pair must lie above the the line $t-s=c$. On the other hand, again by by Lemma 5.5.1, if one of the point queries returned true then any optimal gap pair must lie below (or on) the line $t-s=c$.

The subroutine **sample**(r) is also straightforward. Specifically, every canonical critical pair is either a local minima or an intersection of the boundaries of two constraints from Section 5.4.1. Thus in order to sample a canonical critical pair, we sample either one or two constraints[3], where whether we sample one or two is done in proportion to the number of pairs versus single constraints. Each constraint is determined by either a pair or triple of indices (and a few bits, such as whether the side of bottom of a cell, etc.), and hence each can be sampled in $O(1)$ time (again done proportionally to the number of triples versus pairs of indices). Thus r canonical pairs can be sampled in $O(r)$ time.

Thus what remains is to describe the subroutine **sweep**, for which we have the following.

Lemma 5.5.2. *Given two real values $0 \le c_1 \le c_2$, one can compute the set of all canonical critical (s,t)-pairs of Lemma 5.4.4 such that $c_1 \le t-s \le c_2$, in $O((n^3+k)\log n)$ time, where k is the number of such critical pairs. This algorithm is denoted* **sweep**(c_1,c_2).

Proof. It is well known that one can compute the set of all k intersection points of a set of m x-monotone constant-complexity curves in $O((m+k)\log m)$ time using a horizontal sweep

[3] Note the number of local minima per constraint and the number of times two constraints intersect is a constant, but the constant may be more than one. Thus technically the described sampling is not truly uniform. One can make it uniform, though this distinction is irrelevant for our asymptotic analysis.

line in the standard sweep line algorithm of Bentley and Ottmann [61]. In our case the curves are given by the $O(n^3)$ constraints of Section 5.4.1, clipped to only be defined in the region bounded by the lines $t - s = c_1$ and $t - s = c_2$. The constraints with straight line boundaries are s-monotone, and by Observation 5.4.3 so are the $f_{i,j,k}$ and $g_{i,j,k}$. Thus the claim follows by applying the standard sweep line algorithm to our case. Note that our problem involves degenerate horizontal line segments, which can still be handled (with some care) when using a horizontal sweep line, though alternatively one could avoid such issues using a diagonal $t - s = c$ sweep line in which case one should first cut the $f_{i,j,k}/g_{i,j,k}$ into pieces at their local maxima/minima (with respect to the line $t - s = c$), to maintain monotonicity. □

1 $\mathcal{R} = \textbf{sample}(\alpha n^4)$ `// α a sufficiently large constant`
2 Sort $\widehat{\mathcal{R}} = \{c = t - s \mid (s,t) \in \mathcal{R}\}$ in increasing order
3 Binary search over $\widehat{\mathcal{R}}$ using $\textbf{deciderLine}(c)$ for the interval $[c_1, c_2]$ s.t.
  $\textbf{deciderLine}(c_1) =$ above and $\textbf{deciderLine}(c_2) =$ below `// Set initial values`
  $c_1 = 0,\ c_2 = \infty$
4 $\mathcal{S} = \textbf{sweep}(c_1, c_2)$
5 Call $\textbf{deciderPoint}(s,t)$ on each $(s,t) \in \mathcal{S}$, and
  return the feasible pair with smallest $t - s$ value.

Algorithm 4: Computing the Fréchet gap distance

The algorithm for computing the Fréchet gap distance is shown in Algorithm 4. We need the following lemma to bound the number of critical pairs that we end up searching over.

Lemma 5.5.3. *Let $[c_1, c_2]$ be the interval described in Algorithm 4. Then with exponentially high probability, this interval contains $O(n^3)$ canonical critical pairs.*

Proof. Let $\mathcal{C}$ be the set of all canonical critical pairs as described in Lemma 5.4.4, thus $|\mathcal{C}| \leq \beta n^6$ for some constant β. Let $\widehat{\mathcal{C}} = \{c = t - s \mid (s,t) \in \mathcal{C}\}$, and let $\widehat{\mathcal{R}}$ be the sampled subset of $\widehat{\mathcal{C}}$ as described in Algorithm 4. Note that multiple values in $\mathcal{C}$ may map to a single value in $\widehat{\mathcal{C}}$. This technicality is discussed below, but for now assume $|\mathcal{C}| = |\widehat{\mathcal{C}}|$.

Consider the sorted placement of the values in $\widehat{\mathcal{C}}$ along the real line, and let x be any point on the real line. Let U^+ (resp. U^-) be the closest n^3 values from $\widehat{\mathcal{C}}$ larger (resp. smaller) than x. Suppose $|U^+| = n^3$ (note it may happen that $|U^+| < n^3$, if x is large enough). Then the probability it does not contain a value in $\widehat{\mathcal{R}}$ is

$$\left(1 - \frac{|U^+|}{\beta n^6}\right)^{\alpha n^4} = \left(1 - \frac{n^3}{\beta n^6}\right)^{\alpha n^4} \leq \exp\left(\frac{-\alpha n^7}{\beta n^6}\right) = \exp(-(\alpha/\beta)n) \leq e^{-\hat{c}n}$$

for any constant $\hat{c}$, by choosing α sufficiently large. A similar statement holds for $|U^-|$, and thus taking the union bound, a similar statement holds simultaneously for both U^- and U^+.

Note there are only a polynomial number of possibilities for each value in $\widehat{\mathcal{R}}$ (specifically $O(n^6)$). Thus by setting x to each one of these values, and taking the union bound, it holds that between any two adjacent values in $\widehat{\mathcal{R}}$ (or in the unbounded end intervals), with high probability there are at most $O(n^3)$ values of $\widehat{\mathcal{C}}$, thus implying the claim.

As mentioned above, technically multiple pairs in $\mathcal{C}$ may map to a single value in $\widehat{\mathcal{C}}$. This can be remedied by treating $\widehat{\mathcal{C}}$ as a multi-set. Then by defining an arbitrary order over multi-values, the above analysis will still hold, except potentially at the endpoints of the interval $[c_1, c_2]$ (as we include *all* critical pairs with these values). Observe however that (in the absolute worst case) there are at most $O(n^3)$ pairs which get mapped to either value c_1 or c_2 in $\widehat{\mathcal{C}}$, and so the lemma holds. □

Note that there is a huge amount of slack in the above argument, and in more than one way. Specifically, even though we have an exponentially high probability bound, it can be further improved by taking a larger random sample. Also we could have argued, with polynomially high probability, that the number of canonical critical pairs in $[c_1, c_2]$ is only $O(n^2 \log n)$ (taking more care in the argument at the endpoints). However, ultimately this would not change the asymptotic running time, as the real bottleneck for the algorithm is in searching with the $O(n^5)$ time **deciderLine**.

Theorem 5.5.4. *Given polygonal curves π and σ, each of length at most n, Algorithm 4 computes the Fréchet gap distance in $O(n^5 \log n)$ time.*

Proof. The correctness of Algorithm 4 has essentially already been argued. Specifically, the random sample $\mathcal{R}$ partitions the real line into intervals based on the values in $\widehat{\mathcal{R}}$. One of these intervals contains the optimal gap width, implying the interval $[c_1, c_2]$ found by searching using **deciderLine**(c) is well defined. Moreover, $\mathcal{S}$ contains a canonical critical pair with optimal gap width as **sweep**(c_1, c_2) returns all canonical critical pairs in the region bounded by the lines $t - s = c_1$ and $t - s = c_2$, and by Lemma 5.4.4 the set of canonical critical pairs contains a pair with optimal gap width. As **deciderPoint** is called on all pairs in $\mathcal{S}$, the algorithm will find this optimal gap pair.

For the running time, calling **sample**(αn^4) takes $O(n^4)$ time. Sorting $\widehat{\mathcal{R}}$ takes $O(n^4 \log n)$ time, and searching over $\widehat{\mathcal{R}}$ takes $O(n^5 \log n)$ time as **deciderLine** takes $O(n^5)$ time. By Lemma 5.5.2, **sweep**(c_1, c_2) takes $O((n^3 + |\mathcal{S}|) \log n)$ time. Calling **deciderPoint** on each pair in $\mathcal{S}$ takes $O(|\mathcal{S}|n^2)$ time, as **deciderPoint** takes $O(n^2)$ time. By Lemma 5.5.3, with high probability $|\mathcal{S}| = O(n^3)$, so sweeping and all **deciderPoint** calls combined take $O(n^5)$ time. Thus the overall time is $O(n^5 \log n)$, i.e. dominated by the time to search with **deciderLine**. □

5.6 Approximation

In this section, we propose an efficient algorithm to approximate the Fréchet gap distance, based on the following simple fact. Let d_o be the average of the starting and ending vertex pair distances of π and σ, that is $d_o = \frac{d_b + d_e}{2}$ where $d_b = ||\pi_0 - \sigma_0||$ and $d_e = ||\pi_n - \sigma_m||$.

Observation 5.6.1. *If a parametric point (s, t) is feasible then $s \leq d_o \leq t$.*

This implies we only need to consider parametric points such that $s \leq d_o \leq t$, which we call *centered* points. Define the *radius* of any such point (s, t) to be $\mathsf{r}_{s,t} = \max\{t - d_o, d_o - s\}$, and define the *projection* to be $proj(s, t) = (d_o - \mathsf{r}_{s,t}, d_o + \mathsf{r}_{s,t})$. See Figure 5.10.

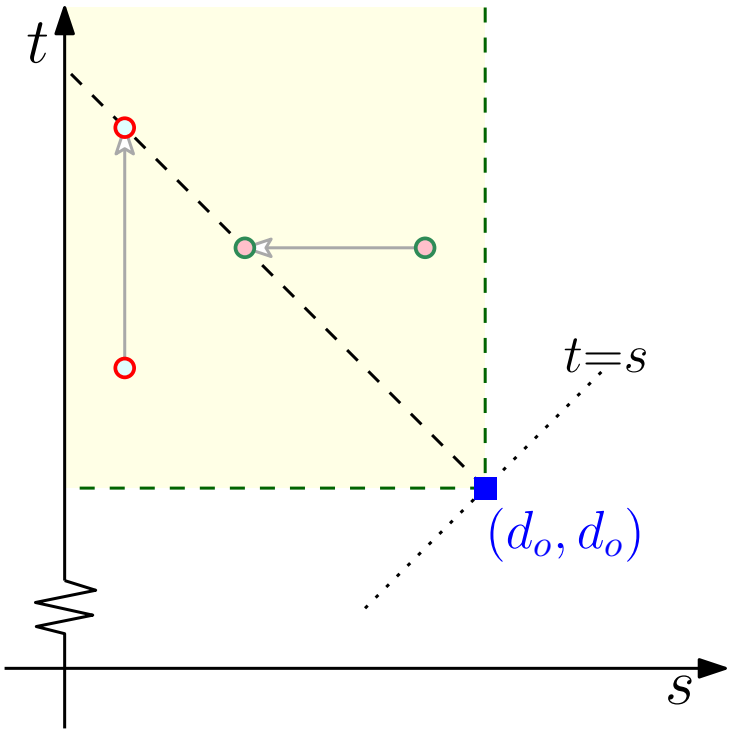


Figure 5.10: Two centered points and their projections.

Observe that in order to get a 2-approximation it suffices to restrict our attention to projected points (as $[s,t] \subseteq [d_o - \mathsf{r}_{s,t}, d_o + \mathsf{r}_{s,t}]$ for any centered point (s,t)), and the advantage is that projected points are more nicely behaved. Specifically, projected points define a linear ordering by the parameter r with the nice property that if $(d_o - \mathsf{r}, d_o + \mathsf{r})$ is feasible then for any $\mathsf{r}' \geq \mathsf{r}$ it holds that $(d_o - \mathsf{r}', d_o + \mathsf{r}')$ is also feasible. Moreover, below we show that the $O(n^6)$ critical intervals of Lemma 5.4.4, can be reduced to $O(n^3)$ in this setting, intuitively since now there is only a single parameter r, rather than independent s and t parameters.

5.6.1 Simplification of Critical Events

In Section 5.4.1 we described a set of four different types of constraints over the (s,t) parametric space (relating to initialization, connectivity, and standard and floating monotonicity events), and saw that the Fréchet gap distance is realized by minimizing $t - s$ over some (unknown) subset of these constraints. Recall from that section that each such constraints partitions the

parametric space into two sets, those satisfying and those violating that constraint. Label these $O(n^3)$ constraints in an arbitrary fashion from $1, \ldots, cn^3$, and for the $\imath$th constraint let $\mathcal{P}_\imath$ denote the set of satisfying points in the parametric space (which is a single connected region). We will assume that the $\mathcal{P}_\imath$ are clipped to the subset of centered points such that $s \leq d_o \leq t$, as we know any optimal gap pair must lie in this region.

Note that any given constraint in Section 5.4.1 is satisfied by lying to the left and above a straight line or nicely behave monotonically increasing function, hence we have the following.

Observation 5.6.2. *If $(s,t) \in \mathcal{P}_\imath$ then $(s',t') \in \mathcal{P}_\imath$ for any $s' \leq s$, $t' \geq t$. This implies:*

1. *$proj(\mathcal{P}_\imath) \subseteq \mathcal{P}_\imath$, where $proj(\mathcal{P}_\imath) = \{proj(s,t) \mid (s,t) \in \mathcal{P}_\imath\}$.*
2. *If $(d_o - \mathsf{r}, \mathsf{r} - d_o) \in proj(\mathcal{P}_\imath)$ then $(d_o - \mathsf{r}', \mathsf{r}' - d_o) \in proj(\mathcal{P}_\imath)$ for any $\mathsf{r}' \geq \mathsf{r}$.*

For each $\mathcal{P}_\imath$ let $D_\imath$ be the minimum radius of a point in $proj(\mathcal{P}_\imath)$, and let $\mathcal{D}$ denote the set of all $D_\imath$. Note that each $\mathcal{P}_\imath$ is a constant complexity region, and so computing $D_\imath$ is a constant time operation.

Lemma 5.6.3. *There is a value $D \in \mathcal{D}$ such that $[d_o - D, d_0 + D]$ is a 2-approximation to an optimal Fréchet gap, that is $[d_o - D, d_0 + D]$ is feasible and $D \leq d_{\mathcal{G}}(\pi, \sigma)$.*

Proof. Let (s^*, t^*) be a point realizing the Fréchet gap distance. Then (s^*, t^*) is determined by some k constraints from Section 5.4.1, and let $I = \{\imath_1, \imath_2, \ldots \imath_k\}$ be the set of indices of these constraints. Specifically, in that section we argued any point in $\bigcap_{\imath \in I} \mathcal{P}_\imath$ is feasible and (s^*, t^*) is the point in the intersection with minimum gap.

Let $D = \max_{\imath \in I} D_\imath$. Since for all $\imath$, $(d_o - D_\imath, d_o + D_\imath) \in proj(\mathcal{P}_\imath)$, the second part of Observation 5.6.2 implies $(d_o - D, d_o + D) \in \bigcap_{\imath \in I} proj(\mathcal{P}_\imath)$. The first part of Observation 5.6.2 implies any point in $\bigcap_{\imath \in I} proj(\mathcal{P}_\imath)$ is also in $\bigcap_{\imath \in I} \mathcal{P}_\imath$, and hence $(d_o - D, d_o - D)$ is feasible.

Let $proj(s^*, t^*) = (d_o - \mathsf{r}^*, d_o + \mathsf{r}^*)$, which is in $\bigcap_{\imath \in I} proj(\mathcal{P}_\imath)$ as $(s^*, t^*) \in \bigcap_{\imath \in I} \mathcal{P}_\imath$. For some index $\imath$, $D = D_\imath$, and since each $D_\imath$ is defined as the minimum radius of a point in $proj(\mathcal{P}_\imath)$, $(d_o - D, d_o + D)$ must therefore be the minimum radius point in $\bigcap_{\imath \in I} proj(\mathcal{P}_\imath)$, and so clearly

$\mathsf{r}^* \geq D$. By definition, $r^* = \max\{d_o - s^*, t^* - d_o\}$ and so $D \leq \max\{d_o - s^*, t^* - d_o\} \leq t - s = d_{\mathcal{G}}(\pi, \sigma)$. □

5.6.2 Approximate Decider

Here we show how to efficiently convert any constant factor approximation into a $(1+\varepsilon)$-approximation, which is relevant as the previous section proved one of the $O(n^3)$ values in $\mathcal{D}$ is a 2-approximation. Specifically, we seek an efficient version of the following decider.

Definition 5.6.4. **appDeciderLine**(c, ε)*: Given positive numbers* c, ε*, returns "true" if* $d_{\mathcal{G}}(\pi, \sigma) \leq c$*, and returns "false" if* $d_{\mathcal{G}}(\pi, \sigma) > (1+\varepsilon)c$*. Either "true" or "false" can be returned if* $d_{\mathcal{G}}(\pi, \sigma) \in (c, (1+\varepsilon)c]$.

Lemma 5.6.5. *There exist an* $O(n^2/\varepsilon)$ *time algorithm for* **appDeciderLine**(c, ε).

Proof. By Observation 5.6.1, $d_o \in [s, t]$ for any feasible interval $[s, t]$. Thus any feasible interval $[s, t]$ with $t - s \leq c$ is contained in the interval $[d_o - c, d_o + c]$, and hence we restrict our attention to this interval. We cover this interval with successive overlapping subintervals of width $(1+\varepsilon)c$, and each shifted by $c\varepsilon$ from the previous one. Specifically, let S_g be the set of subintervals of the form $[d_o - c + i(c\varepsilon), d_o - c + i(c\varepsilon) + (1+\varepsilon)c]$, for $i = 0, \ldots, \lceil \frac{1}{\varepsilon} \rceil$ (Note to make calculations easier below we stop at $d_o - c + \lceil \frac{1}{\varepsilon} \rceil (c\varepsilon) + (1+\varepsilon)c \geq d_o + (1+\varepsilon)c$ rather than $d_o + c$). Our algorithm for **appDeciderLine**(c, ε) simply checks each one of these intervals for feasibility using **deciderPoint**, and if any interval returns "true" then it returns "true" and otherwise it returns "false". **deciderPoint** correctly checks feasibility in $O(n^2)$ time by Theorem 5.3.3 and we are testing, $O(1/\varepsilon)$ intervals, so the running time bound is immediate. We now prove this procedure satisfies the properties of Definition 5.6.4.

If $d_{\mathcal{G}}(\pi, \sigma) \leq c$, then there is a feasible interval $[s, t]$ with $t - s \leq c$, which implies $[s, t] \subseteq [s, s + c] \subseteq [d_o - c + j(c\varepsilon), d_o - c + j(c\varepsilon) + (1+\varepsilon)c]$ where $j = \lfloor \frac{(s - d_o + c)}{c\varepsilon} \rfloor$. One can easily verify that $0 \leq j \leq \lceil 1/\varepsilon \rceil$, and so this interval is in S_g. Hence at least one interval in S_g is

feasible (as containing a feasible subinterval implies feasibility), and so **appDeciderLine**(c, ε) returns "true". On the other hand, if $d_{\mathcal{G}}(\pi, \sigma) > (1+\varepsilon)c$, then no interval in S_g is feasible as each interval in S_g has width $(1+\varepsilon)c$, and so **appDeciderLine**(c, ε) returns "false". Finally, if $c < d_{\mathcal{G}}(\pi, \sigma) \leq c(1+\varepsilon)$, then **appDeciderLine**(c, ε) returns "false" or "true", and we don't care which one. □

Using binary search the above **appDeciderLine**(c, ε) can be used to convert a constant factor approximation (i.e. constant spread interval) into a $(1+\varepsilon)$-approximation.

Lemma 5.6.6. *Given a value $c \geq 0$, one can decide whether $d_{\mathcal{G}}(\pi, \sigma) > c$, or $d_{\mathcal{G}}(\pi, \sigma) < c$, or obtain $(1+\varepsilon)$ approximation to $d_{\mathcal{G}}(\pi, \sigma)$, which can be done in $O(n^2/\varepsilon)$ time.*

Proof. If $c = 0$, just test whether $[d_o, d_o]$ is feasible, and if not return $d_{\mathcal{G}}(\pi, \sigma) > c$. Otherwise, call **appDeciderLine**(c, ε') and **appDeciderLine**$(c/(1+2\varepsilon'), \varepsilon')$, for a value ε' to be determined shortly. Taking the contrapositive of the statements in Definition 5.6.4, if **appDeciderLine**(c, ε) returns "true" then $d_{\mathcal{G}}(\pi, \sigma) \leq c(1+\varepsilon)$, and if **appDeciderLine**(c, ε) returns "false" then $d_{\mathcal{G}}(\pi, \sigma) > c$.

So if **appDeciderLine**(c, ε') returns "false" then $d_{\mathcal{G}}(\pi, \sigma) > c$, and if **appDeciderLine**$(c/(1+2\varepsilon'), \varepsilon')$ returns "true", then $d_{\mathcal{G}}(\pi, \sigma) \leq \frac{c}{(1+2\varepsilon')}(1+\varepsilon') < c$. Otherwise $d_{\mathcal{G}}(\pi, \sigma) \in (\frac{c}{(1+2\varepsilon')}, c(1+\varepsilon')] = \frac{c}{1+2\varepsilon'}(1, (1+\varepsilon')(1+2\varepsilon')] \subset \frac{c}{1+2\varepsilon'}(1, 1+5\varepsilon') = \frac{c}{1+2\varepsilon/5}(1, 1+\varepsilon)$ where $\varepsilon' = \varepsilon/5 < 1$. □

Lemma 5.6.7. *Given an interval $[\alpha, \beta]$, with $\alpha > 0$, one can either report "$d_{\mathcal{G}}(\pi, \sigma) \notin [\alpha, \beta]$" in $O(n^2/\varepsilon)$ time, or obtain $(1+\varepsilon)$ approximation to $d_{\mathcal{G}}(\pi, \sigma)$ in $O(\frac{n^2}{\varepsilon} \log \frac{\beta}{\alpha\varepsilon})$ time, which simplifies to $O(\frac{n^2}{\varepsilon} \log \frac{1}{\varepsilon})$ time when $\beta = O(\alpha)$.*

Proof. By using the Lemma 5.6.6, one can decide whether $d_{\mathcal{G}}(\pi, \sigma) < \alpha$, $d_{\mathcal{G}}(\pi, \sigma) > \beta$, or obtain $(1+\varepsilon)$ approximation to $d_{\mathcal{G}}(\pi, \sigma)$ in $O(n^2/\varepsilon)$ time.

If $d_{\mathcal{G}}(\pi, \sigma) \in [\alpha, \beta]$, divide the interval $[\alpha, \beta]$ into sub intervals with equal step distance $\alpha\varepsilon$ and perform a binary search over these sub intervals. When we try the sub interval

$[\gamma, \gamma + \alpha\epsilon]$, by using Lemma 5.6.6, one can decide whether $d_{\mathcal{G}}(\pi, \sigma) < \gamma$, $d_{\mathcal{G}}(\pi, \sigma) > \gamma + \alpha\epsilon$. If $d_{\mathcal{G}}(\pi, \sigma) < \gamma$, continue binary search on the median sub interval between α and γ. Else if $d_{\mathcal{G}}(\pi, \sigma) > \gamma + \alpha\epsilon$, then continue binary search on the median sub interval between $\gamma + \alpha\epsilon$ and β, otherwise $d_{\mathcal{G}}(\pi, \sigma) \in [\gamma, \gamma + \alpha\varepsilon] \subset [\gamma, \gamma(1 + \epsilon)]$.

We searched over $O(\frac{\beta}{\alpha\varepsilon})$ values, requiring $O(\log \frac{\beta}{\alpha\varepsilon})$ calls to **appDeciderLine**$(c, \varepsilon/5)$, hence our procedure takes $O(\frac{n^2}{\varepsilon} \log \frac{\beta}{\alpha\varepsilon})$ time overall as claimed. □

Corollary 5.6.8. *One can* $(1 + \varepsilon)$*-approximate* $d_{\mathcal{G}}(\pi, \sigma)$ *in* $O(n^3 + n^2 \log(n) \frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$ *time.*

Proof. By Lemma 5.6.3 there is some $D \in \mathcal{D}$ such that $d_{\mathcal{G}}(\pi, \sigma) \in [D, 2D]$. Thus if for each $D \in \mathcal{D}$ we call the procedure of Lemma 5.6.7 to search the interval $[D, 2D]$, then we are guaranteed find a $(1 + \varepsilon)$-approximation. Note that we may query the value $D = 0$, which does not satisfy the conditions of Lemma 5.6.7, though we can easily check if $d_{\mathcal{G}}(\pi, \sigma) = 0$, as then $s = d_o = t$. Each interval we query takes $O(\frac{n^2}{\varepsilon} \log \frac{1}{\varepsilon})$ time. Thus by using standard linear time median selection over the $O(n^3)$ values in $\mathcal{D}$, the running time follows. □

5.6.3 Improving the running time

Here we show how to use sampling to improve the running time of Corollary 5.6.8 by nearly a linear factor. Specifically, our goal is to find the value $D \in \mathcal{D}$ for which by Lemma 5.6.3 $[d_o - D, d_o + D]$ is a 2-approximation to an optimum gap interval.

The approach is similar to that in Section 5.5 (and even closer to the algorithm in [60]). The main difference is that the description of the functions used in the sweeping procedure is more involved, and so we describe this subroutine first before describing the full algorithm.

Sweeping

Given values $\alpha \leq \beta$, we seek a procedure **sweep**(α, β) which returns a superset of all values $D_\imath \in \mathcal{D}$ such that $D_\imath \in [\alpha, \beta]$. The $\mathcal{P}_\imath$ regions differ based on which constraint type from

Section 5.4.1 they correspond to. In particular, $\mathcal{D} = \mathcal{D}_{conn} \cup \mathcal{D}_{mono}$, where $\mathcal{D}_{mono}$ is the set of $D_i \in \mathcal{D}$ corresponding to regions which represent monotonicity events (either standard or floating), and $\mathcal{D}_{conn}$ is the set corresponding to all connectivity events (plus the initialization events). The set $\mathcal{D}_{conn}$ has size $O(n^2)$, and thus all values from this set in $[\alpha, \beta]$ can be found by brute force in quadratic time, so from now on we only consider the set $\mathcal{D}_{mono}$.

Consider all $D_i \in \mathcal{D}_{mono}$ corresponding to a fixed row, i.e. a fixed edge $\pi_i\pi_{i+1}$, of the free space. As discussed in Section 5.4.1, for any column indices $j \leq k$, a standard monotonicity event occurs at a value of t such that $height(\mathcal{I}_j^b) = height(\mathcal{I}_k^a)$ or a value of s such that $height(\mathcal{H}_j^a) = height(\mathcal{H}_k^b)$, and a floating monotonicity event occurs at pairs s, t such that $height(\mathcal{I}_j^b) = height(\mathcal{H}_k^b)$ or $height(\mathcal{H}_j^a) = height(\mathcal{I}_k^a)$. From that section we also know the functions for s and t in terms of these heights. Specifically, $height(\mathcal{I}_k^a)$ is determined by the distance from σ_k, and so the function for t in terms of the height $h = height(\mathcal{I}_k^a)$ is given by $t = \sqrt{\chi^2 + (\gamma - h)^2}$, where χ, γ are the coordinates of vertex σ_k (in the $\pi_i, \pi_{i+1}, \sigma_k$ plane). Note this is also the function for t in terms of $h = height(\mathcal{I}_k^b)$ (i.e. the function is symmetric with one side representing $\mathcal{I}_k^a$ and the other $\mathcal{I}_k^b$). $height(\mathcal{H}_j^a)$ on the other hand is determined by the distance from the edge $\sigma_j\sigma_{j+1}$, and so the function for s in terms of the height $h = height(\mathcal{H}_j^a)$ has three cases, based on whether it is a distance to one of two edge endpoints or an interior point. For the endpoint cases again we have functions of the form $s = \sqrt{\alpha^2 + (\beta - h)^2}$, for some constants α, β, and just as before these functions also describe the endpoint cases for $height(\mathcal{H}_j^b)$. For the interior case the function is of the form $s = c \cdot h + d$ for both $height(\mathcal{H}_j^a)$ and $height(\mathcal{H}_j^b)$, although the constants in the linear functions for $height(\mathcal{H}_j^a)$ and $height(\mathcal{H}_j^b)$ can differ. (Slopes may have opposite sign when $dist(\pi_i\pi_{i+1}, \sigma_j\sigma_{j+1})$ is realized at the interior of both edges, a case not shown in Figure 5.8.)

Fix a row of the free space, and for all pairs of column indices $j \leq k$ plot the above described function for t and all the functions for s (endpoints and interior functions). Specifically, in the plot of these functions the horizontal axis is h and the vertical axis is both s and t.

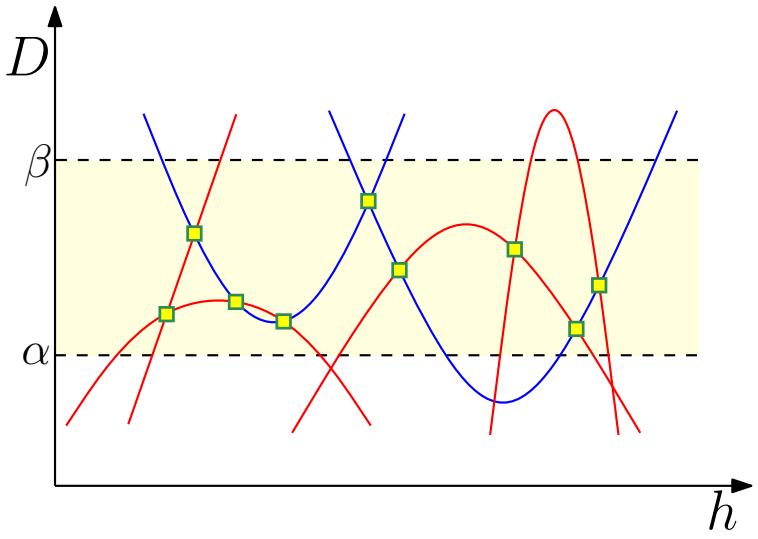


Figure 5.11: Possible $d_o - s$ functions (in red) and $t - d_o$ functions (in blue). The horizontal axis is the height h and the vertical axis is the radius D, i.e. distance from d_o.

Consider a standard monotonicity event. This happens at a value of t (resp. s) such that $h = height(\mathcal{I}_j^b) = height(\mathcal{I}_k^a)$ (resp. $h = height(\mathcal{H}_j^a) = height(\mathcal{H}_k^b)$), i.e. at an intersection point of two of the plotted t (resp. s) functions. Floating monotonicity events as always are a bit trickier (actually the easier standard monotonicity case was already described in [60], though in a different way). Specifically, while any floating event still occurs at a single h value, such events in general do not occur at intersections of the s and t functions since s and t may differ in value. The key observation however is that now we are only concerned with projected points, i.e. points such that $d_o - s = t - d_o$. So instead plot all functions of the form $d_o - s$, $t - d_o$ where s and t are any of the functions described above. In this new plot the horizontal coordinate is again h but the vertical coordinate is now the radius D (i.e. distance from d_o), and projected points now occur at the intersections of these transformed s and t functions, see Figure 5.11. (Note the standard events still occur at intersections.)

Definition 5.6.9. *For a fixed row i of the free space, consider the arrangement of all functions of the form $d_o - s$ and $t - d_o$ in the 2-dimensional D, h space, where s and t are given by the above described functions of h for any pair of columns $j \leq k$. Define $\mathcal{Z}_i^r$ to be the set of*

radius D values of all intersection points in this arrangement for row i, and similarly define $\mathcal{Z}_i^c$ for column i. The set of all intersection radii over all rows and columns is denoted $\mathcal{Z}$.

It would appear from the above discussion that $\mathcal{Z}$ is a superset of $\mathcal{D}_{mono}$, though there is one subtlety. Consider the line in the s, t parametric space defined by the equation $d_o - s = t - d_o$, i.e. the line defining all projected points. For a standard monotonicity event the boundary of $\mathcal{P}_\imath$ is a single horizontal (or vertical) line, which intersects $d_o - s = t - d_o$ and thus the minimum projected point in the region lies on this line boundary[4]. On the other hand, if $\mathcal{P}_\imath$ is from a floating monotonicity event then the boundary is more complicated, as show in Figure 5.9. In this figure, only the blue curve relates to an intersection point from the D, h space, and thus we also must consider when the minimum projected point is not on this curve, i.e. when $d_o - s = t - d_o$ intersects a different part of the boundary. There are three cases. The first two case are when the minimum projected point lies on either the horizontal line at t_1 or the vertical line at s_2, however, as mentioned in Section 5.4.1 these boundaries correspond to connectivity events and hence are already covered by in the set $\mathcal{D}_{conn}$. The third case is when the minimum point lies on the vertical line at $s^l = s_1$ (i.e. the left part of Figure 5.9). Recall $s_1 = dist(\sigma_j\sigma_{j+1}, \pi_i\pi_{i+1})$, that is even though there are $O(n^2)$ functions $f_{i,j,k}$ in the ith free space row, there are only $O(n)$ distinct s_1 values, indexed by edges of σ. Thus we find all intersections of the line $d_o - s = t - d_o$ with the lines $s = s_1$ for all possible s_1. Let $\mathcal{B}$ be the corresponding radii of all such intersections over all rows, and note that $|\mathcal{B}| = O(n^2)$. Thus from the above discussion, $\mathcal{D}_{conn} \cup \mathcal{Z} \cup \mathcal{B}$ is a superset of $\mathcal{D}$ and so we have the following.

Lemma 5.6.10. *For any values $\alpha \leq \beta$, there is an algorithm* **sweep**(α, β) *which in $O((n^2 + |W|) \log n)$ time outputs a set $W \supseteq (\mathcal{D} \cap [\alpha, \beta])$, such that $|W| = O(\max\{n^2, |\mathcal{Z} \cap [\alpha, \beta]|\})$, where $\mathcal{Z}$ is as defined in Definition 5.6.9.*

[4] Technically, here we ignore irrelevant, but possible, constraints where the horizontal line lies below d_o.

Proof. Above it was argued that $\mathcal{D} \subseteq (\mathcal{D}_{conn} \cup \mathcal{Z} \cup \mathcal{B})$. Thus for $W = \mathcal{D}_{conn} \cup (\mathcal{Z} \cap [\alpha, \beta]) \cup \mathcal{B}$, it holds that $W \supseteq (\mathcal{D} \cap [\alpha, \beta])$. As mentioned above, $|\mathcal{D}_{conn}|, |\mathcal{B}| = O(n^2)$, thus $|W| = O(\max\{n^2, |\mathcal{Z} \cap [\alpha, \beta]|\})$. Moreover, $\mathcal{D}_{conn}$ and $\mathcal{B}$ are easily computable in $O(n^2)$ time.

What remains is how to compute $\mathcal{Z} \cap [\alpha, \beta]$, which is done by using standard sweeping. So for some row i consider the set $\mathcal{Z}_i^r$. As discussed above this is the set of D values of the intersection points of a set of n constant complexity monotone functions in the D, h space. Thus $\mathcal{Z}_i^r \cap [\alpha, \beta]$ is determined by the set of all such intersections in a given horizontal strip $[\alpha, \beta]$, which just as in Lemma 5.5.2, can be found using a standard horizontal sweep line [61], which given a set of n monotone constant-complexity curves, finds all intersections in the interval $[\alpha, \beta]$ in $O((n + |\mathcal{Z}_i^r \cap [\alpha, \beta]|) \log n)$ time. Thus doing this sweeping for every row and column finds the set $\mathcal{Z} \cap [\alpha, \beta]$ in $O((n^2 + |\mathcal{Z} \cap [\alpha, \beta]|) \log n)$ time. □

The Algorithm

The full algorithm is shown in Algorithm 5, which the reader may observe has the same high level structure as Algorithm 4.

The only yet unspecified step of this algorithm is the first line where we call the subroutine **sampler**, which samples values from the set $\mathcal{Z}$ of Definition 5.6.9. Any value in $\mathcal{Z}$ is determined by an intersection point of a $d_o - s$ and a $t - d_o$ function, which in turn are specified by: (i) a triple of indices i, j, k, (ii) whether it is a row or column of the free space, and (iii) which of the $O(1)$ types of intersection points it is (i.e. whether it is standard or floating, if standard whether it is a t or s function intersection, also which type of the three possible s functions). Note alternatively, at the cost of increasing the sample size by a constant factor, after sampling i, j, k, one can include all $O(1)$ intersections this triple defines. Thus a value in $\mathcal{Z}$ can be sampled in constant time by sampling these pieces of information and thus **sampler**(γn^2) runs in $O(n^2)$ time.

Let $[D_\alpha, D_\beta]$ be as defined in Algorithm 5. The number of values from $\mathcal{Z}$ which fall in this interval affects the running time **sweep** (see Lemma 5.6.10). We now prove that with

1 $\mathcal{R} = \textbf{sampler}(\gamma n^2)$ // γ a sufficiently large constant
2 Sort $\mathcal{I} = \{[d_o - D, d_o + D] \mid D \in \mathcal{R}\}$ by the values in $\mathcal{R}$
3 Binary search over $\mathcal{I}$, using **deciderPoint**, for interval $[D_\alpha, D_\beta]$ such that **deciderPoint**$(d_o - D_\alpha, d_o + D_\alpha)$ false and **deciderPoint**$(d_o - D_\beta, d_o + D_\beta)$ true
// Set initial values $D_\alpha = 0$, $D_\beta = \infty$
4 $\mathcal{S} = \textbf{sweep}(D_\alpha, D_\beta)$
5 Sort $\mathcal{I}' = \{[d_o - D, d_o + D] \mid D \in \mathcal{S}\}$ by the values in $\mathcal{S}$
6 Binary search over $\mathcal{I}'$ using **deciderPoint** and return the smallest value $\hat{D} \in \mathcal{S}$ such that $[d_o - \hat{D}, d_o + \hat{D}]$ is feasible.

Algorithm 5: Sampling to compute a fast 2-approximation.

high probability this set has quadratic size. The proof is nearly identical to Lemma 5.5.3 (and a proof in [60]), and is included for the sake of completeness.

Lemma 5.6.11. *Let $[D_\alpha, D_\beta]$ be as defined in Algorithm 5. Then with exponentially high probability,* **sweep**(D_α, D_β) *returns a set of size $O(n^2)$.*

Proof. **sampler**(γn^2) takes a γn^2 sized sample $\mathcal{R}$ from the set $\mathcal{Z}$, which is a set of values of size ζn^3 for some constant ζ. Consider the sorted placement of the values in $\mathcal{Z}$ along the real line, and let x be any point on the real line. Let U^+ (resp. U^-) be the closest n^2 values from $\mathcal{Z}$ larger (resp. smaller) than x. Suppose $|U^+| = n^2$ (note it may happen that $|U^+| < n^2$, if x is large enough). Then the probability it does not contain a value in $\mathcal{R}$ is

$$\left(1 - \frac{|U^+|}{\zeta n^3}\right)^{\gamma n^2} = \left(1 - \frac{n^2}{\zeta n^3}\right)^{\gamma n^2} \leq \exp\left(\frac{-\gamma n^4}{\zeta n^3}\right) = \exp(-(\gamma/\zeta)n) \leq e^{-\hat{c}n}$$

for any constant $\hat{c}$, by choosing γ to be sufficiently large. A similar statement holds for $|U^-|$, and taking the union bound, a similar statement holds simultaneously for both U^- and U^+.

Note there are only $O(n^3)$ possibilities for each value in $\mathcal{R}$. Thus by setting x to each one of these values, and taking the union bound, it holds that between any two adjacent values in (the sorted set) $\mathcal{R}$, or in the unbounded end intervals, with high probability there are at most $O(n^2)$ values of $\mathcal{Z}$. Thus the lemma statement holds, as **sweep**(D_α, D_β) returns such a set of $\mathcal{Z}$ values, together with all of $\mathcal{D}_{corr}$ and $\mathcal{B}$ which are each of size $O(n^2)$. □

Theorem 5.6.12. *Given polygonal curves π and σ, each of length at most n, one can $(1+\varepsilon)$-approximate the Fréchet gap distance in $O(n^2(\log n + \frac{1}{\varepsilon}\log\frac{1}{\varepsilon}))$ time.*

Proof. As discussed above, we only require Algorithm 5 to return a 2-approximation, as such an approximation can then be transformed into $(1+\varepsilon)$-approximation in additive $O(\frac{n^2}{\varepsilon}\log\frac{1}{\varepsilon})$ time, using Lemma 5.6.7. Now by Lemma 5.6.3, there is some radius $D' \in \mathcal{D}$ whose corresponding projected point gives a 2-approximation. Recall from Observation 5.6.2, that sorting projected points by increasing radius D, induces a linear ordering of feasibility. Thus for the interval $[D_\alpha, D_\beta]$ found by the algorithm, it must be that $D' \geq D_\alpha$ since D_α is infeasible (or zero). Now if $D' \leq D_\beta$ then $D' \in [D_\alpha, D_\beta]$, and since **sweep** finds a superset of all $\mathcal{D}$ in $[D_\alpha, D_\beta]$, when we search over the returned values we must find at least a 2-approximation. Otherwise $D' > D_\beta$, and hence the feasible D_β or something even better will be found and returned from the sweeping, again getting at least a 2-approximation.

As for the running time, sampling $O(n^2)$ values and sorting takes $O(n^2 \log n)$ time. Binary searching with **deciderPoint** also takes $O(n^2 \log n)$ time as **deciderPoint** takes $O(n^2)$ time. By Lemma 5.6.11, with exponentially high probability the number of values from $\mathcal{Z}$ in the found interval is $O(n^2)$, and thus by Lemma 5.6.10 **sweep** takes $O(n^2 \log n)$ time with exponentially high probability. Finally we do another round of sorting and searching over the $O(n^2)$ values returned from **sweep**, which again takes $O(n^2 \log n)$ time. Adding the time to convert the result to a $(1+\varepsilon)$-approximation gives $O(n^2(\log n + \frac{1}{\varepsilon}\log\frac{1}{\varepsilon}))$ time overall. □

REFERENCES

[1] C. Elkan, "Using the triangle inequality to accelerate k-means," in *Proc. 20th International Conference on International Conference on Machine Learning (ICML)*, 2003, pp. 147–153.

[2] S. Bereg, M. Jiang, W. Wang, B. Yang, and B. Zhu, "Simplifying 3d polygonal chains under the discrete fréchet distance," in *Proc. 8th Latin American Symposium on Theoretical Informatics(LATIN)*, 2008, pp. 630–641.

[3] S. Giannarou and T. Stathaki, "Shape signature matching for object identification invariant to image transformations and occlusion," in *Proc. 12th International Conference on Computer Analysis of Images and Patterns (CAIP)*, 2007, pp. 710–717.

[4] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis, "Approximate algorithms for the traveling salesperson problem," in *15th Annual Symposium on Switching and Automata Theory (SWAT)*, 1974, pp. 33–42.

[5] J. Brickell, I. Dhillon, S. Sra, and J. Tropp, "The metric nearness problem," *SIAM J. Matrix Analysis Applications*, vol. 30, no. 1, pp. 375–396, 2008.

[6] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*, 2nd ed. McGraw-Hill Higher Education, 2001.

[7] T. Chan, "More algorithms for all-pairs shortest paths in weighted graphs," *SIAM Journal on Computing*, vol. 39, no. 5, pp. 2075–2089, 2010.

[8] N. Garg, V. Vazirani, and M. Yannakakis, "Approximate max-flow min-(multi)cut theorems and their applications," *SIAM J. Comput.*, vol. 25, no. 2, pp. 235–251, 1996.

[9] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem," Graduate School of Industrial Administration, Carnegie Mellon University, Tech. Rep. 388, 1976.

[10] H. Alt and L. J. Guibas, "Discrete geometric shapes: Matching, interpolation, and approximation: A survey," Handbook of Computational Geometry, Tech. Rep., 1996.

[11] H. Alt and M. Godau, "Computing the Fréchet distance between two polygonal curves," *Int. J. Comput. Geometry Appl.*, vol. 5, pp. 75–91, 1995.

[12] C. Fan, B. Raichel, and G. Van Buskirk, "Metric violation distance: Hardness and approximation," in *Proc. 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2018, pp. 196–209.

[13] C. Fan, B. Raichel, and G. V. Buskirk, "Metric violation distance: Revisited and extended," *CoRR*, vol. abs/1807.08078, 2018.

[14] C. Fan and B. Raichel, "Computing the fréchet gap distance," in *Proc. 33rd International Symposium on Computational Geometry (SoCG)*, 2017, pp. 42:1–42:16.

[15] S. Khot and O. Regev, "Vertex cover might be hard to approximate to within 2-epsilon," *J. Comput. Syst. Sci.*, vol. 74, no. 3, pp. 335–349, 2008.

[16] S. Chawla, R. Krauthgamer, R. Kumar, Y. Rabani, and D. Sivakumar, "On the hardness of approximating multicut and sparsest-cut," *Computational Complexity*, vol. 15, no. 2, pp. 94–114, 2006.

[17] I. Abraham, C. Gavoille, A. Gupta, O. Neiman, and K. Talwar, "Cops, robbers, and threatening skeletons: padded decomposition for minor-free graphs," in *Proc. 46th ACM Symposium on Theory of Computing (STOC)*, 2014, pp. 79–88.

[18] J. Chuzhoy and S. Khanna, "Polynomial flow-cut gaps and hardness of directed cut problems," *J. ACM*, vol. 56, no. 2, pp. 6:1–6:28, 2009.

[19] A. Agarwal, N. Alon, and M. Charikar, "Improved approximation for directed cut problems," in *Proc. 39th Annual ACM Symposium on Theory of Computing (STOC)*, 2007, pp. 671–680.

[20] G. Baier, T. Erlebach, A. Hall, E. Köhler, P. Kolman, O. Pangrác, H. Schilling, and M. Skutella, "Length-bounded cuts and flows," *ACM Trans. Algorithms*, vol. 7, no. 1, pp. 4:1–4:27, 2010.

[21] E. Lee, "Improved hardness for cut, interdiction, and firefighter problems," in *44th International Colloquium on Automata, Languages, and Programming (ICALP)*, 2017, pp. 92:1–92:14.

[22] A. Kuhnle, V. Crawford, and M. Thai, "Network resilience and the length-bounded multicut problem: Reaching the dynamic billion-scale with guarantees," in *Abstracts of the 2018 ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2018, pp. 81–83.

[23] A. C. Gilbert and L. Jain, "If it ain't broke, don't fix it: Sparse metric repair," in *Proc. 55th Annual Allerton Conference on Communication, Control, and Computing*, 2017, pp. 612–619.

[24] J. Bourgain, "On lipschitz embedding of finite metric spaces in hilbert space," *Israel Journal of Mathematics*, vol. 52, no. 1-2, pp. 46–52, 1985.

[25] N. Linial, E. London, and Y. Rabinovich, "The geometry of graphs and some of its algorithmic applications," *Combinatorica*, vol. 15, no. 2, pp. 215–245, 1995.

[26] P. Indyk and J. Matoušek, "Low-distortion embeddings of finite metric spaces," in *Handbook of Discrete and Computational Geometry*. CRC Press, 2004, pp. 177–196.

[27] J. Matoušek, *Lecture notes on metric embeddings*, 2013.

[28] A. Sidiropoulos, D. Wang, and Y. Wang, "Metric embeddings with outliers," in *Proc. 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2017, pp. 670–689.

[29] I. Abraham, Y. Bartal, T. Chan, K. Dhamdhere, A. Gupta, J. Kleinberg, O. Neiman, and A. Slivkins, "Metric embeddings with relaxed guarantees," in *Proc. 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2005, pp. 83–100.

[30] T. Chan, K. Dhamdhere, A. Gupta, J. Kleinberg, and A. Slivkins, "Metric embeddings with relaxed guarantees," *SIAM J. Comput.*, vol. 38, no. 6, pp. 2303–2329, 2009.

[31] E. Candès and B. Recht, "Exact matrix completion via convex optimization," *Commun. ACM*, vol. 55, no. 6, pp. 111–119, Jun. 2012.

[32] F. Chung, M. Garrett, R. Graham, and D. Shallcross, "Distance realization problems with applications to internet tomography," *J. Comput. Syst. Sci.*, vol. 63, no. 3, pp. 432–448, 2001.

[33] S. Khot, "On the power of unique 2-prover 1-round games," in *Proc. 34th Annual ACM Symposium on Theory of Computing (STOC)*, 2002, pp. 767–775.

[34] A. C. Gilbert and R. Sonthalia, "Generalized metric repair on graphs," *CoRR*, vol. abs/1807.07619, 2018.

[35] L. Valiant, "The complexity of enumeration and reliability problems," *SIAM J. Comput.*, vol. 8, no. 3, pp. 410–421, 1979.

[36] N. Alon and S. Gutner, "Balanced families of perfect hash functions and their applications," *ACM Trans. Algorithms*, vol. 6, no. 3, pp. 54:1–54:12, 2010.

[37] C. Brand, H. Dell, and T. Husfeldt, "Extensor-coding," in *Proc. 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, 2018, pp. 151–164.

[38] I. Abraham, S. Chechik, and S. Krinninger, "Fully dynamic all-pairs shortest paths with worst-case update-time revisited," in *Proc. Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2017, pp. 440–452.

[39] O. Filtser and M. Katz, "The discrete Fréchet distance gap," *arXiv:1506.04861*, 2015.

[40] K. Buchin, M. Buchin, W. Meulemans, and W. Mulzer, "Four soviets walk the dog - with an application to alt's conjecture," in *Proc. 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2014, pp. 1399–1413.

[41] P. Agarwal, R. Avraham, H. Kaplan, and M. Sharir, "Computing the discrete Fréchet distance in subquadratic time," *SIAM Journal on Computing*, vol. 43, no. 2, pp. 429–449, 2014.

[42] K. Bringmann, "Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless seth fails," in *Proc. 55th Annual Symp. on Found. of Comp. Sci. (FOCS)*. IEEE, 2014, pp. 661–670.

[43] A. Driemel, S. Har-Peled, and C. Wenk, "Approximating the Fréchet distance for realistic curves in near linear time," *Discrete & Computational Geometry*, vol. 48, no. 1, pp. 94–127, 2012.

[44] M. Kim, S. Kim, and M. Shin, "Optimization of subsequence matching under time warping in time-series databases," in *Proc. ACM Symp. on Applied Computing*, 2005, pp. 581–586.

[45] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk, "On map-matching vehicle tracking data," in *Proc. 31st VLDB Conference*, 2005, pp. 853–864.

[46] C. Wenk, R. Salas, and D. Pfoser, "Addressing the need for map-matching speed: Localizing global curve-matching algorithms," in *Sci. Statis. Database Manag.*, 2006, pp. 879–888.

[47] J. Serrà, E. Gómez, P. Herrera, and X. Serra, "Chroma binary similarity and local alignment applied to cover song identifica," *Audio, Speech & Lang. Proc.*, vol. 16, no. 6, pp. 1138–1151, 2008.

[48] K. Buchin, M. Buchin, J. Gudmundsson, M. Löffler, and J. Luo, "Detecting commuting patterns by clustering subtrajectories," *Int. J. Comput. Geom. Appl.*, vol. 21, no. 3, pp. 253–282, 2011.

[49] Y. K. Cheung and O. Daescu, "Fréchet distance problems in weighted regions," in *Proc. 20th International Symposium on Algorithms and Computation (ISAAC)*, 2009, pp. 97–111.

[50] A. Driemel and S. Har-Peled, "Jaywalking your dog: computing the Fréchet distance with shortcuts," *SIAM Journal on Computing*, vol. 42, no. 5, pp. 1830–1866, 2013.

[51] M. Buchin, A. Driemel, and B. Speckmann, "Computing the Fréchet distance with shortcuts is np-hard," in *Proc. 30th Annu. Sympos. Comput. Geom. (SoCG)*, 2014, p. 367.

[52] T. Eiter and H. Mannila, "Computing discrete Fréchet distance," Tech. Rep., 1994.

[53] R. Avraham, O. Filtser, H. Kaplan, M. Katz, and M. Sharir, "The discrete and semicontinuous Fréchet distance with shortcuts via approximate distance counting and selection," *ACM Trans. Algorithms*, vol. 11, no. 4, p. 29, 2015.

[54] H. Alt, C. Knauer, and C. Wenk, "Matching polygonal curves with respect to the Fréchet distance," in *Proc. 18th Annu. Symp. on Theo. Aspects of Comp. Sci. (STACS)*, 2001, pp. 63–74.

[55] R. Avraham, H. Kaplan, and M. Sharir, "A faster algorithm for the discrete Fréchet distance under translation," *CoRR*, vol. abs/1501.03724, 2015.

[56] H. Alt, A. Efrat, G. Rote, and C. Wenk, "Matching planar maps," in *Proc. 14th Annual ACM-SIAM symposium on Discrete algorithms (SODA)*, 2003, pp. 589–598.

[57] G. Rote, "Computing the Fréchet distance between piecewise smooth curves," *Computational Geometry*, vol. 37, no. 3, pp. 162–174, 2007.

[58] K. Buchin, M. Buchin, and C. Wenk, "Computing the Fréchet distance between simple polygons in polynomial time," in *22nd Annu. Sympos. Comput. Geom.*, 2006, pp. 80–87.

[59] H. Alt and M. Buchin, "Can we compute the similarity between surfaces?" *Discrete & Computational Geometry*, vol. 43, no. 1, pp. 78–99, 2010.

[60] S. Har-Peled and B. Raichel, "The Fréchet distance revisited and extended," *ACM Transactions on Algorithms (TALG)*, vol. 10, no. 1, p. 3, 2014.

[61] J. Bentley and T. Ottmann, "Algorithms for reporting and counting geometric intersections," *IEEE Trans. Computers*, vol. 28, no. 9, pp. 643–647, 1979.

BIOGRAPHICAL SKETCH

Chenglin Fan joined the PhD program in Computer Science at The University of Texas at Dallas in August 2016. He worked as a software developer in Shenzhen Institutes of Advanced Technology from 2011 to 2014. Chenglin Fan graduated with a master's degree in computer science and technology from Central South University in 2011, and a bachelor's degree in information and computer science from the University of South China in 2008.

CURRICULUM VITAE

Chenglin Fan

Contact Information:

Department of Computer Science
The University of Texas at Dallas
800 W. Campbell Rd.
Richardson, TX 75080, U.S.A.

Email: cxf160130@utdallas.edu

Educational History:

08/2016-Now	PhD student in Computer Science. The University of Texas at Dallas, Richardson, U.S.A. Advisor: Benjamin Raichel
08/2014 -06/2016	PhD student in Computer Science (transfer out). Montana State University, Bozeman, U.S.A.
06/2011	ME in Computer Science and Technology. Central South University, Changsha, China
07/2008	BS in Information and Computer Sciences, University of South China, China

Research Interests:

Design and Analysis of Algorithms, Computational Geometry

Employment History:

Software Developer, Shenzhen Institutes of Advanced Technology, July 2011 – July 2014

Publications in the Dissertation:

1. "Computing the Fréchet Gap Distance". **Chenglin Fan**, Benjamin Raichel. Proceedings of the 33rd Annual Symposium on Computational Geometry (**SoCG**), Pages 42:1-42:16, 2017.

2. "Metric Violation Distance: Hardness and Approximation". **Chenglin Fan**, Benjamin Raichel, Gregory Van Buskirk. Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (**SODA**), Pages 196-209, 2018.