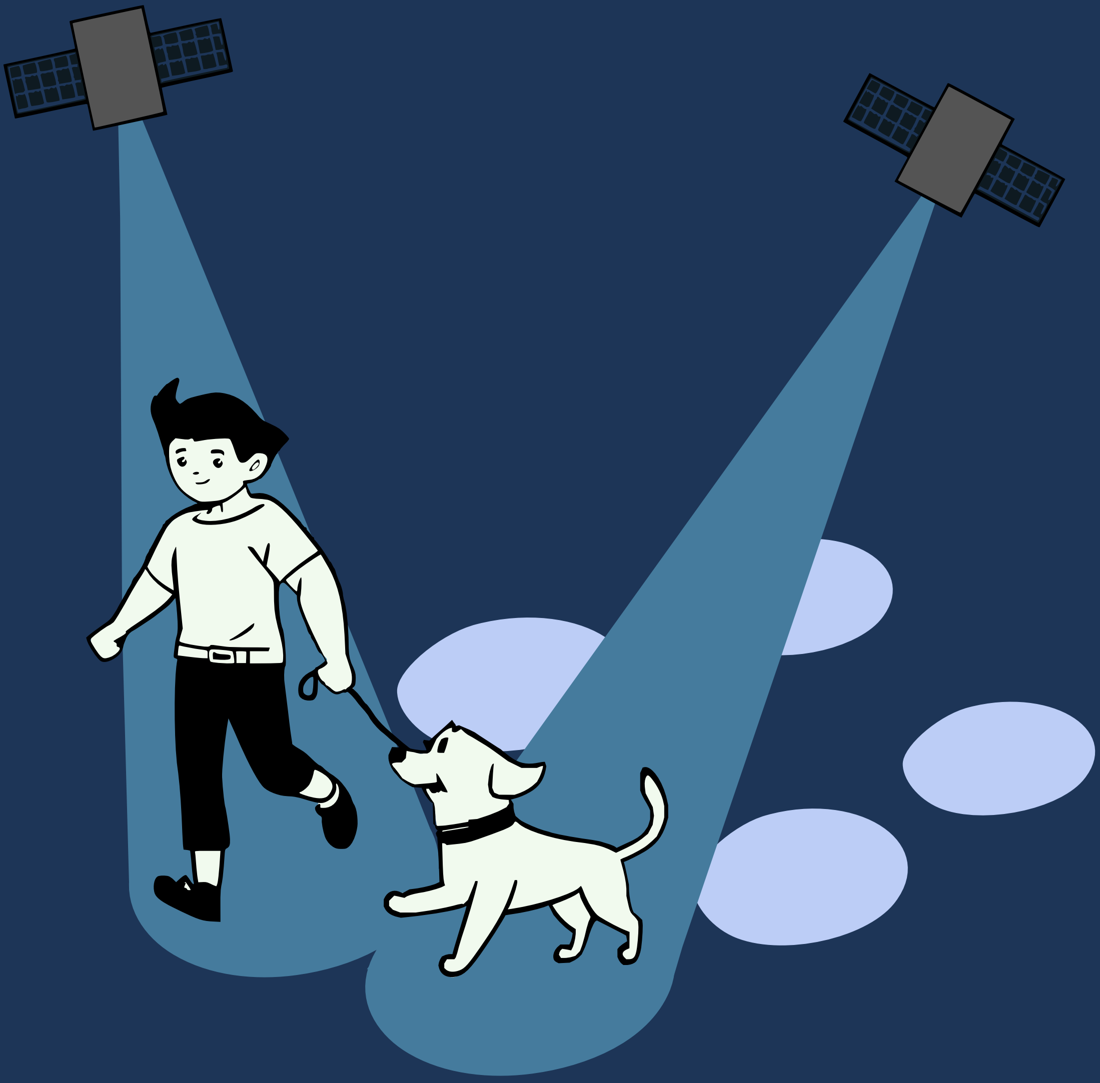
Algorithms for Imprecise Trajectories



Aleksandr Popov

Algorithms for Imprecise Trajectories

Aleksandr Popov

The work in this thesis is supported by the Dutch Research Council (NWO) under project no. 612.001.801.

A catalogue record is available from the Eindhoven University of Technology library.
ISBN: 978-90-386-5841-4.

Printed by Ipskamp Printing, Enschede.

This thesis has been typeset in LaTeX. The body text is typeset using TeX Gyre Pagella, an enhanced version of URW Palladio, based on Hermann Zapf's Palatino. The headings and the captions are typeset using Lucida Sans, designed by Chuck Bigelow and Kris Holmes and licensed through the TeX Users Group. The figures are drawn using Ipe and Ti*k*Z. The cover is made using Inkscape.

Thank you to Mariana Souza for help in making the cover.

©2023 by Aleksandr Popov. All rights are reserved.

Algorithms for Imprecise Trajectories

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
rector magnificus prof. dr. S.K. Lenaerts, voor een
commissie aangewezen door het College voor Promoties,
in het openbaar te verdedigen
op donderdag 12 oktober 2023 om 16:00 uur

door

Aleksandr Andreevich Popov

geboren te Sint-Petersburg, Rusland

Dit proefschrift is goedgekeurd door de promotoren en de samenstelling van de promotiecommissie is als volgt:

voorzitter:	prof. dr. J.J. Lukkien
1e promotor:	prof. dr. K.A. Buchin (TU Dortmund)
2e promotor:	prof. dr. B. Speckmann
copromotor:	dr. ir. M.J.M. Roeloffzen
leden:	prof. dr. M.T. de Berg
	prof. dr. C. Knauer (Universität Bayreuth)
	prof. dr. W.J.H. Mulzer (Freie Universität Berlin)
adviseur:	dr. M. Löffler (Universiteit Utrecht)

Het onderzoek of ontwerp dat in dit proefschrift wordt beschreven is uitgevoerd in overeenstemming met de TU/e Gedragscode Wetenschapsbeoefening.

Contents

Contents **v**

Acknowledgements **vii**

1 Introduction **1**
1.1 Uncertainty 6
1.2 Trajectory Analysis and Processing 12
1.3 Trajectory Analysis Under Uncertainty 19
1.4 Contributions 20

2 Preliminaries **28**
2.1 Curves 28
2.2 Uncertainty 29
2.3 Fréchet Distance 30
2.4 Uncertain Curves and Distances 33

3 Similarity of Uncertain Curves in 2D **35**
3.1 Hardness Results 38
3.2 Algorithms for Lower Bound Fréchet Distance 70
3.3 Algorithms for Upper Bound and Expected Fréchet Distance 85
3.4 Conclusions 97

4 Similarity of Uncertain Curves in 1D **98**
4.1 Lower Bound Fréchet Distance: General Approach 99
4.2 Lower Bound Fréchet Distance: One Dimension 105
4.3 Upper Bound Fréchet Distance 117
4.4 Weak Fréchet Distance 122
4.5 Conclusions 131

5 Uncertain Curve Simplification 134
5.1 Overview of the Approach . . . 137
5.2 Shortcut Testing: Intermediate Points . . . 138
5.3 Shortcut Testing: All Points . . . 151
5.4 Combining Steps . . . 162
5.5 Conclusions . . . 164

6 Map-Matching Queries under Fréchet Distance on Low-Density Spanners 166
6.1 Straight Path Queries . . . 171
6.2 Map-Matching Segment Queries . . . 176
6.3 General Map-Matching Queries . . . 182
6.4 Conclusions . . . 186

7 Segment Visibility Counting Queries in Polygons 187
7.1 Preliminaries . . . 191
7.2 Point Queries . . . 198
7.3 Segment Queries . . . 202
7.4 Segment Query for a Set of Segments . . . 213
7.5 Extensions, Discussion, and Future Work . . . 216

8 Conclusions 226
8.1 Discussion of Results . . . 226
8.2 Future Work . . . 228

Bibliography 230

Summary 252

Curriculum Vitae 254

Acknowledgements

Four years ago, I embarked on my PhD journey. When Kevin suggested to do a master's project and a PhD in computational geometry, it was an easy decision for me—I had enjoyed all the courses from ALGO, and I knew it would be fulfilling work. Though this journey came with unexpected twists and turns, I look at it with satisfaction. I would like to thank the people that have made it possible.

First of all, I would like to thank Kevin for being a great supervisor, always encouraging and finding the time to provide insight into any problem. I would also like to thank Marcel for his thoroughness and continuous support with drafts, questions, and doubts throughout my project. I would like to thank Maarten for the infrequent but insightful contributions, and for the opportunity to go to Dagstuhl. Finally, I would like to thank Bettina for all her help and guidance at the final stages of my PhD.

I would also like to thank the members of the committee, Mark de Berg, Christian Knauer, and Wolfgang Mulzer, for taking the time to review my thesis. I hope that it was a pleasant read.

During my PhD, I shared an office with some wonderful people. Pre-COVID, I was fortunate to be seated with Bram, Dáni, and Huib, from whom I learnt a lot about doing a PhD and about a wide array of topics in algorithms. Later, I shared the cool office with Bram, Max, Pantea, and Thijs, from whom I learnt a lot about all sorts of topics outside of our research. My gratitude goes to all of them, for all the good times and fun discussions.

In the last year of my PhD, I also got to go on a research visit to the University of Sydney. Thank you to Joachim for hosting, thank you to André, Antonia, Joachim, Kevin, Maike, and Sampson for exciting

research discussions, and thank you to Andrew, Anushka, Lindsey, Omid, Sampson, and Zijin for being so friendly and hospitable, making me feel right at home. Finally, thanks to Antonia, Lea, and Lukas for exploring Sydney and the Blue Mountains with me.

I am grateful to the current and the former members of ALGO for many fun and insightful discussions during lunches, board game evenings, and workshops, Andrés, Arpan, Bart, Bettina, Bram, Céline, Dániel, Henk, Huib, Irene, Irina, Jari, Jules, Kevin Buchin, Kevin Verbeek, Leonidas, Leonie, Leyla, Marcel, Mark, Martijn, Max van Mulken, Max Sondag, Michał, Morteza, Nathan, Pantea, Ruben, Shivesh, Thijs Beurskens, Thijs van der Horst, Tim, Tom, Willem, and Wouter. I would also like to thank the people I had the pleasure to collaborate with, André Nusser, André van Renssen, Antonia, Benjamin, Bram, Carolin, Chenglin, Frank, Ivor, Jérôme, Joachim, Kevin Buchin, Kevin Verbeek, Maarten, Maike, Marcel, Milutin, Sampson, and Tim.

Finally, I would like to thank a few people outside the university. Mark, I appreciate our discussions on algorithms, cybersecurity, and everything in between over the years. Kate, it has always been great to catch up whenever I visit; I hope to be able to do so again soon. Nikita, through university and beyond, your friendship has meant a lot to me. I'd like to thank my parents for all their support and advice, in recent years and before. Mari, I do not know where I'd be now without you. Thank you for your love and support for all these years, and looking forward to many more years together.

Eindhoven, September 2023

CHAPTER 1

Introduction

The world is in motion. No matter what scale you choose, from molecules to stars—nothing in the world stands still. We humans spend a significant part of our time moving: we commute to work, we go shopping, we go out, we exercise, we travel. Movement is an integral part of our existence and of the way we experience the world, and has been since the dawn of time (Figure 1.1).

As a species, we have always been interested in ourselves and the world around us. Ever since the technological advances of the twentieth century, we have been able to fuel our curiosity with massive amounts

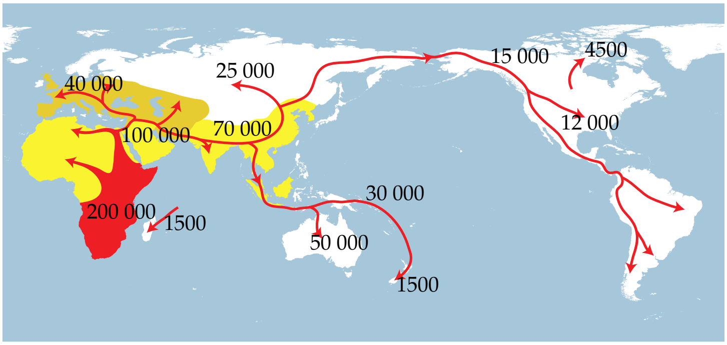


Figure 1.1. Prehistoric human migration patterns [185]. Dates indicate how many years ago the migration happened.

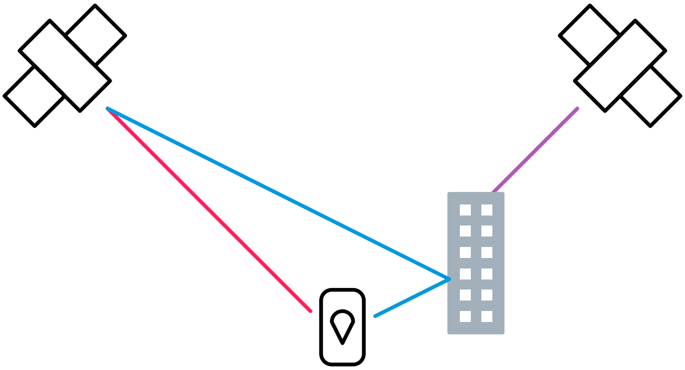
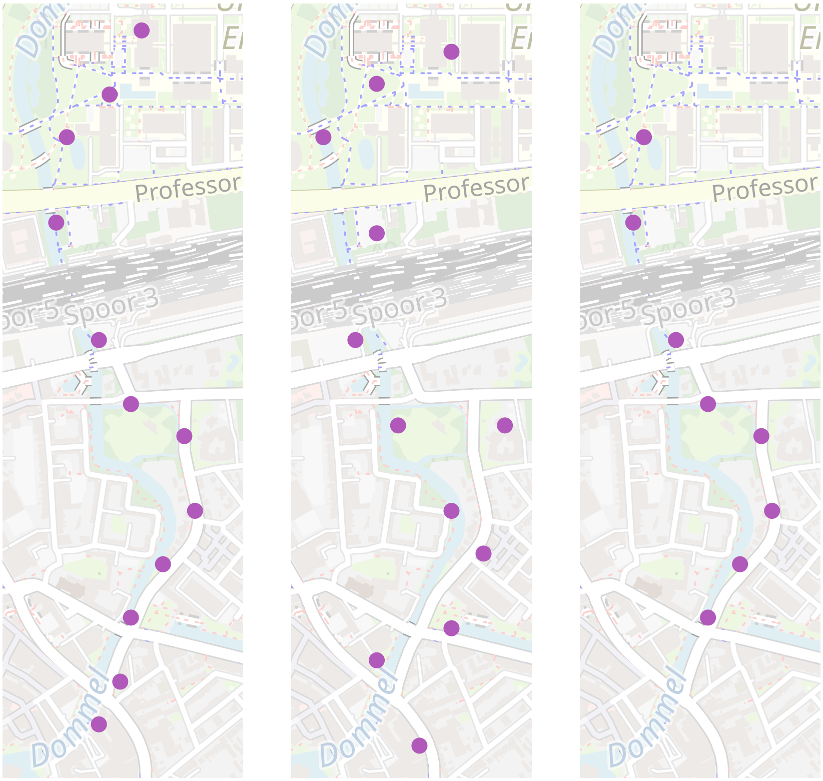


Figure 1.2. Multipath effects occur when the signal reflects, e.g. from a building, and reaches the receiver by two or more paths (red and blue in this case). This leads to difficulties in calculating the distance to the satellite, and results in positioning errors. From other directions, the (purple) signal may be entirely obstructed.

of data that we collect and process. Among other types of data, we collect *trajectory data* capturing movement between locations.

The massive amount of collected data means that we need to analyse the data automatically, to let humans extract insight from various derived statistics and visualisations rather than raw data. Naturally, trajectory analysis has been an important area of research with countless contributions from different areas of science and engineering, motivated by many applications where data-driven approaches can improve the status quo. For example, trajectory analysis can improve our understanding of urban mobility [193], as well as guide policies on the matter [133]; or it can help us understand the mechanisms of interaction between animals [91]; or it can help improve performance in sport [129]; or it can even be used to detect unusual events based on deviations in mass movement characteristics [213].

Unfortunately, data collection and data analysis are complicated in practice. A common way to acquire location measurements is using a satellite navigation system, like GPS, which yields *imprecise* locations, with varying degree of imprecision depending on the conditions. Satellite navigation is based on measuring the time that it takes for signal to travel between several satellites and a receiver, so any obstructions to these signals, like in Figure 1.2, may result in positioning errors [197];



(a) From original data, we can identify the subject's workplace.

(b) Each data point is randomly perturbed, decreasing precision.

(c) Data from the sensitive areas has been removed, thus losing information.

Figure 1.3. Individual data points can be anonymised in a dataset by perturbing the points or by removing parts of the data.

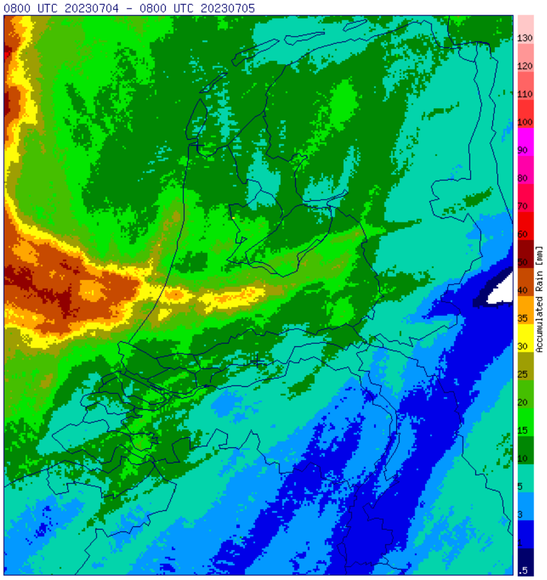
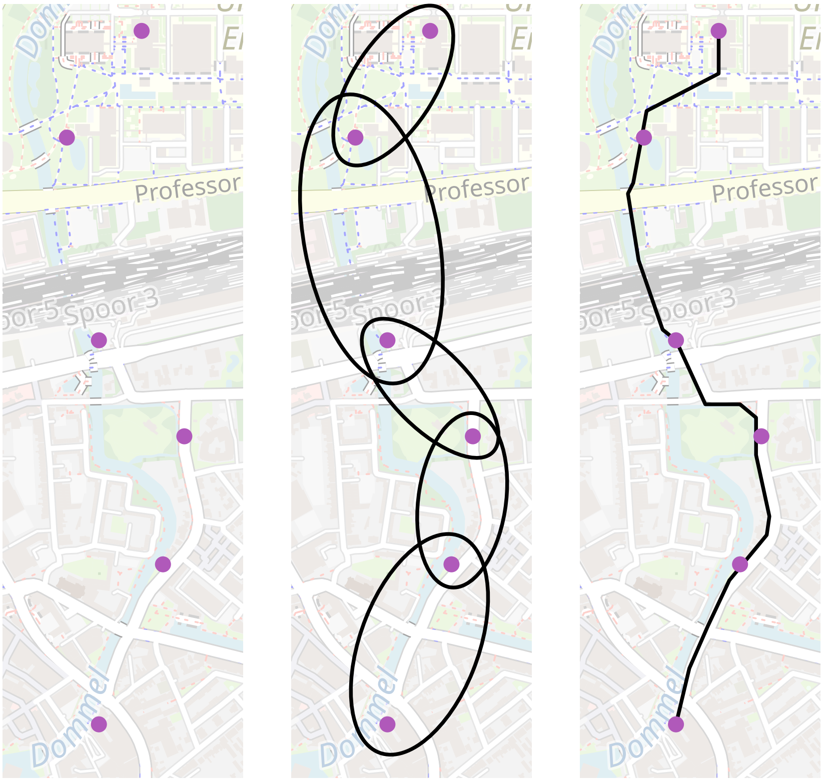


Figure 1.4. A storm path is inherently uncertain, because storm does not cover a single point and often does not have a sharp boundary. Here: storm development over the Netherlands on 5th July 2023. Image from KNMI [155].

furthermore, the geometry of the satellites with respect to the receiver plays an important role [196]. It is, however, possible to estimate the distortion [28] and to work further with explicitly modelled imprecision. In other applications, we may intentionally obstruct the exact recorded locations, like in Figure 1.3, so that we cannot tell if a specific person's data are included in the data set [113, 161, 182]. Depending on the national law, it may even be illegal to precisely track and record one's own location using satellite navigation [144]. Finally, sometimes the locations are the result of a forecast rather than observed events, or capture nebulous phenomena like storms, as in Figure 1.4, so there is inherent uncertainty in the (predicted) locations. In all these cases, there is some imprecision in the measured locations.

Furthermore, it is usually infeasible to record trajectory data continuously or even with extremely high frequency, if we are not using video cameras. Therefore, we only know the subject's (imprecise) locations



(a) Sparse GPS data. **(b)** Bounded speed model. **(c)** Using the map.

Figure 1.5. When the GPS data are sparse (a), we do not know where the subject was between the measurements. However, we can use physical models that bound the object's speed or acceleration to bound the locations possibly visited by the subject (b); or we can use other movement context, like a road network, to infer a likely path (c).

at the time of measurements, but do not have any information on their movement between measurements. Our trajectory data has gaps, and we usually need to fill them in some way, dealing with the uncertainty *between* measurements. See Figure 1.5 for example approaches.

In practically oriented areas, like geographic information science, the quality of the data has always been a concern, along with broader issues related to the way humans interact with their environment, which can collectively be viewed as uncertainty [115, 119]. In such areas, there is a large body of work that aims to develop robust mechanisms and tools that give useful results in the presence of uncertainty, and there are even some sophisticated models for uncertainty; however, these models are commonly used to preprocess the data, and are used less frequently in the analysis algorithms directly. In computational geometry, there is a significant amount of work on trajectory analysis and on uncertainty; given the amount of interest in these topics, there is surprisingly little work in their intersection.

Handling uncertainty explicitly by consciously choosing how to model it and by incorporating it into the algorithms allows us to get results that are resilient and transparent in terms of the quality of the analysis. In this thesis, we aim to do just that—develop algorithms for analysing uncertain trajectories.

1.1 Uncertainty

To get an accurate overview of the problems, we need to first consider the notion of uncertainty: where does it come from in trajectory data and how has it been dealt with, in computational geometry and beyond?

1.1.1 Uncertainty in Trajectory Data

Let us consider trajectory data acquisition. On the scale of humans and animals, the most common capture method is periodically recording measurements from a *global navigation satellite system (GNSS)*, such as GPS, GLONASS, Galileo, or BeiDou. Disregarding technical details that would better fit in a dedicated manuscript,[1] the principle used by these four satellite navigation systems is the same: all the satellites of a constellation have precisely synchronised atomic clocks, and each broadcasts a specific signal; the receiver needs to simultaneously get an unobstructed signal from at least four satellites. The broadcast positions of the satellites and the time offsets allow the receiver to compute their location in $\mathbb{R}^3$, with the origin at the Earth's centre [2] (here $\mathbb{R}$ denotes the set of real numbers). Measurements happen with certain intervals, so in the end, we obtain a sequence of locations paired with timestamps. We generally want to use a reference system that easily gives us a

---

[1] Readers interested in the workings of GNSS are referred to a book [197] on the topic.

position with respect to the surface of the Earth rather than its centre, so we convert these coordinates into latitude, longitude, and height above mean sea level, often also ignoring the latter. This is the form in which trajectory data is usually stored after acquiring it: a sequence of triplets of latitude, longitude, and timestamps.

In the described setting, we have to handle multiple types of uncertainty introduced earlier. Some sources of errors can be mitigated or are essentially negligible. However, other sources may give errors on the scale of at least tens of metres. For instance, atmospheric effects during e.g. storms can obstruct the GNSS signals [197, Sections 4.2, 5.4]. Neglecting clock corrections yields large positional errors, as well [197, Section 5.2]. Whenever the receiver is located close to reflective constructions, the reflected GNSS signals produce a *multipath,* thus giving the illusion of being further away from the receiver than they are and inducing an error [197, Section 4.2]. Finally, in urban environments, the issues arise in *canyons,* that is, rows of tall buildings flanking a relatively narrow street: in addition to multipath issues, the buildings make it more difficult to locate the receiver in the direction orthogonal to the street due to the lack of a direct line of sight to the relevant satellites [122].

Sometimes no GNSS-based location is available: for example, in tunnels, inside buildings, or under water. In some of these cases, there may still be alternative ways of locating the tracked object. *Wi-Fi positioning* can be very accurate [156], but requires the access points to be in a database so we know where they are located; setting up such a system requires a dedicated effort, so trajectory data rarely stems from these systems. Systems based on *GSM location* have significantly worse accuracy, generally not better than 50 metres [166, Table 2]. Previously, land-based radio beacon systems like Decca, Omega, Loran-C, Alpha, or Chayka were used for navigation, with comparatively large errors, but these are not relevant to readers that are not historically inclined.

As we have just discussed, for most used sources of trajectory data, we need to deal with uncertainty in the measurements. The other type of uncertainty we mentioned, uncertainty between measurements, arises naturally with all of these methods as we measure the location at discrete time steps rather than continuously. Depending on the requirements on the battery life in tracking devices, the measurements may intentionally be very sparse. The nature of trajectory data thus dictates the challenges related to handling uncertainty.

1.1.2 Uncertainty in Computational Geometry

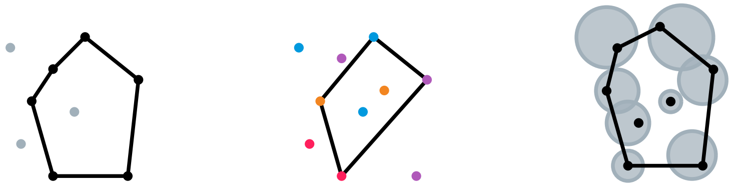
In computational geometry, there are well-known models that bridge the divide between the theoretical models of computation using real numbers and the real-life computers using bounded-precision floating-point numbers, implemented in state-of-the-art libraries like CGAL [210]. Indeed, some early inspiration for modelling data uncertainty stems from this line of research: Salesin, Stolfi, and Guibas [195] proposed a way to model imprecision that arises in computation, where they assume every point ends up somewhere in a disk of radius ε during the computation, but this approach has also been widely used to model data imprecision. Let us discuss the different proposed models.

Modelling uncertain points. We first focus on modelling measurement uncertainty. The models may have a different level of complexity, with the more complex ones representing the reality better; but they are also more difficult to work with, both conceptually and computationally.

Suppose we are working on a simple problem, like finding the convex hull of a point set in two dimensions. Intuitively, the convex hull is the smallest convex shape that contains all the points. If we are to work on a set of uncertain points, we have several choices to make when picking a model. (See also Figure 1.6.) One choice relates to the type of questions we want answered.

Probabilistic questions. Examples may be approximating a distribution of the number of points on the convex hull, or some other variable of interest; or computing expected values of these variables. Such questions require us to model the possible point locations with probability distributions, then derive the random variable of interest as a function of the probabilistic points. While a very flexible approach, it is also generally more difficult to use: it is sometimes difficult to propagate the distributions cleanly through the problem, and it may be computationally difficult to obtain the results.

Extremal questions. In this case, we do not make any statements about the likelihood of the point's location, but only describe the possible locations. The questions are mostly related to the largest or smallest values of certain parameters. For example, asking for the largest or smallest area of a convex hull of a set of uncertain points is



(a) Existential model. **(b)** Indecisive model. **(c)** Imprecise model.

Figure 1.6. We can pick any combination here to model uncertainty for the algorithmic questions about the convex hull of a point set. In the existential model, some points may be off. In the indecisive model, we pick a point of each colour. In the imprecise model, we pick a point in the region. All of these can be assigned a probability distribution.

an extremal question. To resolve these, we generally need some sharp bounds on the possible point locations, but further can treat the problem geometrically, without involving probabilities.

The second major decision is to pick the type of uncertainty regions that we use; this choice is orthogonal to the first one. We assume that we are operating in $\mathbb{R}^d$, that is, d-dimensional real coordinate space.

Existential model. Also referred to as 0/1 model, each uncertain point is either 'on' or 'off'. In other words, an uncertain point has one precise location, and we may choose whether to include it or not include it in the point set. This can be done in a way to compute the lower or upper bounds on some parameter (extremal questions); or each point may be assigned a probability of being 'on' (probabilistic questions). The motivation here is e.g. in radio transmission, where certain repeaters may be off and we still want to establish connectivity.

Indecisive model. In this model, each uncertain point is a finite set of possible discrete locations. For some questions, this model may be viewed as an extension of the existential model, although this is not true in general—there may be no way to express a point being off in this model. The problems in the indecisive model are also referred to as *colour-spanning* problems. Essentially, instead of having n uncertain points with k options per point, we have m points with n colours, and we need to select exactly one point of each colour while solving the problem. Sometimes we are allowed to select at most one point of every colour, which is equivalent to combining the indecisive and the existential models.

This model is used as a simplified version of complex probabilistic models. Suppose we model a point location with a distribution; instead of working with it directly, we can sample k points $p_i \in \mathbb{R}^d$, $i \in \{1, 2, \ldots, k\}$ from this distribution, then for sufficiently large k, the indecisive point $U = \{p_1, p_2, \ldots, p_k\}$ approximates the distribution. If all the samples are assigned equal probability, we can now approximate the solutions to the probabilistic questions in the original model with less algebraic complexity. If we ask extremal questions, we approximately decide whether something is true with high probability in the original model.

Imprecise model. In this model, each uncertain point is a compact connected region $U \subset \mathbb{R}^d$. Commonly used examples include balls in $\mathbb{R}^3$; or disks, line segments, and convex polygons in $\mathbb{R}^2$; or intervals in $\mathbb{R}$. The model with ε-disks [195] introduced earlier is a special case of this one.

We list some examples of work in computational geometry that uses these models outside the context of trajectory analysis. See Section 1.3 for an overview of the work on explicitly modelled uncertain trajectories.

When uncertainty in computational geometry started to get traction, the first set of systematic work has focused on basic geometric problems. In the existential model, there has been work on probabilistic queries for range searching, skylines, and nearest neighbours [215], as well as on separability [112]. In the imprecise model, the results mostly considered with the upper and the lower bounds for problems like finding the closest pair [31], the smallest bounding box, the smallest enclosing circle, and the minimum width strip on a set of imprecise points modelled as disks, squares, and line segments [171, 174]. In similar vein, there is work on the diameter [171, 174], perimeter of a polygon, and finding tours [172] on uncertain points. There is also an extensive set of early results on convex hulls [12, 158, 173, 208].

A bit more recently, similar geometric problems have been studied on indecisive points [149]. In this model, there is work on finding the smallest axis-aligned [3] and arbitrary [90] rectangles or strips that contain the points. More complex problems, like unit disk graph connectivity [107] or covering [25], have been studied in this model too.

Since then, many more geometric problem have been studied under uncertainty, including nearest-neighbour finding [6, 10], covering [17, 21], TSP [33, 84, 146], range queries [1, 4, 8, 9, 14, 134, 169], skylines [5, 143, 190], visibility in uncertain polygons [54, 68], Voronoi diagrams [200, 207], and clustering [150].

There are a few research directions that stand out otherwise. For instance, triangulations in general and Delaunay triangulations in particular have been considered in a preprocessing setting: we are given a set of uncertain points and want to construct a data structure for fast triangulation; at query time, we get a precise point set, with each precise point realising an uncertain one, and we want to compute the triangulation fast [55, 92, 159, 177]. In the same model, there are also results on sorting [142] and computing the Pareto front [143] and the onion decomposition [175]. The use of uncertainty in this model is quite different from the previously mentioned research.

Some of the aforementioned research uses the probabilistic model and not the extremal one [5, 33, 54, 61, 146, 176]. Problems on uncertain terrains [97, 120, 121] have been studied, too. Finally, there are efforts to visualise uncertain data and solutions to various problems under uncertainty [203, 214]. Over all, there is a very broad range of results on geometric computing with uncertainty.

Getting closer to problems on trajectories, there is research on moving points [62, 66, 104, 105, 106, 191, 202]. The uncertainty model is generally that we know the location of the point precisely when we query its location, and otherwise its uncertainty region grows. The main questions then concern the query strategies to keep the uncertainty under control. Nevertheless, the flavour is still quite different from trajectory analysis. In Section 1.3, we bridge the gap and discuss the research into trajectory analysis under uncertainty.

1.1.3 Uncertainty in Other Areas

Uncertainty is certainly a very broad and vague term, referring to different concepts in different areas, and it would be unfair to other disciplines to only describe the efforts made in computational geometry. The entirety of statistics, for instance, is built on uncertainty: we observe some data that may or may not be modelled by a given probability distribution; we can only be certain about this to a certain degree, expressed with the p-value of a statistical test. We accept a certain level of uncertainty, but we ensure that it is low enough before concluding our analysis. In statistics, uncertainty is a fundamental concept; if one were to ignore it, the field would not exist.

A similar attitude towards uncertainty is traced to machine learning through the Vapnik–Chervonenkis theory. When learning a classifier, we only use the training data, in the hopes that it represents a real underlying distribution. We can use statistics to guide us to minimise the empirical risk, that is, the likelihood of a mistake based on the data set. For instance, this approach is taken by support vector machines. We explicitly acknowledge the uncertainty in the area that does not contain any data points, and model it consciously to reduce its impact.

Uncertainty has also gotten a lot of attention in geographic information science and in movement ecology, specifically applied to trajectory data, as we discuss in Section 1.3.

1.2 Trajectory Analysis and Processing

Having discussed the concept of uncertainty, we now need to cover the traditional tasks and methods of trajectory analysis that do not explicitly model uncertainty. In general, a trajectory is a sequence of measurements, each including a location, a timestamp, and associated data. To simplify the setting, trajectories stored as discussed in Section 1.1.1 (latitude, longitude, optionally elevation) are usually projected from the geoid into the metric space on $\mathbb{R}^2$ or $\mathbb{R}^3$ equipped with Euclidean distance. Fortunately, there is usually a fitting projection that keeps subsequent analysis accurate.

The field of trajectory analysis is vast [91]. In the following overview, we focus on several important analysis tasks, namely, simplification (reduce the number of measured points while keeping most of the information), clustering (organise a set of trajectories into groups), and map matching (snap a noisy trajectory to a known map). (See Figure 1.7.) An observant reader will notice that all three tasks heavily rely on the notion of trajectory *similarity*, which should capture how closely two trajectories are related. We discuss similarity measures like the Hausdorff and the Fréchet distances in detail in Section 1.2.1.

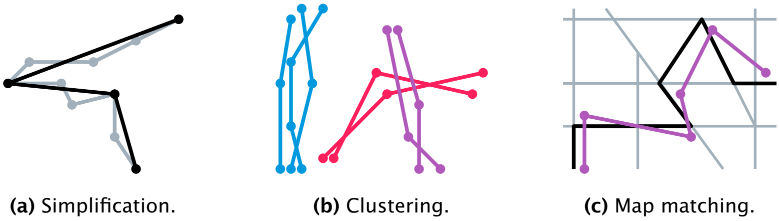

(a) Simplification. **(b)** Clustering. **(c)** Map matching.

Figure 1.7. A common preprocessing task is to simplify a trajectory while roughly preserving its shape. Other common tasks include clustering and map matching.

Simplification. Captured trajectory data are naturally quite complex and noisy. It may be useful to *simplify* the data for the purposes of conserving storage and speeding up analysis and visualisation, by reducing the number of stored data points without significantly distorting the trajectory (see Figure 1.7a). This ties in with the question of similarity: to make sure we do not distort the trajectory too much, one often imposes a threshold on some similarity measure between the original trajectory and its simplification.

There are two common ways to approach simplification: either the points of the simplified trajectory are a subsequence of the original trajectory, or they are placed freely. All of the work discussed here does not make special provisions for time, treating trajectories as sequences of locations. By far the most common approach is to select a subsequence of the input points, and to measure the deviation between two consecutive output points and the corresponding subtrajectory, rather than between the complete input and output curves.

A simple well-known heuristic is the Ramer–Douglas–Peucker algorithm [94, 192]. It implicitly uses the Hausdorff distance to bound the deviation from the input curve; the approach by Agarwal, Har-Peled, Mustafa, and Wang [11] uses the Fréchet distance instead to efficiently compute an approximate solution. Another commonly used algorithm is due to Imai and Iri [145]: this one provides guarantees on the quality of the output and can be adapted to use either the Hausdorff or the Fréchet distance. There are many related approaches [11, 29, 37, 53, 70, 123, 132, 181]. There is also some work on the approaches to simplification [152] that do not restrict the output to the input points, or that compute the distances on the complete trajectories [160].

Clustering. We often get a massive number of trajectories, difficult to visualise clearly and impossible to analyse manually. However, many of these may exhibit similar behaviour, so perhaps we do not need to analyse all trajectories separately, but rather study the behaviour of groups of trajectories. This problem is called *clustering:* group the set of entities into a small number of subsets, so that each subset contains similar entities (see Figure 1.7b). As you may surmise from the general phrasing, this problem can be stated on many types of entities, not only trajectories, as long as the entities live in a suitable metric space.

Indeed, a lot of work on clustering is done on point sets, and it is widely used in the machine learning community, among others. Prominent examples of clustering algorithms include many heuristics for the NP-hard problem of k-means clustering, as well as the related k-medians and k-medoids problems. The goal in each of these problems is to partition the input into k clusters, each with its own centroid. There are also density-based algorithms, such as DBSCAN [103], that consider clusters as areas of high density; and hierarchical clustering algorithms like single-linkage clustering and complete-linkage clustering that create a hierarchy of clusters bottom-up, merging the clusters closest together on every level of the hierarchy; the difference lies in how the distance between clusters is defined. All of these approaches could be easily adapted for use on trajectories, if each trajectory is represented as a point in an appropriate metric space.

However, there is also some work targeting trajectory clustering specifically, for example, work on k, ℓ-centre clustering, where we pick a centroid per cluster, like in the k-medians or k-means problem, and ensure that the centroid has complexity at most ℓ, i.e. that the trajectory has at most ℓ measurements [47, 48, 49, 98]. Finally, there is work tracking groups (clusters) of moving entities over time, which can be seen as clustering subtrajectories [44], and work on detecting patterns in groups of moving points [124].

Map matching. Very often, when working with trajectories, we know they move on a *network.* Cars generally drive on roads, aeroplanes follow air corridors, and trains are bound to the tracks—these are all examples of entities moving on networks. Even if the data we have collected is noisy, as long as we know that it comes from an entity moving on a network, we can handle the uncertainty in the best possible way: by decreasing its amount. In other words, even if a measurement is uncertain, we know it must have been on the network, and this knowledge allows us to make it significantly more precise. The *map-matching* problem is as follows: given a map (network) and a trajectory, both embedded in $\mathbb{R}^2$ or $\mathbb{R}^3$, find a path on the map closest to the trajectory (see Figure 1.7c). Map matching has received considerable attention, with multiple surveys comparing approaches on different types of data [19, 71, 138, 164, 212, 217]. Recently, there has also been a host of work using the Fréchet distance between the trajectory and the candidate paths on the map [22, 35, 69, 79, 80, 128, 201]. It is also possible to combine multiple sources of extra information, for instance, the map and some physical limitations on the movement (maximum acceleration, etc.) to decrease uncertainty even further [88]; and it is possible to connect to the points that are not directly on the network [26]. We can also look at a related problem of reconstructing a map based on a set of trajectories [16, 40, 89].

1.2.1 Trajectory Similarity

One of the most fundamental analysis tasks on trajectories is quantifying their similarity. A similarity measure that effectively captures the relevant differences between trajectories is essential to further analysis and processing tasks like simplification or map matching. Furthermore, we can plug a suitably defined similarity measure into general structures for e.g. nearest neighbour search or clustering, and thus elegantly reuse the work done on point sets.

There are many ways to group similarity measures. We list some common features to consider [205, 209]:

Absolute or relative time and space. Depending on the desired notion of similarity, we may want to treat trajectories as translation invariant, that is, we may want to ignore the absolute spatial separation between them. For example, if we are comparing handwritten signatures, we do not want to capture the fact that they were placed in different locations on the page. Similarly, we may want to treat time as relative rather than absolute if we are trying to compare paths that people take through a railway station: we do not necessarily care that one person was there two minutes later, but we do still care about their speeds. These concepts may sometimes be implemented during preprocessing, but other times they need to be incorporated into the similarity measure.

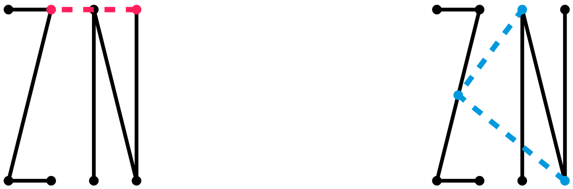
Incorporating time. Depending on the application, time may be treated as an extra regular dimension; as a special dimension; or ignored. Note that treating time as an extra dimension is similar to ignoring it: we just lift the points into one dimension higher. One simple measure that treats time in a special way is the *lock-step Euclidean distance* [209], which computes the total distance between time-aligned pairs of points; a more sophisticated example is the *Skorokhod distance* [180], which captures both the maximum distance between pairs of points under distorted time, and the needed time distortion. Most similarity measures used in computational geometry do not have special provisions for time.

Metric or non-metric. A *metric* on a set $\mathbb{D}$ is a function $d : \mathbb{D} \times \mathbb{D} \to \mathbb{R}$ satisfying three properties: $d(x, x) = 0$ for any $x \in \mathbb{D}$; symmetry, i.e. $d(x, y) = d(y, x)$ for any $x, y \in \mathbb{D}$; and triangle inequality, i.e. $d(x, y) \leq d(x, y) + d(y, z)$ for any $x, y, z \in \mathbb{D}$. It is worth considering whether a similarity measure is a metric—many similarity measures do not satisfy the triangle inequality, and some are not symmetric, both limiting the scope of their application.

Discrete or continuous. Some similarity measures are *continuous,* that is to say, they operate on the entire inferred path of an object; others are *discrete,* so they only take into account distances between measured points. Discrete measures usually perform poorly if the sampling is irregular; to apply continuous measures, we need to interpolate the path between the measurements in some way.

One similarity measure commonly used as a baseline is the *Hausdorff distance* [139, Kapitel VIII, § 6]. Defined for subsets of a metric space, it can be applied to trajectories by treating them as sets of measurements rather than sequences. Alternatively, with some method to fill in the gaps between measurements, it can be applied to entire paths treated as sets. It is a metric; but it does not capture the fact that trajectories are sequential, thus often giving unnatural results. (See Figure 1.8.)

There are several common similarity measures coming from time series analysis, loosely related to the edit distance. In the *edit distance on real sequence* [82], we traverse the trajectories only considering the measurements: we may skip the current measurement, adding one edit;



(a) Hausdorff distance.

(b) Fréchet distance.

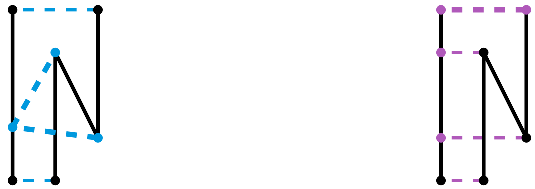
Figure 1.8. The Hausdorff distance gives unnatural results for trajectories compared to the Fréchet distance. In this example of dissimilar trajectories, the Hausdorff distance is much lower than the Fréchet distance.

or we may match it to the current point on the other trajectory, only adding one edit if the points are too far apart. This measure is robust to outliers, since skipping points does not incur a prohibitive cost, but it is a non-metric, because the triangle inequality does not hold.

The *longest common subsequence (LCSS)* [211], adapted to trajectories, is a related measure. We traverse the two trajectories and consider points equivalent if they are close enough to each other in space; the resulting value is the maximum number of equivalent points. Just like the edit distance on real sequence, LCSS is robust to outliers, but a non-metric.

Perhaps the most established similarity measure is the *dynamic time warping (DTW)* [163], seemingly stemming from speech recognition techniques. It is a discrete measure, and its main feature is enforced monotonicity: we traverse the trajectories from start to end, not going back and not skipping over measurements. Of all the ways to traverse the trajectories, we pick the one that minimises the sum of all pairwise distances. Unfortunately, the triangle inequality does not hold for DTW, so it is also a non-metric. There is also a continuous version of this measure called *continuous dynamic time warping* [36, 59] that captures trajectory similarity fairly naturally, but to date, there are no efficient algorithms for it in two dimensions or higher.

Finally, the *Fréchet distance* [114] is similar to DTW, but it instead minimises the largest distance, it is continuous, and it is a metric, making it by far the preferred similarity measure. It captures the sequential nature of trajectories naturally, and it is straightforward to compute in $O(mn \log mn)$ time on two trajectories with m and n measurements, respectively [23, 117]. Further research show that it can be computed slightly faster [45], and assuming the Strong Exponential



(a) Fréchet distance. **(b)** Weak Fréchet distance.

Figure 1.9. For the Fréchet distance and the weak Fréchet distance, we establish the matchings differently, capturing slightly different notions of similarity.

Time Hypothesis, it cannot be computed or even approximated well in strongly subquadratic time [39, 60]. However, for several restricted versions the Fréchet distance can be calculated faster, for example, for c-packed curves [96], when the edges between the measurements are long [125], or when the alignment of the curves is restricted [41, 179].

There is also a discrete variant, fittingly called the *discrete Fréchet distance* [7, 100], which can be computed in $O(mn^{\log\log n}/\log n)$ time, faster than the Fréchet distance, but may yield unexpected results when facing irregular sampling. Unless SETH fails, it also cannot be approximated well in strongly subquadratic time [39, 60].

A relaxation where one has to move continuously, but not necessarily monotonously, is called the *weak Fréchet distance* [23]. It can be computed in quadratic time [137], and cannot be approximated well in strongly subquadratic time unless SETH fails, except in 1D, where a linear-time algorithm exists [43, 60]. One can define the discrete weak Fréchet distance in a straightforward way, and it can be computed in quadratic time as well. See Figure 1.9.

There are many further variants, like the Fréchet distance with shortcuts [63, 87, 95], the generalised Fréchet distance [136], the Fréchet gap distance [108, 111], the k-Fréchet distance [24], or the translation-invariant Fréchet distance [38, 110]. The Fréchet distance and its variants have found many applications, not only in trajectory analysis, but also in the context of protein alignment [147] or handwriting recognition [216].

In this thesis, we focus on the Fréchet distance and its variants. The reason for that is that it is by far the most natural and useful similarity measure for trajectories that captures the natural notions of similarity well and is an easy to compute metric, allowing its efficient use in many trajectory analysis tasks like the ones listed earlier. We discuss the definitions and the standard algorithms in Section 2.3.

1.3 Trajectory Analysis Under Uncertainty

As we have seen, trajectory analysis is a broad area of research, and there has been a lot of interest in computing under uncertainty. Given the wealth of work on uncertain point sets in the models of Section 1.1.2, it is perhaps surprising that there is little comprehensive research into trajectory analysis using these models.

Measurement uncertainty. For the fundamental problem of trajectory similarity with uncertain measurements modelled as in Section 1.1.2, there has been little research prior to this thesis, and we are not even aware of any work in these models for other trajectory analysis tasks.

Knauer, Löffler, Scherfenberg, and Wolle [154] have considered the directed Hausdorff distance between imprecise point sets; it can be applied on trajectories, but Hausdorff distance in general does not capture the ordering of the measurements, and hence does not reflect trajectory similarity in a natural way, unlike the Fréchet distance.

Buchin and Sijben [65] have proposed an algorithm for computing the discrete Fréchet distance on uncertain points described by probability distributions. However, they consider a different kind of a problem—their goal is to determine a Fréchet alignment that maximises the probability that the distance is below a given threshold. In other words, the computed value is a lower bound for the probability under our definition, because some possible uncertain point locations may require a different alignment.

The discrete Fréchet distance has also been studied in the extremal model. Ahn, Knauer, Scherfenberg, Schlipf, and Vigneron [18] have solved the lower bound problem for the discrete Fréchet distance, where we aim to find the smallest discrete Fréchet distance over all the possible point locations; Fan and Zhu [109] have shown that the upper bound problem is NP-hard in some settings. We study multiple related problems in this thesis.

Uncertainty between measurements. There has been significantly more research into modelling the uncertainty between measurements.

The concept of space–time prisms, where the speed of the object between measurements is assumed to be bounded, has been applied to trajectory similarity [64] and to map matching [165]. In disciplines such as movement ecology and biology, there is a long tradition of using random walks, such as Brownian bridges, Ornstein–Uhlenbeck processes, Wiener processes, or Markov chains. There are many related models [157] that further build upon these stochastic processes to capture the relevant information; see a survey [86] for a more detailed overview. In geographic information science, there is also a substantial body of work [178, Chapter 5] on analysis in presence of noise [72, 183] and on interpolation [162], as well as other topics involving uncertainty [83].

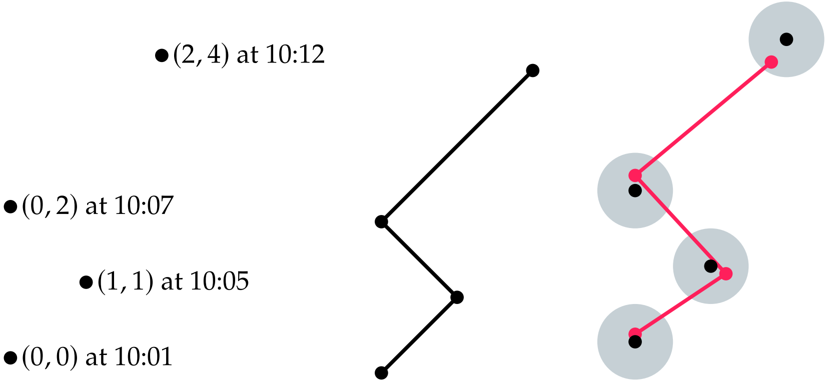
1.4 Contributions

We seek to expand the study of algorithms for trajectory analysis that model uncertainty. We next describe the contributions of this thesis. First, we discuss trajectory analysis with measurement uncertainty in Chapters 3 to 5. In particular, we first discuss the natural questions for the variants of the Fréchet distance in two dimensions in Chapter 3; as many turn out to be NP-hard, we study the one-dimensional case in Chapter 4. We then discuss trajectory simplification under uncertainty. Furthermore, we build upon the traditional view in Chapter 6 to utilise context in trajectory analysis without modelling uncertainty explicitly, by doing map matching; finally, we provide some contributions to visibility in simple polygons under uncertainty in Chapter 7.

In Chapters 3 to 6, we use the Fréchet distance as the base. We therefore do not make any special provisions for the time stamps. To make that clear, we talk about *uncertain curves* rather than uncertain trajectories. In this thesis, we adopt the distinction that a trajectory has associated time stamps, but a polygonal curve is simply a sequence of points in some space and does not treat time in any special way. We show an example in Figure 1.10.

Similarity of Uncertain Curves in 2D

In Chapter 3, we discuss the extremal questions on the (discrete) Fréchet distance under uncertainty. In particular, we consider the lower bound



(a) Trajectory data. **(b)** Polygonal curve. **(c)** Imprecise curve.

Figure 1.10. Captured trajectory data is a sequence of measurements with timestamps. By treating it as a sequence of locations without time having a special role, we get a polygonal curve. If we capture imprecision at each measurement, we get an imprecise curve, with a possible true curve shown in red.

and the upper bound (discrete) Fréchet distance, that is, the smallest and the largest Fréchet distance for any possible choice of point locations.

There are known algorithmic results for the lower bound discrete Fréchet distance [18], originally stated for disks as uncertainty regions, but easily extensible to line segments and to indecisive points. We study the upper bound Fréchet distance, both the discrete and the continuous variants, and find that it is NP-hard to decide whether it is above a given threshold in several models, namely, for indecisive points and for imprecise points modelled as line segments and as disks. We also study the lower bound Fréchet distance, showing that it is computable in polynomial time for indecisive points, but it is NP-hard to decide if it is below a threshold for imprecision modelled as vertical line segments.

We also consider a probabilistic question: assuming a uniform distribution over each uncertain point, what is the expected (discrete) Fréchet distance? It turns out to be #P-hard to answer this question for indecisive points and for imprecise points modelled using line segments. This set of hardness results is summarised in Table 1.1.

On the positive side, we present a FPTAS in constant dimension for computing the lower bound Fréchet distance when Δ/δ is polynomially bounded, where Δ is a bound on the diameter of an uncertainty region

Table 1.1. Hardness results for the decision problems in 2D. Ahn et al. [18] solve the lower bound problem for disks, but their algorithm extends to the indecisive curves as well as line-segment imprecision.

		indecisive	imprecise	
			disks	line segments
DFD	LB	polynomial [18]	polynomial [18]	polynomial [18]
	UB	NP-complete	NP-hard	NP-hard
	Exp	#P-hard	—	#P-hard
FD	LB	polynomial	—	NP-hard
	UB	NP-complete	NP-hard	NP-hard
	Exp	#P-hard	—	—

and δ is the lower bound Fréchet distance of interest. We also show a near-linear-time 3-approximation for the decision problem with uncertainty modelled as δ-separated convex regions, which can be turned into a 9-approximation for computing the lower bound Fréchet distance.

For the upper bound and the expected (discrete) Fréchet distance, we also present polynomial-time algorithms on indecisive curves with time bands, when the allowed Fréchet matchings are restricted.

Of the results described above, the author has worked on those pertaining to the upper bound and the expected (discrete) Fréchet distance, as well as on the algorithm for the lower bound Fréchet distance on indecisive curves. The results in this chapter are available in the following publications:

Kevin Buchin, Chenglin Fan, Maarten Löffler, Aleksandr Popov, Benjamin Raichel, and Marcel Roeloffzen. 'Fréchet Distance for Uncertain Curves'. In: *ACM Transactions on Algorithms* 19.3, 29 (2023). ISSN: 1549-6325. DOI: 10.1145/3597640.

Kevin Buchin, Chenglin Fan, Maarten Löffler, Aleksandr Popov, Benjamin Raichel, and Marcel Roeloffzen. 'Fréchet Distance for Uncertain Curves'. In: *Proceedings of the 47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*. Ed. by Artur Czumaj, Anuj Dawar, and Emanuela Merelli. Leibniz International Proceedings in Informatics 168. Wadern, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020, 20. ISBN: 978-3-95977-138-2. DOI: 10.4230/LIPIcs.ICALP.2020.20.

Table 1.2. Complexity results for the lower bound problems for uncertain curves in Chapter 4. The upper bound Fréchet distance remains NP-hard.

	Fréchet distance		weak Fréchet distance	
	discrete	continuous	discrete	continuous
1D	polynomial [18]	polynomial	NP-hard	polynomial
2D	polynomial [18]	NP-hard (Ch. 3)	NP-hard	NP-hard

Similarity of Uncertain Curves in 1D

In Chapter 4, we take the problems of the previous chapter to one dimension. Evidently, any setting where an efficient algorithm exists in two dimensions is still solvable in polynomial time in one dimension. For instance, the lower bound Fréchet distance with indecisive curves can be computed in polynomial time, as well as the lower bound discrete Fréchet distance with both indecisive and imprecise curves. In this chapter, we show that, unlike in two dimensions, the Fréchet distance on imprecise[2] curves can be computed in polynomial time.

We also formulate a general approach that leads to exponential-time algorithms in two dimensions and beyond, but may be helpful in further delineating the boundary of hardness.

We show that the upper bound (discrete) Fréchet distance remains NP-hard in one dimension both for indecisive and imprecise curves.

Finally, we study the lower bound weak Fréchet distance. We show that it can be computed in polynomial time with uncertainty modelled as intervals in 1D. In contrast, we present an NP-hardness argument for the lower bound discrete weak Fréchet distance both with indecisive and imprecise curves. This argument also implies that the continuous version becomes hard in two dimensions. See Table 1.2 for a summary.

The author has contributed to the results not pertaining to the weak Fréchet distance. The work in this chapter has been published in:

Kevin Buchin, Maarten Löffler, Tim Ophelders, Aleksandr Popov, Jérôme Urhausen, and Kevin Verbeek. 'Computing the Fréchet Distance between Uncertain Curves in One Dimension'. In: *Computational Geometry: Theory & Applications* 109, 101923 (2023). ISSN: 0925-7721. DOI: `10.1016/j.comgeo.2022.101923`.

---

[2] In one dimension, the only reasonable type of imprecision region is an interval.

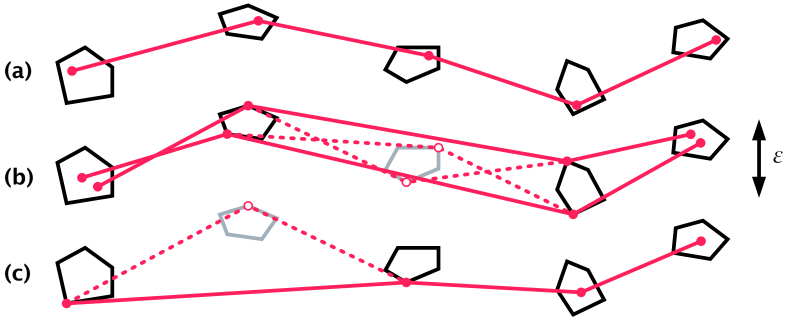


Figure 1.11. **(a)** An uncertain curve modelled with convex polygons and a potential realisation. **(b)** A valid simplification under the Hausdorff distance with the threshold ε: for every realisation, the subsequence is within Hausdorff distance ε from the full sequence. **(c)** An invalid simplification under the Hausdorff distance with the threshold ε: there is a realisation for which the subsequence is not within Hausdorff distance ε from the full sequence.

Kevin Buchin, Maarten Löffler, Tim Ophelders, Aleksandr Popov, Jérôme Urhausen, and Kevin Verbeek. 'Computing the Fréchet Distance between Uncertain Curves in One Dimension'. In: *Proceedings of the 17th International Symposium on Algorithms and Data Structures (WADS 2021)*. Ed. by Anna Lubiw, Mohammad Salavatipour, and Meng He. Lecture Notes in Computer Science 12808. Berlin, Germany: Springer, 2021, pp. 243–257. ISBN: 978-3-030-83507-1. DOI: `10.1007/978-3-030-83508-8_18`.

Uncertain Curve Simplification

As we discussed earlier, trajectory simplification is an important processing step: we may want to reduce storage or improve computation times of further analyses while keeping the trajectory close to the original data. In Chapter 5, we aim to define what this means under uncertainty. The particular problem we choose is as follows: find the shortest subsequence of uncertain points so that no matter what the true location of each uncertain point is, the resulting polygonal curve is a valid simplification of the original polygonal curve under the Hausdorff or the Fréchet distance. We consider this problem with indecisive points as well as imprecise points modelled as disks, line segments, and convex polygons. In all the settings, we present polynomial-time algorithms for this problem. See Figure 1.11.

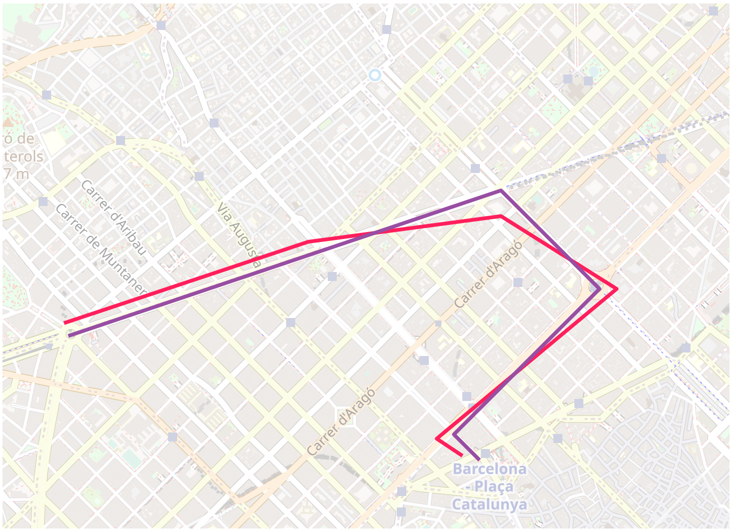


Figure 1.12. Given a map (of Barcelona), we can preprocess it so that, given a query red trajectory, we efficiently compute the purple path of the map. Map data from OpenStreetMap [188].

The results of Chapter 5 have been published as:

Kevin Buchin, Maarten Löffler, Aleksandr Popov, and Marcel Roeloffzen. 'Uncertain Curve Simplification'. In: *Proceedings of the 46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021)*. Ed. by Filippo Bonchi and Simon J. Puglisi. Leibniz International Proceedings in Informatics 202. Wadern, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, 26. ISBN: 978-3-95977-201-3. DOI: `10.4230/LIPIcs.MFCS.2021.26`.

Map-Matching Queries on Low-Density Spanners

In Chapter 6, we take a more traditional approach and study the map-matching problem on a realistic road network. (See Figure 1.12.) We show an efficient algorithm that matches an arbitrary noisy trajectory to a known realistic map. There are somewhat impractical conditional lower bounds on this problem, even if the map is known in advance. We circumvent them by imposing practical restrictions on the map that are satisfied by most real-world maps. We further ensure that the map complexity does not slow down the analysis by preprocessing the map.

The results of Chapter 6 are also available as:

Kevin Buchin, Maike Buchin, Joachim Gudmundsson, Aleksandr Popov, and Sampson Wong. *Map-Matching Queries under Fréchet Distance on Low-Density Spanners*. Presented at EuroCG 2023, Barcelona, Spain. 2023. URL: `https://dccg.upc.edu/eurocg23/wp-content/uploads/2023/05/Booklet_EuroCG2023.pdf` (visited on 01/07/2023).

Segment Visibility Counting Queries in Polygons

In the presence of uncertainty, a typically interesting question is to determine if two subjects met. Extending the concept leads us to visibility questions: how likely is it that two objects saw each other? We study a related problem in simple polygons. Two objects see each other if and only if there is a line of sight between them unobstructed by the polygon boundary. Ideally, we would solve this problem in presence of measurement uncertainty, with some probability distribution on the regions. Instead, we can sample the probability distributions, getting indecisive points. If every point has k options, we can consider all k^2 line segments connecting two consecutive points. We can even incorporate uncertainty between measurements by adding intermediate locations. If we can count the pairs of such line segments on which the objects see each other, then we can approximate the probability of them seeing each other. In Chapter 7, we make an important step in this direction by showing how to efficiently count the pairs of line segments that are weakly visible to each other. See Figure 1.13.

The results of Chapter 7 have been published as:

Kevin Buchin, Bram Custers, Ivor van der Hoog, Maarten Löffler, Aleksandr Popov, Marcel Roeloffzen, and Frank Staals. 'Segment Visibility Counting Queries in Polygons'. In: *Proceedings of the 33rd International Symposium on Algorithms and Computation (ISAAC 2022)*. Ed. by Sang Won Bae and Heejin Park. Leibniz International Proceedings in Informatics 248. Wadern, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 58. ISBN: 978-3-95977-258-7. DOI: `10.4230/LIPIcs.ISAAC.2022.58`.

Other Results

In addition to the results presented in this thesis, the author has worked on centre-based clustering with continuous dynamic time warping [36] and on oriented spanners [52].

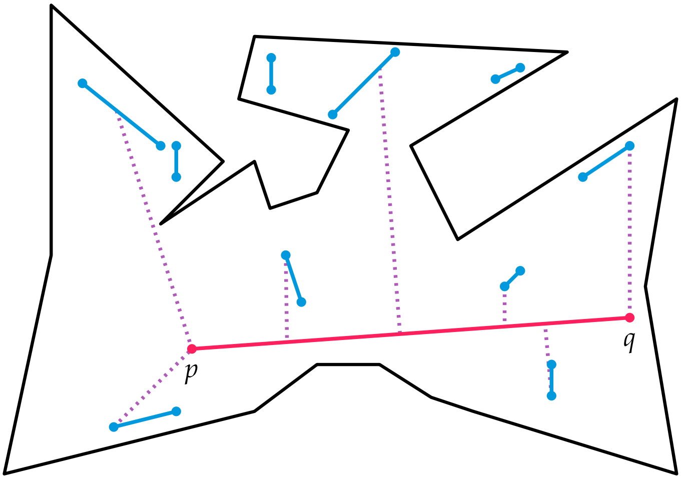


Figure 1.13. We preprocess a simple polygon and the blue line segments so that we can efficiently count how many are weakly visible to the red query segment.

CHAPTER 2

Preliminaries

In this chapter, we introduce the notation and the definitions relevant to the rest of this thesis, as well as recall the definitions and computational approaches to the (discrete) Fréchet distance.

2.1 Curves

Denote a *sequence* of points in $\mathbb{R}^d$ by $\pi = \langle p_1, \ldots, p_n \rangle$. For only two points $p, q \in \mathbb{R}^d$, we also use pq instead of $\langle p, q \rangle$. This notation can also be used if we interpret π as a *polygonal curve* on n vertices (of *length* n). It is defined by linearly interpolating between the successive points in the sequence and can be seen as a continuous function, for $i \in [n-1]$ and $\alpha \in [0, 1]$:

$$\pi(i + \alpha) = (1 - \alpha)p_i + \alpha p_{i+1} .$$

Here we use the notation[1] $[n] \overset{\text{def}}{=} \{1, 2, \ldots, n\}$ for any $n \in \mathbb{N}$. Denote a subsequence of a sequence π from index i to j with $\pi[i : j] = \langle p_i, p_{i+1}, \ldots, p_j \rangle$. Occasionally we use the notation $\langle \pi(i) \mid i \in I \rangle_{i=1}^{n}$ to denote a curve built on a subsequence of vertices of π, where vertices are only taken if their indices are in set I. For example, setting $I =$

---

[1] Where relevant, we use := and =: to denote assignment, $\overset{\text{def}}{=}$ for equivalent quantities in definitions or to point out equality by earlier definition, and = in other contexts. We also use ≡, but we introduce its usage as needed.

$\{1, 3, 4\}$, $n = 5$, $\pi = \langle p_1, p_2, \ldots, p_5\rangle$ means $\langle \pi(i) \mid i \in I\rangle_{i=1}^{n} = \langle p_1, p_3, p_4\rangle$. Where we deem it important to distinguish between points that are a part of the curve and other points, we denote the polygonal curve by $\pi = \langle \pi_1, \pi_2, \ldots, \pi_n\rangle$. We denote the *concatenation* of two sequences π and σ by $\pi \sqcup \sigma$; this also naturally defines concatenation of polygonal curves. Finally, $p \sqcup q$ (or simply pq) denotes the line segment between points p and q. We can generalise this notation:

$$\bigsqcup_{i \in [n]} p_i \overset{\text{def}}{=} p_1 \sqcup p_2 \sqcup \ldots \sqcup p_n = \langle p_1, p_2, \ldots, p_n\rangle \equiv \pi\,.$$

We also introduce the notation for the order of points along a curve. Let $p := \pi(a)$ and $q := \pi(b)$ for some $a, b \in [1, n]$. Then $p \prec q$ iff $a < b$, $p \preccurlyeq q$ iff $a \leq b$, and $p \equiv q$ iff $a = b$. Note that we can have $p = q$ for $a \neq b$ if the curve intersects itself.

Given two points $p, q \in \mathbb{R}^d$, denote their Euclidean distance by $\|p - q\|$. Throughout the thesis, we treat the dimension d as a small constant. For two compact sets $X, Y \subset \mathbb{R}^d$, denote their distance by

$$\|X - Y\| \overset{\text{def}}{=} \min_{x \in X, y \in Y} \|x - y\|\,.$$

Finally, given points $p, q, r \in \mathbb{R}^d$, define the distance from p to the segment qr as $d(p, qr) \overset{\text{def}}{=} \min_{t \in qr} \|p - t\|$.

2.2 Uncertainty

An *uncertainty region* $U \subset \mathbb{R}^d$ describes a possible location of a true point: it has to be inside the region, but there is no information as to where exactly. A *realisation* p of such a point is one of the points from the region U. When needed we assume the realisations are drawn from U according to a known probability distribution $\mathbb{P}$. We denote the diameter of any compact set (e.g. an uncertain point) $U \subset \mathbb{R}^d$ by $\operatorname{diam}(U) \overset{\text{def}}{=} \max_{p,q \in U} \|p - q\|$.

We use several uncertainty models, so the regions U are of different shape. An *indecisive point* is a form of an uncertain point where the uncertainty region is represented as a discrete set of points, and the true point is one of them: $U = \{p^1, \ldots, p^k\}$, with $k \in \mathbb{N}$ and $p^i \in \mathbb{R}^d$ for all $i \in [k]$. *Imprecise points* are modelled with uncertainty regions that are compact connected sets. In particular, we consider *line segments*, *balls*, referred to as *disks* in two dimensions, and *polygonal closed convex sets.* We denote a line segment between $p, q \in \mathbb{R}^d$ with $S(p, q)$. We denote a ball with the centre $c \in \mathbb{R}^d$ and the radius $r \in \mathbb{R}^+$ as $D(c, r)$. Formally, $D(c, r) \overset{\text{def}}{=} \{p \in \mathbb{R}^d \mid \|p - c\| \leq r\}$. We define a *polygonal closed convex set (PCCS)* as a closed convex set with bounded area that can be described as the intersection of a *finite* number of closed half-spaces. Note that this definition includes both convex polygons and line segments (in 2D). Given a PCCS U, let $V(U)$ denote the set of vertices of U, i.e. the vertices of a convex polygon or the endpoints of a line segment.

2.3 Fréchet Distance

The *Fréchet distance* is often described through an analogy with a person and a dog walking along their respective curves without backtracking, where the Fréchet distance is the shortest leash needed for such a walk. Formally, consider a set of *reparametrisations* Φ_ℓ of length ℓ, defined as continuous non-decreasing surjective functions $\phi : [0, 1] \to [1, \ell]$. Given two polygonal curves π and σ of lengths m and n, respectively, and the corresponding reparametrisations $\phi_1 \in \Phi_m$ and $\phi_2 \in \Phi_n$, define

$$\underset{\phi_1, \phi_2}{\operatorname{cost}}(\pi, \sigma) = \max_{t \in [0,1]} \|\pi(\phi_1(t)) - \sigma(\phi_2(t))\| .$$

We call the pair $\mu = (\phi_1, \phi_2)$ an *alignment* or a *matching.*

The cost represents the maximum distance between two points traversing the curves from start to end according to ϕ_1 and ϕ_2 (which allow varying speed, but no backtracking). The *Fréchet distance* $d_{\mathrm{F}}(\pi, \sigma)$ is defined as the minimum possible cost over all such traversals:

$$\begin{aligned} d_{\mathrm{F}}(\pi, \sigma) &\overset{\text{def}}{=} \inf_{\phi_1 \in \Phi_m, \phi_2 \in \Phi_n} \underset{\phi_1, \phi_2}{\operatorname{cost}}(\pi, \sigma) \\ &= \inf_{\phi_1 \in \Phi_m, \phi_2 \in \Phi_n} \max_{t \in [0,1]} \|\pi(\phi_1(t)) - \sigma(\phi_2(t))\| . \end{aligned}$$

In the person–dog analogy for the Fréchet distance, the best choice of reparametrisations means that the person and the dog choose the best speed, and the leash length is then the largest needed leash length during the walk.

The *discrete Fréchet distance* $d_{\mathrm{dF}}(\pi, \sigma)$ is defined similarly, except that we do not traverse the edges of the curves, but jump from one vertex to

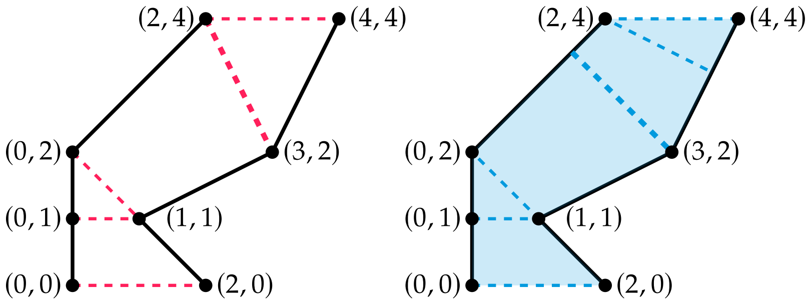


Figure 2.1. Left: Discrete Fréchet distance, with an optimal coupling shown in dashed red lines. Right: Fréchet distance, where the dashed blue lines indicate the specific values for an optimal alignment ϕ_1, ϕ_2.

the next on one or both curves. We define a valid *coupling* as a sequence $c = \langle (p_1, q_1), \ldots, (p_r, q_r) \rangle$ of pairs from $[m] \times [n]$ where $(p_1, q_1) = (1, 1)$, $(p_r, q_r) = (m, n)$, and, for any $i \in [r-1]$, we have

$$(p_{i+1}, q_{i+1}) \in \{(p_i + 1, q_i), (p_i, q_i + 1), (p_i + 1, q_i + 1)\} \,.$$

Let C be the set of all valid couplings on curves of lengths m and n; then

$$d_{\mathrm{dF}}(\pi, \sigma) \overset{\mathrm{def}}{=} \inf_{c \in C} \max_{s \in [|c|]} \|\pi(p_s) - \sigma(q_s)\| \,,$$

where $c_s = (p_s, q_s)$ for all $s \in [|c|]$. This variant can be seen as minimising the cost over all discrete matchings, restricted to vertices. Both distances are illustrated in Figure 2.1.

We also define the *weak Fréchet distance* $d_{\mathrm{wF}}(\pi, \sigma)$ as

$$d_{\mathrm{wF}}(\pi, \sigma) \overset{\mathrm{def}}{=} \inf_{\text{weak matching } \mu} \mathrm{cost}_{\mu}(\pi, \sigma) \,,$$

where the weak matching is not a pair of reparametrisations, but a path $(\alpha, \beta)\colon [0, 1]^2 \to [1, m] \times [1, n]$, with $\alpha(0) = 1$, $\alpha(1) = m$ and $\beta(0) = 1$, $\beta(1) = n$.

Computing the discrete Fréchet distance. We recall the standard dynamic programming approach by Eiter and Mannila [100]. The

Table 2.1. Left: Distance matrix on vertices for the example of Figure 2.1. Right: Dynamic program for the discrete Fréchet distance, filled from the bottom left corner. Rows correspond to points from the left trajectory, columns—to points from the right trajectory. The optimal path is marked in grey.

4	$\sqrt{10}$	$\sqrt{5}$	2
$2\sqrt{2}$	$\sqrt{2}$	3	$2\sqrt{5}$
$\sqrt{5}$	1	$\sqrt{10}$	5
2	$\sqrt{2}$	$\sqrt{13}$	$4\sqrt{2}$

4	$\sqrt{10}$	$\sqrt{5}$	$\sqrt{5}$
$2\sqrt{2}$	2	3	$2\sqrt{5}$
$\sqrt{5}$	2	$\sqrt{10}$	5
2	2	$\sqrt{13}$	$4\sqrt{2}$

algorithm is deduced in a standard manner from the following recursion:

$$\begin{aligned} d_{\mathrm{dF}}(\pi[1:i+1], \sigma[1:j+1]) = \max(&\|\pi(i+1) - \sigma(j+1)\|, \\ &\min(d_{\mathrm{dF}}(\pi[1:i], \sigma[1:j]), \\ &\quad d_{\mathrm{dF}}(\pi[1:i+1], \sigma[1:j]), \\ &\quad d_{\mathrm{dF}}(\pi[1:i], \sigma[1:j+1]))) \,. \end{aligned}$$

That is, the discrete Fréchet distance is the maximum of the distance of the newly added element in the coupling and the value that was considered best previously. Due to the coupling restrictions, there are only three possible subproblems that we need to consider, and we may choose the best of them, thus obtaining the recursion above. It is straightforward to turn it into a dynamic program.

Table 2.1 shows the distance matrix and the computation of the discrete Fréchet distance for the example of Figure 2.1. Each cell of the table on the right shows the value of the discrete Fréchet distance so far; the final result can be read out from the top right corner of the table, and the coupling that yields this result can be read from the sequence of grey cells. Notice that the table shows the same coupling as Figure 2.1.

Given two trajectories of lengths m and n in two dimensions, this approach takes $\Theta(mn)$ time. More recently, Agarwal et al. [7] presented an algorithm that computes the discrete Fréchet distance in time $O(mn \cdot {}^{\log\log n}/_{\log n})$ in two dimensions, for $m \leq n$. However, it is rather complex and does not help the intuition about the problems discussed in this thesis, so we will not go into further detail. The decision version of the problem can be solved in a similar fashion, but propagating boolean values instead.

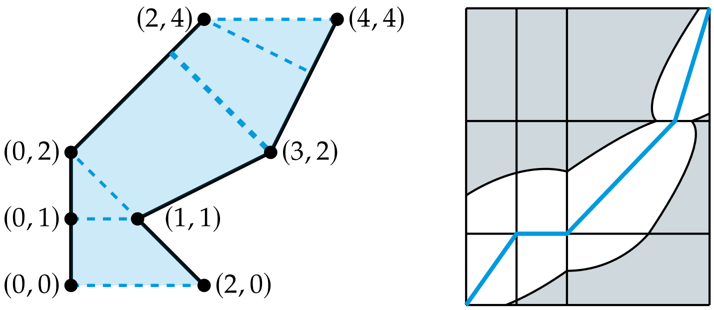


Figure 2.2. Left: Visualisation of the Fréchet distance. Right: Free-space diagram for the threshold $\varepsilon = 2.15$. One can draw a monotone path from the lower left corner to the upper right corner of the diagram, so the Fréchet distance between the trajectories is below the threshold.

Computing the Fréchet distance. One can use a similar approach to solve the decision version of the Fréchet distance problem with threshold δ, except now we have free and blocked areas within each cell of the table rather than simply having a boolean value in each cell. The resulting table is called a *free-space diagram.* It is a two-dimensional diagram on $[1, m] \times [1, n]$, where each point (x, y) corresponds to the pair $(\pi(x), \sigma(y))$. The point (x, y) is *free* if and only if $\|\pi(x) - \sigma(y)\| \leq \delta$. The free space is the collection of all the free points.

On polygonal curves, each cell becomes an intersection of an ellipse with the cell, with the inside of the ellipse being free. The answer to the problem is True if and only if there is a monotone path from the bottom left corner to the top right corner of the free-space diagram. A free-space diagram for the example of the two polygonal curves of Figure 2.1 is shown in Figure 2.2. Algorithmically this can be checked by keeping the open intervals on the edges of the cells, i.e. the white segments on the cell borders shown in Figure 2.2. The algorithm then runs in time $\Theta(mn)$. For further details, the reader is invited to consult the work by Alt and Godau [23, 117], or the work by Buchin et al. [45] that describes a slightly faster approach.

2.4 Uncertain Curves and Distances

We call a sequence of uncertainty regions an *uncertain curve:* $\mathcal{U} = \langle U_1, \ldots, U_n \rangle$. If we pick a point from each uncertainty region of $\mathcal{U}$, we get a polygonal curve π that we call a *realisation* of $\mathcal{U}$, denoting it with $\pi \Subset \mathcal{U}$. That is, if for some $n \in \mathbb{N}$ we have $\pi = \langle p_1, \ldots, p_n \rangle$ and $\mathcal{U} = \langle U_1, \ldots, U_n \rangle$, then $\pi \Subset \mathcal{U}$ if and only if $p_i \in U_i$ for all $i \in [n]$. We denote the set of all realisations of an uncertain curve $\mathcal{U}$ by $\operatorname{Real}(\mathcal{U})$. In a probabilistic setting, we write $\pi \Subset_{\mathbb{P}} \mathcal{U}$ to denote that each point of π gets drawn from the corresponding uncertainty region independently according to distribution $\mathbb{P}$.

For uncertain curves $\mathcal{U}$ and $\mathcal{V}$, define the upper bound, the lower bound, and the expected discrete Fréchet distance as

$$\begin{aligned}
d_{\mathrm{dF}}^{\max}(\mathcal{U}, \mathcal{V}) &= \max_{\pi \Subset \mathcal{U}, \sigma \Subset \mathcal{V}} d_{\mathrm{dF}}(\pi, \sigma)\,, \\
d_{\mathrm{dF}}^{\min}(\mathcal{U}, \mathcal{V}) &= \min_{\pi \Subset \mathcal{U}, \sigma \Subset \mathcal{V}} d_{\mathrm{dF}}(\pi, \sigma)\,, \\
d_{\mathrm{dF}}^{\mathbb{E}(\mathbb{P})}(\mathcal{U}, \mathcal{V}) &= \mathbb{E}_{\pi \Subset_{\mathbb{P}} \mathcal{U}, \sigma \Subset_{\mathbb{P}} \mathcal{V}}[d_{\mathrm{dF}}(\pi, \sigma)]\,.
\end{aligned}$$

We extend the definitions to the continuous Fréchet distance $d_{\mathrm{F}}^{\max}$, $d_{\mathrm{F}}^{\min}$, $d_{\mathrm{F}}^{\mathbb{E}(\mathbb{P})}$ using d_{F}:

$$\begin{aligned}
d_{\mathrm{F}}^{\max}(\mathcal{U}, \mathcal{V}) &= \max_{\pi \Subset \mathcal{U}, \sigma \Subset \mathcal{V}} d_{\mathrm{F}}(\pi, \sigma)\,, \\
d_{\mathrm{F}}^{\min}(\mathcal{U}, \mathcal{V}) &= \min_{\pi \Subset \mathcal{U}, \sigma \Subset \mathcal{V}} d_{\mathrm{F}}(\pi, \sigma)\,, \\
d_{\mathrm{F}}^{\mathbb{E}(\mathbb{P})}(\mathcal{U}, \mathcal{V}) &= \mathbb{E}_{\pi \Subset_{\mathbb{P}} \mathcal{U}, \sigma \Subset_{\mathbb{P}} \mathcal{V}}[d_{\mathrm{F}}(\pi, \sigma)]\,.
\end{aligned}$$

For the weak Fréchet distance, we only use the lower bounds in this thesis, but the other definitions can be made in a similar way.

$$\begin{aligned}
d_{\mathrm{dwF}}^{\min}(\mathcal{U}, \mathcal{V}) &= \min_{\pi \Subset \mathcal{U}, \sigma \Subset \mathcal{V}} d_{\mathrm{dwF}}(\pi, \sigma)\,, \\
d_{\mathrm{wF}}^{\min}(\mathcal{U}, \mathcal{V}) &= \min_{\pi \Subset \mathcal{U}, \sigma \Subset \mathcal{V}} d_{\mathrm{wF}}(\pi, \sigma)\,.
\end{aligned}$$

If the distribution is clear from the context, we write $d_{\mathrm{F}}^{\mathbb{E}}$ and $d_{\mathrm{dF}}^{\mathbb{E}}$. The definitions above also apply if one of the curves is precise, as a precise curve is a special case of an uncertain curve.

CHAPTER 3

Similarity of Uncertain Curves in 2D

For many trajectory analysis tasks, it is imperative that we establish a trajectory similarity measure that captures the appropriate features of the trajectories. In Section 1.2.1, we have argued that the Fréchet distance is a suitable metric for this. We have also argued that it would be beneficial to consider such fundamental constructions under uncertainty. In this and the next chapter, we bridge the gap by studying the Fréchet distance under uncertainty.

As we have already defined it in Section 1.4, a *trajectory* is a sequence of measured locations with associated time stamps. The Fréchet distance does not have special provisions for time, so we may turn the time into a regular dimension or ignore it completely; either way, we obtain a sequence of locations in some space. For the purposes of this chapter, assume we have a sequence of locations in $\mathbb{R}^2$. Such a sequence without explicit time stamps is referred to as a *polygonal curve.* An *uncertain curve* is a sequence of uncertain points.

Each uncertain point is a set of potential locations. Recall the types of models for uncertain points introduced in Section 1.1.2. In this chapter, we consider the *extremal* questions in the *indecisive* and the *imprecise* models. In the indecisive model, each point is a finite set of potential locations, and in the imprecise model, each point is a compact connected region. A *realisation* of a set of uncertain points is a selection of one point from each uncertain point. When asking extremal questions, the goal is typically to compute the realisation of a set of uncertain points that minimises or maximises some quantity (e.g. area, distance, perimeter) of some underlying geometric structure (e.g. convex hull, Euclidean minimum spanning tree). See Section 1.1.2 for a detailed overview of such problems.

Here we consider both the continuous and discrete Fréchet distance for uncertain curves. Our uncertain input is given as a sequence of regions, from which a polygonal curve is *realised* by selecting one point from each region. Our goal is to find, for a given pair of uncertain curves, the upper bound, the lower bound, and the expected Fréchet distance, where the upper (resp. lower) bound Fréchet distance is the maximum (resp. minimum) distance over any realisation. For the expected Fréchet distance, we assume a probability distribution is provided that describes how each vertex on a curve is chosen from the region.

Previous work. There has been surprisingly little work using these uncertainty models in curve and trajectory analysis. Buchin and Sijben [65] have studied the discrete Fréchet distance for uncertain points modelled by a probability distribution. However, their problem is quite different from our variant: they show how to compute the distance distribution for a fixed coupling between the two curves and then solve the problem of finding the optimal coupling that achieves a given Fréchet distance. We look at the problem with the different order of quantifiers: we know how to find the optimal coupling for any realisation and want to find 'optimal' realisations yielding a certain distance.

Previously, Ahn et al. [18] considered the lower bound problem as we define it for the discrete Fréchet distance, giving a polynomial-time algorithm for uncertain points modelled by balls or hyperrectangles in constant dimension. The authors also gave efficient approximation algorithms for the discrete upper bound Fréchet distance for uncertain inputs, where the approximation factor depends on the spread of the region diameters or how well-separated they are. Subsequently, Fan and Zhu showed that the discrete upper bound Fréchet distance is NP-hard for uncertain inputs modelled as thin rectangles [109]. To our

Table 3.1. Hardness results for the decision problems in this chapter. Ahn et al. [18] solve the lower bound problem for disks, but their algorithm extends to the indecisive curves as well as to line-segment imprecision. DFD and FD stand for the (discrete) Fréchet distance; LB, UB, and Exp refer to the lower bound, the upper bound, and the expected value, respectively.

		indecisive	imprecise	
			disks	line segments
DFD	LB	Polynomial [18]	Polynomial [18]	Polynomial [18]
	UB	NP-complete	NP-hard	NP-hard
	Exp	#P-hard	—	#P-hard
FD	LB	Polynomial	—	NP-hard
	UB	NP-complete	NP-hard	NP-hard
	Exp	#P-hard	—	—

knowledge, we are the first to consider either variant for the continuous Fréchet case, and the first to consider the expected Fréchet distance.

Contributions. In this chapter, we present an extensive study of the Fréchet distance for uncertain curves in two (and higher) dimensions. We provide a wide range of hardness results and present several approximations and polynomial-time solutions to restricted versions. We are the first to consider the continuous Fréchet distance in the uncertain setting, as well as the first to consider the expected Fréchet distance.

On the negative side, we present a plethora of hardness results (see Table 3.1; details follow in Section 3.1). The hardness of the lower bound case is curious: while the discrete Fréchet distance on imprecise inputs [18] and, as we prove, continuous Fréchet distance on indecisive inputs both permit a simple dynamic programming solution, the continuous Fréchet distance problem on imprecise input has just enough (literal) wiggle room to show NP-hardness by reduction from SubsetSum. In Chapter 4, we explore this in 1D and find a similar dichotomy for the weak Fréchet distance.

We complement the lower bound hardness result by two approximation algorithms (Section 3.2). The first is a FPTAS for general uncertain curves in constant dimension when the ratio between the diameter of the uncertain points and the lower bound Fréchet distance is polynomially bounded. The second is a 3-approximation for separated imprecise curves, but uses a simpler greedy approach that runs in near-linear time.

The NP-hardness of the upper bound by a reduction from CNF-SAT is less surprising, but requires a careful set-up and analysis of the geometry to then extend it to a reduction from #CNF-SAT to the expected (discrete or continuous) Fréchet distance under the uniform distribution. However, by adding the common constraint that the alignment between the curves needs to stay within a Sakoe–Chiba [194] band of constant width (see Section 3.3 for definition and results), we can solve these problems in polynomial time for indecisive curves. Sakoe–Chiba bands are frequently used for time-series data [32, 151, 194] and trajectories [41, 93], when the alignment should (or is expected to) not vary too much from a certain 'natural' alignment.

3.1 Hardness Results

In this section, we first present the hardness results for the upper bound and the expected value of the continuous and the discrete Fréchet distance for indecisive and imprecise curves. We then prove hardness of finding the lower bound continuous Fréchet distance on imprecise curves. Refer to Chapter 2 for the relevant definitions and notation.

3.1.1 Upper Bound and Expected Fréchet Distance

We present proofs of NP-hardness and #P-hardness for the upper bound and the expected Fréchet distance for both indecisive and imprecise curves by showing polynomial-time reductions from CNF-SAT (satisfiability of a boolean formula) and #CNF-SAT (its counting version). We consider the upper bound problem for indecisive curves and then illustrate how the construction can be used to show #P-hardness for the expected Fréchet distance (both discrete and continuous). We then demonstrate how the construction can be adapted to show hardness for imprecise curves. All our constructions are in two dimensions.

Upper Bound Fréchet Distance: Basic Construction

Define the following problem.

Problem 3.1. Upper Bound Discrete Fréchet: Given two uncertain curves $\mathcal{U}$ and $\mathcal{V}$ and a threshold $\delta \in \mathbb{R}^+$, decide if $d_{\mathrm{dF}}^{\max}(\mathcal{U}, \mathcal{V}) > \delta$.

We can similarly define its continuous counterpart, using $d_{\mathrm{F}}^{\max}$ instead.

Problem 3.2. Upper Bound Continuous Fréchet: Given two uncertain curves $\mathcal{U}$ and $\mathcal{V}$ and a threshold $\delta \in \mathbb{R}^+$, decide if $d_{\mathrm{F}}^{\max}(\mathcal{U}, \mathcal{V}) > \delta$.

We first give some extra definitions to make the proofs clearer. Suppose we are given a CNF-SAT *formula* C with

$$C = \bigwedge_{i \in [n]} C_i\,, \qquad C_i = \bigvee_{j \in J_i} x_j \vee \bigvee_{k \in K_i} \neg x_k \quad \text{for all } i \in [n].$$

Here n and m are the numbers of clauses and variables, respectively, x_j for any $j \in [m]$ is a boolean variable, and $J_i \subseteq [m]$, $K_i \subseteq [m] \setminus J_i$ for all $i \in [n]$ are sets of relevant variable indices. Such a variable may be assigned 'true' or 'false'; an *assignment* is a function $a : \{x_1, \dots, x_m\} \to \{\mathsf{True}, \mathsf{False}\}$ that assigns a value to each variable, $a(x_j) = \mathsf{True}$ or $a(x_j) = \mathsf{False}$ for any $j \in [m]$. We denote by $C[a]$ the result of substituting $x_j \mapsto a(x_j)$ in C for all $j \in [m]$ As an aid to the reader, the problem we reduce from is as follows.

Problem 3.3. CNF-SAT: Given a CNF-SAT formula C, decide if there is an assignment a such that $C[a] = \mathsf{True}$.

We pick some value $0 < \varepsilon < 0.25$.[1] Construct a *variable curve,* where each variable corresponds to an indecisive point with locations $(0, 0.5 + \varepsilon)$ and $(0, -0.5 - \varepsilon)$; the locations are interpreted as assigning the variable True and False. Any realisation of the curve corresponds to a variable assignment.

Intuitively, one curve encodes the variables, and the other encodes the structure of the formula. We define a variable gadget on a variable curve to encode the value of a boolean variable; and we define assignment gadgets on the other curve to encode the literals x and $\neg x$ occurring in the formula. The gadgets interact with each other, so if a literal is true, the distance is large. The assignment gadgets have positions for 'true', 'false', and 'do not care' values, the latter being used to skip a variable

---

[1] This range is determined by the relative distances in the construction.

unused in a clause. We repeat the construction for each variable on both curves with some synchronisation enforcement, constructing a variable clause gadget and an assignment clause gadget, so the distance is large if the clause is satisfied by setting the variables in a specific way. Finally, we construct the full variable curve and the clause curve. Here the goal is that we have a single copy of variables that can be assigned `True` or `False`; and we can choose which clause we want to align with them. The other clauses are caught by extra points on the variable curve so as to not affect the distance. Some clauses are not satisfied, and they will yield a small distance; others are satisfied and yield a large distance; so since we can choose the clause freely, we only get a large distance between complete curves if *all* the clauses give a large distance, so all are satisfied, and so is the formula. Finding the upper bound Fréchet distance now corresponds to finding the realisation of the points that achieves the large distance, or finding the truth assignment of the variables that satisfies the formula. We show the locations used by the gadgets and their nesting in Figure 3.1. We present an example construction for a specific formula and a realisation in Figure 3.2, highlighting also the possible alignment options between the clause curve and the variable curve and the resulting distances. Next, we define the gadgets formally level by level and prove that the distances are correct.

Literal level. Define a *variable gadget*, where an indecisive point corresponds to a variable and is followed by a precise point far away, to force synchronisation with the other curve:

$$\mathrm{VG}_j = \{(0, 0.5 + \varepsilon), (0, -0.5 - \varepsilon)\} \sqcup (2, 0)\,.$$

Consider a specific clause C_i of the formula. We define an *assignment gadget* $\mathrm{AG}_{i,j}$ for each variable x_j and clause C_i depending on how the variable occurs in the clause.

$$\mathrm{AG}_{i,j} = \begin{cases} (0, -0.5) \sqcup (1, 0) & \text{if } x_j \text{ is a literal of } C_i, \\ (0, 0.5) \sqcup (1, 0) & \text{if } \neg x_j \text{ is a literal of } C_i, \\ (0, 0) \sqcup (1, 0) & \text{otherwise.} \end{cases}$$

Note that if the assignment $x_j =$ `True` makes the clause C_i true, then the first precise point of the corresponding assignment gadget appears at

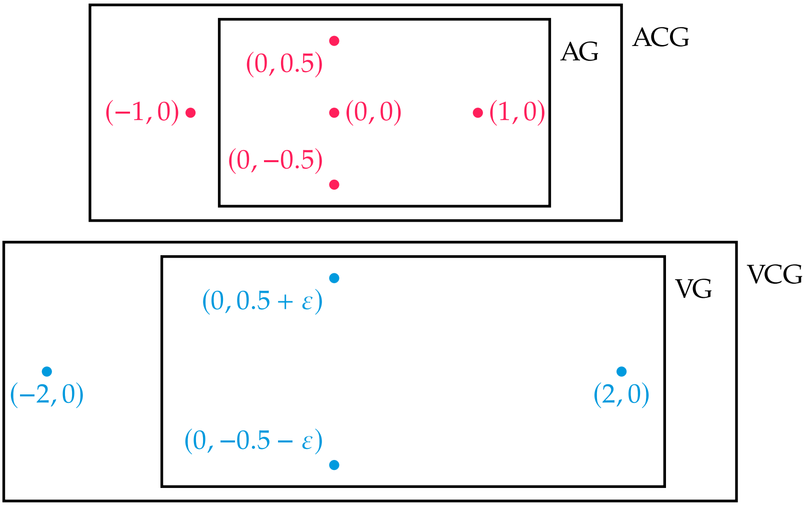


Figure 3.1. Illustration of the gadgets used in the basic construction. Assignment gadgets are repeated to make up the assignment clause gadgets; they are repeated to make up the clause curve. Variable gadgets are repeated to make up the variable clause gadget; it is prepended and appended by $(0, 0)$ to make up the variable curve.

distance $1 + \varepsilon$ from the realisation corresponding to setting $x_j = \mathsf{True}$ of the indecisive point in VG_j.

We now show the relation between the gadgets. To do so, we introduce the *one-to-one coupling* as a valid coupling $c = \langle (p_1, q_1), \ldots, (p_r, q_r) \rangle$, where the coupling is restricted to $(p_{s+1}, q_{s+1}) = (p_s + 1, q_s + 1)$ for all $s \in [r - 1]$. Necessarily, such a coupling only exists for curves of equal length.

Lemma 3.4. *Suppose we are given a clause C_i and a variable x_j that both occur in a CNF-SAT formula C, and we restrict the set of valid couplings $\mathcal{C}$ to only contain one-to-one couplings. We only get the discrete Fréchet distance equal to $1 + \varepsilon$ if the realisation of VG_j we pick corresponds to the assignment of x_j that ensures the clause C_i is satisfied; otherwise, the discrete Fréchet distance is 1. In other words, if we consider $\pi \Subset \mathrm{VG}_j$ that corresponds to setting $a(x_j)$, then*

$$d_{\mathrm{dF}}(\pi, \mathrm{AG}_{i,j}) = \begin{cases} 1 + \varepsilon & \text{if assigning } x_j \text{ satisfies } C_i, \\ 1 & \text{otherwise.} \end{cases}$$

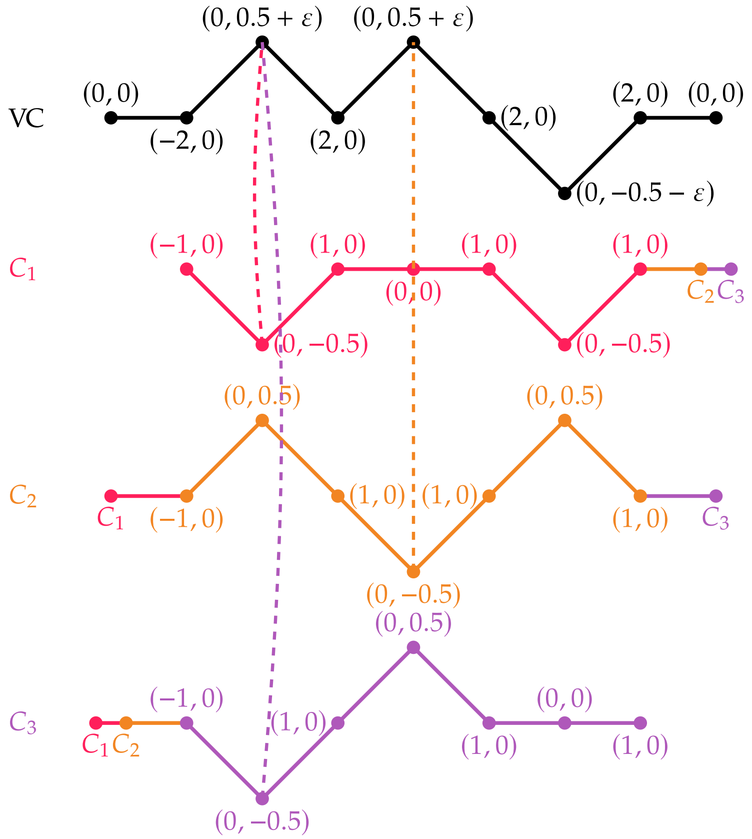


Figure 3.2. Realisation of VC for the assignment x_1 = True, x_2 = True, x_3 = False and CC for the formula $C = (x_1 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2)$. We show the variable curve; and three times the clause curve, since we have three feasible options for matching the curves, corresponding to the three clauses. The other clauses are matched to $(0, 0)$ and are collapsed to a point in the figure. Note that C = True with the given variable assignment. Also note that we can choose any of C_1, C_2, C_3 to couple to VC; we always get the bottleneck distance of $1 + \varepsilon$, as all three are satisfied, so here $d_{\mathrm{dF}}(\mathrm{VC}, \mathrm{CC}) = 1 + \varepsilon$.

Proof. First of all, observe that as we only consider one-to-one couplings, the second points of both gadgets must be coupled; the distance between them is $\|(2, 0) - (1, 0)\| = 1$; thus, the discrete Fréchet distance between the curves must be at least 1.

Now consider the possible realisations of VG_j. Say, we pick the realisation $(0, 0.5 + \varepsilon) \sqcup (2, 0)$, which corresponds to assigning $a(x_j) = \mathtt{True}$. If x_j is a literal of C_i, so $C_i[a] = \mathtt{True}$, then by construction we know that $\mathrm{AG}_{i,j}$ is $(0, -0.5) \sqcup (1, 0)$. Since we consider only the one-to-one couplings, we must couple the first points together, yielding the distance $\|(0, 0.5 + \varepsilon) - (0, -0.5)\| = 1 + \varepsilon > 1$, so the discrete Fréchet distance in this case is $1 + \varepsilon$, and indeed we picked the assignment that ensures that C_i is satisfied. If instead $\neg x_j$ is a literal of C_i, so $C_i[a] = \mathtt{False}$, then we know that $\mathrm{AG}_{i,j}$ is $(0, 0.5) \sqcup (1, 0)$, and it is easy to see that, as $\|(0, 0.5 + \varepsilon) - (0, 0.5)\| = \varepsilon < 1$, we get the discrete Fréchet distance of 1, and that we picked an assignment that does not ensure that C_i is satisfied.

A symmetric argument can be applied when we consider the realisation $(0, -0.5 - \varepsilon) \sqcup (2, 0)$ for VG_j: if $\neg x_j$ is a literal of C_i, then we get the discrete Fréchet distance of $1 + \varepsilon$ and we picked an assignment that surely satisfies C_i.

Finally, consider the case when $\mathrm{AG}_{i,j} = (0, 0) \sqcup (1, 0)$. This implies that assigning a value to x_j has no effect on C_i, i.e. a literal involving x_j does not occur in C_i, so neither assignment (and neither realisation of VG_j) would ensure that C_i is satisfied. Also observe that $\|(0, 0.5 + \varepsilon) - (0, 0)\| = \|(0, -0.5 - \varepsilon) - (0, 0)\| = 0.5 + \varepsilon < 1$, so both realisations yield the discrete Fréchet distance of 1.

So, we can conclude that we get the distance $1 + \varepsilon$ if and only if the partial assignment of a value to x_j ensures that C_i is satisfied; otherwise, we get the distance 1. □

Clause level. We can repeat the construction, yielding a *variable clause gadget* and an *assignment clause gadget:*

$$\mathrm{VCG} = (-2, 0) \sqcup \bigsqcup_{j \in [m]} \mathrm{VG}_j\,, \qquad \mathrm{ACG}_i = (-1, 0) \sqcup \bigsqcup_{j \in [m]} \mathrm{AG}_{i,j}\,.$$

Consider the Fréchet distance between the two gadgets. Observe that coupling a synchronisation point from one gadget with a non-synchronisation point in the other yields a distance larger than $1 + \varepsilon$, whereas coupling synchronisation points pairwise and non-synchronisation points pairwise will yield the distance at most $1 + \varepsilon$. So we only consider one-to-one couplings, i.e. we couple point i on one curve to point i on the other curve, for all i.

Now if a realisation corresponds to a satisfying assignment, then for some x_j, we have picked the realisation that is opposite from the coupled point on the clause curve, yielding the bottleneck distance of $1 + \varepsilon$. If the realisation corresponds to a non-satisfying assignment, then the synchronisation points establish the bottleneck, yielding the distance 1. So we can clearly distinguish between a satisfying and a non-satisfying assignment for a clause. It is crucial now that we show the following.

Lemma 3.5. *Given a CNF-SAT formula C containing some clause C_i and m variables $x_1, \dots, x_m$, consider curves $\alpha_1 \sqcup \mathrm{VCG} \sqcup \alpha'_1$ and $\alpha_2 \sqcup \mathrm{ACG}_i \sqcup \alpha'_2$ for arbitrary precise curves α_1, α'_1, α_2, α'_2 with $|\alpha_1| = k$ and $|\alpha_2| = l$. If an optimal coupling between $\alpha_1 \sqcup \mathrm{VCG} \sqcup \alpha'_1$ and $\alpha_2 \sqcup \mathrm{ACG}_i \sqcup \alpha'_2$ for any realisation of* VCG *has a pair $(k + 1, l + 1)$, then there is an optimal coupling that has pairs $(k + s, l + s)$ for all $s \in [2m + 1]$, i.e. there is an optimal coupling that is one-to-one for any realisation of* VCG.

Proof. Observe that both gadgets have exactly $2m + 1$ points. Suppose the optimal coupling Opt has a pair $(k + 1, l + 1)$, so it couples the first points of VCG and ACG_i. If Opt is already one-to-one for all $s \in [2m+1]$, there is nothing to be done. Suppose now that it is one-to-one until some $1 \leq r < 2m + 1$, so it has pairs $(k + s, l + s)$ for all $s \in [r]$, but it does not have a pair $(k + (r + 1), l + (r + 1))$. Then one of the following cases occurs.

- $r = 2q + 2$ is even; then we know that the point $(2, 0)$ in VG_{q+1} is not coupled to the point $(1, 0)$ in $\mathrm{AG}_{i,q+1}$, but the preceding indecisive point is coupled to the assignment point. Then either $(2, 0)$ is coupled to an assignment point, with the distance at least 2, or $(1, 0)$ is coupled to an indecisive point, yielding the distance of $\sqrt{1 + (0.5 + \varepsilon)^2} > 1$. If we eliminate that pair and instead couple $(2, 0)$ to $(1, 0)$, we will still have a valid coupling and obtain the distance of 1 on this pair; thus, the new coupling is not worse that the original one, and so it is also an optimal coupling that is one-to-one for all $s \in [r + 1]$.

- $r = 2q + 1$ is odd; then we know that the indecisive point in VG_{q+1} is not coupled to the assignment point in $\mathrm{AG}_{i,q+1}$, but the preceding $(2, 0)$ and $(1, 0)$ (or $(-2, 0)$ and $(-1, 0)$) are coupled. Then either Opt has a pair of the indecisive point and $(1, 0)$, or it has a pair of the assignment point and $(2, 0)$. (The cases for $(-1, 0)$ and $(-2, 0)$ are symmetrical.) In either case, we want to eliminate that pair from the coupling and instead add the pair of the indecisive point and the assignment point, yielding a valid coupling that is one-to-one for all $s \in [r + 1]$. To complete the proof for this case, we need to show that such a coupling is optimal.

  Consider the first possible coupling. The distance between the indecisive point and $(1, 0)$ is $\sqrt{1 + (0.5 + \varepsilon)^2}$, whereas the distance between the indecisive and the assignment point is ε, $0.5 + \varepsilon$, or $1 + \varepsilon$. As $\varepsilon < 0.25$, note that

$$\begin{aligned} 0.25 + \varepsilon &> 2\varepsilon \\ 1 + 0.25 + \varepsilon + \varepsilon^2 &> 1 + 2\varepsilon + \varepsilon^2 \\ 1 + (0.5 + \varepsilon)^2 &> (1 + \varepsilon)^2 \\ \sqrt{1 + (0.5 + \varepsilon)^2} &> 1 + \varepsilon \,, \end{aligned}$$

  so our change to the optimal coupling will replace a pair with a pair of lower distance, so the new coupling is at least as good as the original one, and thus optimal.

  Now consider the second coupling. The distance between the assignment point and $(2, 0)$ is at least 2, and $2 > 1 + \varepsilon > 0.5 + \varepsilon > \varepsilon$, so again our change yields an optimal coupling.

By induction on r, we conclude that the statement of the lemma holds. □

We can now use the two previous results to show the following.

Lemma 3.6. *Given a CNF-SAT formula C containing some clause C_i and m variables $x_1, \dots, x_m$, construct curves $\alpha_1 \sqcup \mathrm{VCG} \sqcup \alpha'_1$ and $\alpha_2 \sqcup \mathrm{ACG}_i \sqcup \alpha'_2$ for arbitrary precise curves α_1, α'_1, α_2, α'_2 with $|\alpha_1| = k$ and $|\alpha_2| = l$. If some optimal coupling between $\alpha_1 \sqcup \mathrm{VCG} \sqcup \alpha'_1$ and $\alpha_2 \sqcup \mathrm{ACG}_i \sqcup \alpha'_2$ for any realisation of* VCG *has a pair $(k + 1, l + 1)$ and $d_{\mathrm{dF}}(\alpha_1, \alpha_2) \leq 1$ and $d_{\mathrm{dF}}(\alpha'_1, \alpha'_2) \leq 1$, then the discrete Fréchet distance between the curves is $1 + \varepsilon$ for realisations of* VCG *that correspond to satisfying assignments for C_i, and* 1 *for realisations that do not. In other words, if* $\pi \Subset \text{VCG}$ *corresponds to an assignment* a *and we only consider the restricted couplings, then*

$$d_{\text{dF}}(\alpha_1 \sqcup \pi \sqcup \alpha_1', \alpha_2 \sqcup \text{ACG}_i \sqcup \alpha_2') = \begin{cases} 1 + \varepsilon & \textit{if } C_i[a] = \texttt{True}, \\ 1 & \textit{otherwise.} \end{cases}$$

Proof. First of all, since some optimal coupling between $\alpha_1 \sqcup \text{VCG} \sqcup \alpha_1'$ and $\alpha_2 \sqcup \text{ACG}_i \sqcup \alpha_2'$ for any realisation of VCG has a pair $(k+1, l+1)$, we can use Lemma 3.5 to find an optimal coupling Opt that is one to one on the subcurves corresponding to the gadgets. That means that we can, essentially, split the curves, if we consider only such restricted couplings:

$$\begin{aligned} &d_{\text{dF}}(\alpha_1 \sqcup \pi \sqcup \alpha_1', \alpha_2 \sqcup \text{ACG}_i \sqcup \alpha_2') \\ &\qquad = \max(d_{\text{dF}}(\alpha_1, \alpha_2), d_{\text{dF}}(\pi, \text{ACG}_i), d_{\text{dF}}(\alpha_1', \alpha_2')) \\ &\qquad = \max(1, d_{\text{dF}}(\pi, \text{ACG}_i))\,, \end{aligned}$$

where the last equality follows from the fact that $d_{\text{dF}}(\pi, \text{ACG}_i) \geq 1$, since the first points are in a coupling and have the distance 1, and from the assumption that $d_{\text{dF}}(\alpha_1, \alpha_2) \leq 1$ and $d_{\text{dF}}(\alpha_1', \alpha_2') \leq 1$. Note that here we do not restrict the coupling on α_1, α_2 and α_1', α_2'.

To obtain the end result, we need to consider the distance between π and ACG_i under a one-to-one coupling. Using Lemma 3.4, it is easy to see that if we have $a(x_j) = \texttt{True}$ for some variable x_j and x_j is a literal in C_i, then $C_i[a] = \texttt{True}$, and $d_{\text{dF}}(\pi, \text{ACG}_i) = 1 + \varepsilon$; similarly, if $a(x_j) = \texttt{False}$ for some variable x_j and $\neg x_j$ is a literal in C_i, then $C_i[a] = \texttt{True}$, and $d_{\text{dF}}(\pi, \text{ACG}_i) = 1 + \varepsilon$. If there is no such x_j, then $C_i[a] = \texttt{False}$ and $d_{\text{dF}}(\pi, \text{ACG}_i) = 1$. We conclude that the lemma holds. □

Formula level. Next, we define the *variable curve* and the *clause curve* as follows:

$$\text{VC} = (0, 0) \sqcup \text{VCG} \sqcup (0, 0)\,, \qquad \text{CC} = \bigsqcup_{i \in [n]} \text{ACG}_i\,.$$

Observe that the synchronisation points at $(-2, 0)$ and $(-1, 0)$ ensure that for any optimal coupling, we match up VCG with some ACG_i as described before. Also note that all the points on CC are within distance 1 from $(0, 0)$. Therefore, we can always pick any one of n

clauses to couple to VCG, and couple the remaining points to $(0,0)$; the bottleneck distance will then be determined by the distance between VCG and the chosen ACG_i.

Now consider a realisation of VCG. If the corresponding assignment does not satisfy C, then we can synchronise VCG with a clause that is false to obtain the distance of 1. If the assignment corresponding to the realisation satisfies all the clauses, we must synchronise VCG with a satisfied clause, which yields a distance of $1 + \varepsilon$. We show the following important property of our construction.

Lemma 3.7. *Given a CNF-SAT formula C with n clauses and m variables, construct the curves* VC *and* CC *as defined above and consider a realisation* $(0,0) \sqcup \pi \sqcup (0,0)$ *of curve* VC*, corresponding to some assignment a. Then, under no restrictions on the couplings except those imposed by the definition,*

$$d_{\mathrm{dF}}((0,0) \sqcup \pi \sqcup (0,0), \mathrm{CC}) = \begin{cases} 1 + \varepsilon & \text{if } C[a] = \textit{True}, \\ 1 & \text{if } C[a] = \textit{False}. \end{cases}$$

In other words, the discrete Fréchet distance is $1 + \varepsilon$ if the realisation corresponds to a satisfying assignment, and is 1 otherwise.

Proof. We can show this by proving that the premises of Lemma 3.6 are satisfied.

First of all, note that all the points of CC are within distance 1 from $(0,0)$. Furthermore, note that we can always give a coupling with the distance at most $1 + \varepsilon$: couple $(0,0)$ to $(-1,0)$ from ACG_1, then walk along the realisation of VCG and ACG_1 in a one-to-one coupling, and then couple the remaining points in CC to $(0,0)$. As all the points of CC are within distance 1 from $(0,0)$ and as this is otherwise the construction of Lemma 3.6, this coupling yields the discrete Fréchet distance of at most $1 + \varepsilon$ for any realisation of VC. Therefore, any coupling that has pairs further away than $1 + \varepsilon$ cannot be optimal. Observe that the only point within that distance from $(-2,0)$ is $(-1,0)$. Therefore, we only need to consider couplings that couple the first point of the realisation of VCG to the first point of some ACG_i as possibly optimal. Thus, for each of the n couplings we get, we can apply Lemma 3.6. There are two cases to consider.

- There is some gadget ACG_i with the distance 1 to π under the one-to-one coupling. Then we can choose that gadget to couple to

π and couple all the other points in CC to $(0, 0)$ at the beginning or at the end of VC as suitable. As all the points of CC are within distance 1 from $(0, 0)$, this coupling will yield distance 1; as a lower distance is impossible, this coupling is optimal, so then $d_{\mathrm{dF}}((0, 0) \sqcup \pi \sqcup (0, 0), \mathrm{CC}) = 1$. Observe that by our construction this situation corresponds to the case when $C_i[a] = \mathtt{False}$, by Lemma 3.6, and so indeed $C[a] = \mathtt{False}$.

- The distance between any gadget ACG$_i$ and π under the one-to-one coupling is $1 + \varepsilon$. Then, no matter which gadget we choose to couple to π, we will get the distance of $1 + \varepsilon$, so in this case $d_{\mathrm{dF}}((0, 0) \sqcup \pi \sqcup (0, 0), \mathrm{CC}) = 1 + \varepsilon$. Note that, by our construction, this means that $C_i[a] = \mathtt{True}$ for all $i \in [n]$; therefore, indeed $C[a] = \mathtt{True}$.

As we have covered all the possible cases, we conclude that the lemma holds. ☐

We illustrate the gadgets of the construction in Figure 3.1. We also show an example of the correspondence between a boolean formula and our construction in Figure 3.2.

Upper Bound Discrete Fréchet Distance on Indecisive Points

Theorem 3.8. *The problem* Upper Bound Discrete Fréchet *for indecisive curves is NP-complete.*

Proof. First of all, observe that if two realisations of lengths n and m are given as a certificate for a 'Yes'-instance of the problem, then one can verify the solution by computing the discrete Fréchet distance between the realisations and checking that it is indeed larger than the threshold δ. The computation can be done in time $\Theta(mn)$, using the algorithm proposed by Eiter and Mannila [100]. Therefore, the problem is in NP.

Now suppose we are given an instance of CNF-SAT, i.e. a CNF-SAT formula C with n clauses and m variables. We construct the curves VC and CC, as described previously, and get an instance of Upper Bound Discrete Fréchet on curves VC and CC with the threshold $\delta = 1$. If the answer is 'Yes', then we also output 'Yes' as an answer to CNF-SAT; otherwise, we output 'No'.

Using Lemma 3.7, we see that if there is some assignment a such that $C[a] = \texttt{True}$, then for the corresponding realisation, the discrete Fréchet distance is $1 + \varepsilon$; the other way around, if for some realisation we get the distance $1 + \varepsilon$, then by our construction, all the clauses are satisfied and $C[a] = \texttt{True}$; and so $d_{\mathrm{dF}}^{\max}(\mathrm{VC}, \mathrm{CC}) = 1 + \varepsilon$. On the other hand, if there is no such assignment a, then for any assignment a, there is some C_i with $C_i[a] = \texttt{False}$, yielding $C[a] = \texttt{False}$, and also for any realisation of VC, there is some gadget ACG_i that yields the discrete Fréchet distance of 1; and so $d_{\mathrm{dF}}^{\max}(\mathrm{VC}, \mathrm{CC}) = 1$. Therefore, the formula C is satisfiable if and only if $d_{\mathrm{dF}}^{\max}(\mathrm{VC}, \mathrm{CC}) > 1$, and so our answer is correct.

Furthermore, observe that the curves have $2m + 2$ and $2mn + n$ points, respectively, and so the instance of Upper Bound Discrete Fréchet that gives the answer to CNF-SAT can be constructed in polynomial time. Thus, we conclude that Upper Bound Discrete Fréchet for indecisive curves is NP-hard; combining it with the first part of the proof shows that it is NP-complete. □

Upper Bound Fréchet Distance on Indecisive Points

We use the same construction as for the discrete Fréchet distance. To follow the same proof structure, we need to present arguments for the continuous case that lead up to an alternative to Lemma 3.7. For the arguments to work, we need to further restrict the range of ε to be $[0.12, 0.25)$.

Consider the construction drawn in Figure 3.3. The key points here are that $(0, 0.5 + \varepsilon)$ is far from any point on the clause curve, and that $(2, 0)$ is only close enough to $(1, 0)$. We can obtain a lemma similar to Lemma 3.4.

Lemma 3.9. *Given a clause C_i and a variable x_j that both occur in the CNF-SAT formula C, we only get the Fréchet distance equal to $(1 + \varepsilon) \cdot 2/\sqrt{5}$ if the realisation of VG_j we pick corresponds to the assignment of x_j that ensures the clause C_i is satisfied; otherwise, the Fréchet distance is 1. In other words, if we consider $\pi \Subset \mathrm{VG}_j$ that corresponds to setting $a(x_j)$, then*

$$d_{\mathrm{F}}(\pi, \mathrm{AG}_{i,j}) = \begin{cases} (1 + \varepsilon) \cdot \frac{2}{\sqrt{5}} & \text{if assigning } x_j \text{ satisfies } C_i, \\ 1 & \text{otherwise.} \end{cases}$$

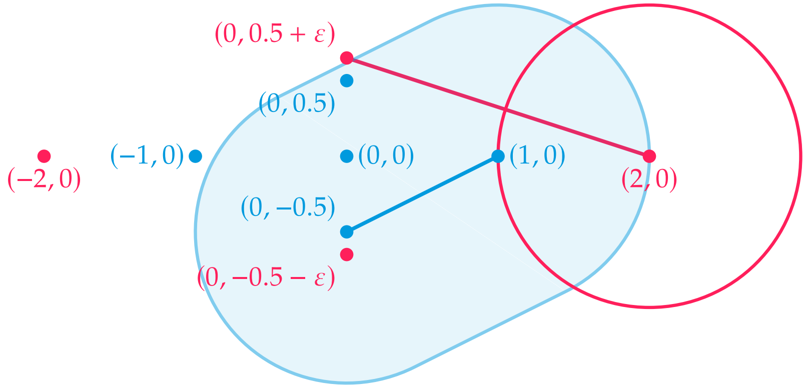


Figure 3.3. Construction for $\varepsilon = 0.15$. Shaded blue area shows the points within distance 1 from the segment $(0, -0.5) \sqcup (1, 0)$. Observe that $(0, 0.5 + \varepsilon)$ is outside that region, and that $(1, 0)$ is the only blue point within distance 1 from $(2, 0)$.

Proof. Consider the possible realisations of VG_j. Suppose we pick the realisation $(0, 0.5 + \varepsilon) \sqcup (2, 0)$, which corresponds to assigning $a(x_j) = \mathsf{True}$. If x_j is a literal in C_i, so $C_i = \mathsf{True}$, then by construction we know that $\mathrm{AG}_{i,j}$ is $(0, -0.5) \sqcup (1, 0)$. As noted in Figure 3.3, the distance between $(0, 0.5 + \varepsilon)$ and any point on $(0, -0.5) \sqcup (1, 0)$ is larger than 1. To be more specific, the distance between the point (x, y) and the line defined by $(x_1, y_1) \sqcup (x_2, y_2)$ can be determined using a standard formula as

$$d = \frac{|x(y_2 - y_1) - y(x_2 - x_1) + x_2 y_1 - x_1 y_2|}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}} .$$

In our case, we get

$$d = \frac{|0 - (0.5 + \varepsilon) \cdot (1 - 0) - 1 \cdot 0.5 - 0|}{\sqrt{(1 - 0)^2 + (0 + 0.5)^2}} = \frac{2 \cdot (1 + \varepsilon)}{\sqrt{5}} .$$

As the point $(0, 0.5 + \varepsilon)$ must be aligned with some point on $\mathrm{AG}_{i,j}$, the Fréchet distance we get in this case cannot be smaller than d. Furthermore, it is easy to see that the point $(0, 0.5 + \varepsilon)$ is the furthest point from $\mathrm{AG}_{i,j}$; thus, we get that the Fréchet distance is exactly d.

On the other hand, if $\neg x_j$ is a literal in C_i, then by construction we know that $\mathrm{AG}_{i,j}$ is $(0, 0.5) \sqcup (1, 0)$. As noted in Figure 3.3, the distance between $(2, 0)$ and any point on $(0, 0.5) \sqcup (1, 0)$ is at least 1, with the smallest distance achieved at $(1, 0)$. It is clear that this is the furthest pair of points on the two gadgets in this case; thus, we get the Fréchet distance of 1.

A symmetric argument can be applied when we consider the realisation $(0, -0.5 - \varepsilon) \sqcup (2, 0)$ for VG_j: if $\neg x_j$ is a literal in C_i, then we get the Fréchet distance of d and we picked an assignment that satisfies C_i; and in the other case, we get that C_i is not necessarily satisfied and the Fréchet distance is 1.

Finally, consider the case when $\mathrm{AG}_{i,j} = (0, 0) \sqcup (1, 0)$. Again, this implies that assigning a value to x_j has no effect on C_i, so neither assignment (and neither realisation of VG_j) would ensure that C_i is satisfied. Also observe that both realisations give rise to curves that are entirely within distance 1 of $(0, 0) \sqcup (1, 0)$, yielding the Fréchet distance of 1. $\square$

We can now naturally get a lemma similar to Lemma 3.6.

Lemma 3.10. *Given a CNF-SAT formula C containing some clause C_i and m variables $x_1, \dots, x_m$, construct curves $\alpha_1 \sqcup \mathrm{VCG} \sqcup \alpha'_1$ and $\alpha_2 \sqcup \mathrm{ACG}_i \sqcup \alpha'_2$ for arbitrary precise curves α_1, α'_1, α_2, α'_2 with $|\alpha_1| = k$ and $|\alpha_2| = l$. If some optimal alignment ϕ_1, ϕ_2 between $\alpha_1 \sqcup \mathrm{VCG} \sqcup \alpha'_1$ and $\alpha_2 \sqcup \mathrm{ACG}_i \sqcup \alpha'_2$ for any realisation of VCG has some value t such that $\phi_1(t) = k + 1$ and $\phi_2(t) = l + 1$ and $d_{\mathrm{F}}(\alpha_1, \alpha_2) \le 1$ and $d_{\mathrm{F}}(\alpha'_1, \alpha'_2) \le 1$, then the Fréchet distance between the curves is $(1 + \varepsilon) \cdot 2/\sqrt{5}$ for the realisations of VCG that correspond to satisfying assignments for C_i, and 1 for other realisations. In other words, if $\pi \Subset \mathrm{VCG}$ corresponds to assignment a and we only consider the restricted alignments, then*

$$d_{\mathrm{F}}(\alpha_1 \sqcup \pi \sqcup \alpha'_1, \alpha_2 \sqcup \mathrm{ACG}_i \sqcup \alpha'_2) = \begin{cases} (1 + \varepsilon) \cdot \frac{2}{\sqrt{5}} & \text{if } C_i[a] = \mathit{True}, \\ 1 & \text{otherwise.} \end{cases}$$

Proof. First of all, observe that as we traverse VCG, we need to align $(2, 0)$ with $(1, 0)$ to obtain an optimal alignment. Therefore, essentially, the traversal can be split into m parts, each of which corresponds to traversing VG_j and $\mathrm{AG}_{i,j}$ at the same time for all $j \in [m]$. We can use Lemma 3.9 to note that if some variable x_j is assigned a value that makes the clause C_i satisfied, then the Fréchet distance becomes $(1 + \varepsilon) \cdot 2/\sqrt{5}$; if that is not the case for any variables, then we can traverse the entire curve, as well as α_1 and α'_1 by linearly interpolating our position between the vertices of the curves and otherwise using the alignment derived from the coupling of the discrete case, while staying within distance 1 of the other curve, yielding the Fréchet distance of 1. The distance also cannot be smaller than 1 due to aligning $(2, 0)$ and $(1, 0)$. □

Now we can provide a lemma that mirrors Lemma 3.7.

Lemma 3.11. *Given a CNF-SAT formula C with n clauses and m variables, construct the curves* VC *and* CC *as defined above and consider a realisation* $(0, 0) \sqcup \pi \sqcup (0, 0)$ *of curve* VC*, corresponding to some assignment a. Then*

$$d_{\mathrm{F}}((0,0) \sqcup \pi \sqcup (0,0), \mathrm{CC}) = \begin{cases} (1+\varepsilon) \cdot \frac{2}{\sqrt{5}} & \text{if } C[a] = \textit{True}, \\ 1 & \text{if } C[a] = \textit{False}. \end{cases}$$

In other words, the Fréchet distance is $(1+\varepsilon) \cdot 2/\sqrt{5}$ *if the realisation π corresponds to a satisfying assignment, and is* 1 *otherwise.*

Proof. First of all, observe that any point of CC is within distance 1 of $(0, 0)$; furthermore, when starting to traverse π, we must match $(-2, 0)$ with $(-1, 0)$ in an optimal alignment. Thus, the premise of Lemma 3.10 is satisfied, and, using reasoning similar to that of Lemma 3.7, we observe that an optimal alignment chooses one of the clauses to traverse in parallel with the variable curve, and so if there is a clause that is not satisfied, then we get the Fréchet distance of 1, and if all of them are satisfied, then all of them yield the Fréchet distance of $(1 + \varepsilon) \cdot 2/\sqrt{5}$. □

Finally, we can show the main result.

Theorem 3.12. *The problem* Upper Bound Continuous Fréchet *for indecisive curves is NP-complete.*

Proof. First of all, observe that if two realisations of lengths n and m are given as a certificate for a 'Yes'-instance of the problem, then one can verify the solution by checking that the Fréchet distance between the realisations is larger than the threshold δ. The computation can be done in time $\Theta(mn)$, using the algorithm by Alt and Godau [23, 117]; so the problem is in NP.

Now suppose we are given an instance of CNF-SAT, i.e. a CNF-SAT formula C with n clauses and m variables. We construct the curves VC and CC, as described previously, and get an instance of Upper Bound

CONTINUOUS FRÉCHET on curves VC and CC with the threshold $\delta = 1$. If the answer is 'Yes', then we also output 'Yes' as an answer to CNF-SAT; otherwise, we output 'No'.

Using Lemma 3.11, we can see that if there is an assignment a such that $C[a] = \mathsf{True}$, then for the corresponding realisation, the Fréchet distance is $(1 + \varepsilon) \cdot {}^2/\sqrt{5}$; the other way around, if for some realisation we get the distance $(1 + \varepsilon) \cdot {}^2/\sqrt{5}$, then by our construction, all the clauses are satisfied and $C[a] = \mathsf{True}$; and so $d_{\mathrm{F}}^{\max}(\mathrm{VC}, \mathrm{CC}) = (1 + \varepsilon) \cdot {}^2/\sqrt{5}$. On the other hand, if there is no such assignment a, then for any assignment a, there is some C_i with $C_i[a] = \mathsf{False}$, yielding $C[a] = \mathsf{False}$, and also for any realisation of VC, there is some gadget ACG_i that yields the Fréchet distance of 1; and so $d_{\mathrm{F}}^{\max}(\mathrm{VC}, \mathrm{CC}) = 1$. Therefore, the formula C is satisfiable if and only if $d_{\mathrm{F}}^{\max}(\mathrm{VC}, \mathrm{CC}) > 1$, and so our answer to the CNF-SAT instance is correct.

Furthermore, as before, the instance of UPPER BOUND DISCRETE FRÉCHET that gives the answer to CNF-SAT can be constructed in polynomial time. Thus, we conclude that UPPER BOUND CONTINUOUS FRÉCHET for indecisive curves is NP-hard; combining it with the first part of the proof shows that it is NP-complete. ☐

Expected Fréchet Distance on Indecisive Points

We show that finding the expected discrete Fréchet distance is #P-hard under the uniform distribution by providing a polynomial-time reduction from #CNF-SAT, i.e. the problem of finding the number of satisfying assignments to a CNF-SAT formula. Define the following problem and its continuous counterpart.

Problem 3.13. EXPECTED DISCRETE FRÉCHET: Find $d_{\mathrm{dF}}^{\mathbb{E}(\mathbb{U})}(\mathcal{U}, \mathcal{V})$ for uncertain curves $\mathcal{U}, \mathcal{V}$.

Problem 3.14. EXPECTED CONTINUOUS FRÉCHET: Find $d_{\mathrm{F}}^{\mathbb{E}(\mathbb{U})}(\mathcal{U}, \mathcal{V})$ for uncertain curves $\mathcal{U}, \mathcal{V}$.

The main idea is to derive an expression for the number of satisfying assignments in terms of $d_{\mathrm{dF}}^{\mathbb{E}(\mathbb{U})}(\mathrm{VC}, \mathrm{CC})$. This works, since there is a one-to-one correspondence between boolean variable assignment and a choice of realisation of VC, so counting the number of satisfying assignments corresponds to finding the proportion of realisations yielding a large Fréchet distance. We can establish the result for EXPECTED CONTINUOUS FRÉCHET similarly.

Theorem 3.15. *The problems* EXPECTED DISCRETE FRÉCHET *and* EXPECTED CONTINUOUS FRÉCHET *for indecisive curves are #P-hard.*

Proof. Suppose we are given an instance of the #CNF-SAT problem, i.e. a CNF-SAT formula C with n clauses and m variables. Denote the (unknown) number of satisfying assignments of C by N. We can construct the curves VC and CC in the same way as previously. We then get an instance of EXPECTED DISCRETE FRÉCHET on indecisive curves under the uniform distribution. Assuming we solve it and get $d_{\mathrm{dF}}^{\mathbb{E}(\mathbb{U})}(\mathrm{VC}, \mathrm{CC}) = \mu$, we can now compute N:

$$N = (\mu - 1) \cdot \frac{2^m}{\varepsilon}\,.$$

N is then the output for the instance of #CNF-SAT that we were given. Clearly, construction of the curves can be done in polynomial time; so can the computation of N; hence, the reduction takes polynomial time.

We still need to show that the result we obtain is correct. For each assignment, there is exactly one realisation of the curve VC. Furthermore, as we choose the realisation of each indecisive point uniformly and independently, all the realisations of VC have equal probability of 2^{-m}. There are N satisfying assignments; and each of the corresponding realisations yields the discrete Fréchet distance of $1 + \varepsilon$. In the remaining $2^m - N$ cases, the distance is 1. Using the definition of expected value, we can derive

$$\mu = d_{\mathrm{dF}}^{\mathbb{E}(\mathbb{U})}(\mathrm{VC}, \mathrm{CC}) = N \cdot 2^{-m} \cdot (1 + \varepsilon) + (2^m - N) \cdot 2^{-m} \cdot 1 = 1 + \frac{N \cdot \varepsilon}{2^m}\,.$$

Then it is easy to see that indeed $N = (\mu - 1) \cdot \frac{2^m}{\varepsilon}$. So, we get the correct number of satisfying assignments, if we know the expected value under the uniform distribution. Therefore, EXPECTED DISCRETE FRÉCHET for indecisive curves is #P-hard.

One can derive a very similar formula to show that EXPECTED CONTINUOUS FRÉCHET is also #P-hard for indecisive curves. We can use almost the same reduction as for the discrete case, so given an instance of #CNF-SAT (CNF-SAT formula C with n clauses and m variables), we construct the two curves, solve EXPECTED CONTINUOUS FRÉCHET to obtain the value of μ, and compute

$$N = 2^m \cdot (\mu - 1) \cdot \frac{\sqrt{5}}{2(1 + \varepsilon) - \sqrt{5}}$$

as the output for #CNF-SAT. To show that the output is correct, note that

$$\mu = 2^{-m} \cdot N \cdot \frac{2}{\sqrt{5}} \cdot (1 + \varepsilon) + 2^{-m} \cdot (2^m - N) \cdot 1 = 1 + 2^{-m} \cdot N \cdot \left(\frac{2}{\sqrt{5}} (1 + \varepsilon) - 1 \right),$$

so we can express N as

$$N = 2^m \cdot (\mu - 1) \cdot \frac{\sqrt{5}}{2(1 + \varepsilon) - \sqrt{5}}\,.$$

Again, the reduction is correct and can be done in polynomial time, so EXPECTED CONTINUOUS FRÉCHET for indecisive curves is #P-hard. □

We use the uniform distribution; however, we only need to compute the probability of picking a realisation that corresponds to a satisfying assignment, $N \cdot 2^{-m}$ above. If we can do so for a different distribution, then the rest of the proof does not require modifications to show #P-hardness.

Upper Bound Discrete Fréchet Distance on Imprecise Points

Here we consider imprecise points modelled as disks and as line segments; the results and their proofs turn out to be very similar. We denote the disk with the centre at $p \in \mathbb{R}^d$ and radius $r \geq 0$ as $D(p, r)$. We denote the line segment between points p_1 and p_2 by $S(p_1, p_2)$.

Disks. We use a construction very similar to that of the indecisive points case, except now we change the gadget containing a non-degenerate indecisive point so that it contains a non-degenerate imprecise point, for all $j \in [m]$:

$$\mathrm{VG}_j = D((0, 0), 0.5 + \varepsilon) \sqcup (2, 0)\,.$$

Essentially, the two original indecisive points are now located on the points realising the diameter of the disk.

We can reuse the proof leading up to Theorem 3.8 if we can show the following.

Lemma 3.16. *Suppose* $d_{\mathrm{dF}}^{\max}(\mathrm{VC}, \mathrm{CC}) = \nu$. *If one considers all the realisations* π *of* VC *that yield* $d_{\mathrm{dF}}(\pi, \mathrm{CC}) = \nu$, *then among them there will always be a realisation that only places the imprecise point realisations at either* $(0, 0.5 + \varepsilon)$ *or* $(0, -0.5 - \varepsilon)$.

Proof. First of all, note that the points $(2, 0)$ and $(1, 0)$ are still in the curves in the same quality as before, so they must be coupled, and hence the lowest discrete Fréchet distance achievable with any realisation is 1.

Now consider a realisation of an imprecise point. Suppose that all the clause assignment points for that imprecise point are placed at $(0, -0.5)$. Then geometrically it is obvious that the distance is maximised by placing the realisation at $(0, 0.5 + \varepsilon)$; if there is a realisation that achieves the best possible value ν without doing this, then we can move this point and still get ν.

Suppose that some clause assignment points are at $(0, -0.5)$ and some at $(0, 0.5)$. As the realisation comes from the disk of radius $0.5 + \varepsilon$, there is no realisation that is further than 1 away from both assignment points; therefore, to maximise the distance we have to choose one of the two locations, and then the previous case applies.

So, it is clear that, from an arbitrary optimal realisation, moving to the (correct) indecisive point realisation will still yield an optimal realisation for the maximum discrete Fréchet distance; thus, the statement of the lemma holds. ☐

Line segments. We change the gadget to be, for all $j \in [m]$:

$$\mathrm{VG}_j = S((0, -0.5 - \varepsilon), (0, 0.5 + \varepsilon)) \sqcup (2, 0)\,.$$

Again, the two original indecisive points are now located on the ends of the segment; moreover, the segment is a strict subset of the disk.

We can state a similar lemma.

Lemma 3.17. *Suppose* $d_{\mathrm{dF}}^{\max}(\mathrm{VC}, \mathrm{CC}) = \nu$. *If one considers all the realisations* π *of* VC *that yield* $d_{\mathrm{dF}}(\pi, \mathrm{CC}) = \nu$, *then among them there will always be a realisation that only places the imprecise point realisations at either* $(0, 0.5 + \varepsilon)$ *or* $(0, -0.5 - \varepsilon)$.

Proof. Since the line segments include these points and are subsets of the disks, the statement of Lemma 3.16 immediately yields this result. □

Now we can state the following theorem for both models.

Theorem 3.18. *The problem* UPPER BOUND DISCRETE FRÉCHET *for imprecise curves modelled as line segments or as disks is NP-hard.*

Proof. As shown in Lemma 3.16 and Lemma 3.17, for the same CNF-SAT formula, the upper bound discrete Fréchet distance on indecisive and imprecise points is equal for our construction. So, trivially, UPPER BOUND DISCRETE FRÉCHET is NP-hard for imprecise curves. □

Upper Bound Fréchet Distance on Imprecise Points

We use exactly the same construction as in the previous section. The argument here follows the previous ones very closely, so we can immediately state the following theorem.

Theorem 3.19. *The problem* UPPER BOUND CONTINUOUS FRÉCHET *for imprecise curves modelled as line segments or as disks is NP-hard.*

Proof. Note that we can apply the same arguments as in Lemma 3.16 and Lemma 3.17 to reduce this problem to the one on indecisive points. Then, we can apply the same argument as in the proof of Theorem 3.18 to conclude that the problem is NP-hard. □

Expected Discrete Fréchet Distance on Imprecise Points

We can also consider the value of the expected Fréchet distance on imprecise points. We show the result only for points modelled as line segments; in principle, we believe that for disks a similar result holds, but the specifics of our reduction do not allow for clean computations.

We cannot immediately use our construction: we treat subsegments at the ends of the imprecision segments as `True` and `False`, but we have no interpretation for points in the centre part of a segment. So, we want to separate the realisations that pick any such invalid points. To that aim, we introduce extra gadgets to the clause curve that act as clauses, but catch these invalid realisations, so each of them yields the

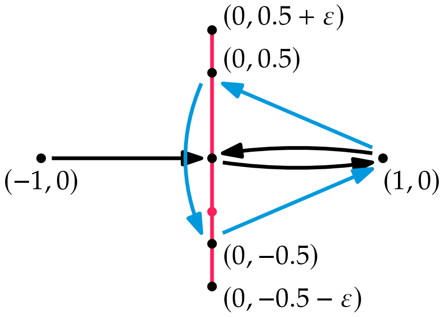


Figure 3.4. The curve FG_j hops between $(0, 0)$ and $(1, 0)$ for every variable x_k (in black) except when $k = j$; in the latter case, the curve goes to $(0, 0.5)$, $(0, -0.5)$, and back to $(1, 0)$ (in blue). Consider the line segment on the variable curve representing x_j (in red). As a consequence, for any realisation of the variable clause gadget such that the realisation of x_j falls within $S((0, -0.5), (0, 0.5))$, the gadget FG_j can be aligned with VCG to obtain Fréchet distance 1.

distance of 1. Now we have three distinct cases: realisation is satisfying, non-satisfying, or invalid. For every $j \in [m]$, define

$$\begin{aligned}\mathrm{FG}_j = (-1, 0) &\sqcup \bigsqcup_{k\in[j-1]} \Big((0, 0) \sqcup (1, 0)\Big)\\ &\sqcup (0, 0.5) \sqcup (0, -0.5) \sqcup (1, 0)\\ &\sqcup \bigsqcup_{k\in[m]\setminus[j]} \Big((0, 0) \sqcup (1, 0)\Big)\,.\end{aligned}$$

We define a clause gadget that ignores all the variables except for x_j and then features both 'true' and 'false' for x_j. The intuition is that any realisation corresponding to the invalid state of a variable will be close to both $(0, 0.5)$ and $(0, -0.5)$, and every other variable value is close to $(0, 0)$, so aligning the gadget FG_j with the variable curve will yield a small Fréchet distance if x_j is in an invalid state. See also Figure 3.4. We then define the clause curve as

$$\mathrm{CC} = \bigsqcup_{i\in[n]} \mathrm{ACG}_i \sqcup \bigsqcup_{j\in[m]} \mathrm{FG}_j\,.$$

We can now choose to couple one of the FG clauses to the variable curve. As before, due to the synchronisation points we can never get the Fréchet distance below 1. If one of the realisations x_j of the segments falls into the interval $[(0, -0.5), (0, 0.5)]$, then it will be not further away than 1 from both the corresponding points on FG_j; all the other points, being in the middle at $(0, 0)$, are guaranteed to be at most $0.5 + \varepsilon < 1$ away from their coupled point; so, the one-to-one coupling[2] will yield the discrete Fréchet distance of 1; thus, the optimal discrete Fréchet distance in this case is 1. Therefore, we only need to consider the situations when all the realisations happen to fall in either the interval $((0, 0.5), (0, 0.5 + \varepsilon)]$ or $[(0, -0.5 - \varepsilon), (0, -0.5))$. We will treat the first interval as `True` and the second interval as `False`. Denote the number of satisfying assignments by N. To find the expression for the expected discrete Fréchet distance, we need to consider three cases.

- At least one realisation of m variables falls within the y-interval $[-0.5, 0.5]$. Note that the realisation on each segment is uniform and independent of other segments. Under the uniform distribution, we get

  $$\begin{aligned}&\Pr[\text{at least one realisation from } [-0.5, 0.5]]\\ &= 1 - \prod_{j \in [m]} \frac{2\varepsilon}{1 + 2\varepsilon} = 1 - \left(\frac{2\varepsilon}{1 + 2\varepsilon}\right)^m .\end{aligned}$$

  Note that in each such case we get the discrete Fréchet distance of 1, as discussed before.

- All realisations fall outside the y-interval $[-0.5, 0.5]$, and they correspond to a non-satisfying assignment. Each specific non-satisfying assignment corresponds to picking values on the specific interval, either $((0, 0.5), (0, 0.5 + \varepsilon)]$ or $[(0, -0.5 - \varepsilon), (0, -0.5))$, so:

  $$\Pr[\text{specific assignment}] = \prod_{j \in [m]} \frac{\varepsilon}{1 + 2\varepsilon} = \left(\frac{\varepsilon}{1 + 2\varepsilon}\right)^m .$$

  There are $2^m - N$ such assignments, and each of them contributes the value of 1.

- All realisations fall outside the y-interval $[-0.5, 0.5]$, and they correspond to a satisfying assignment. Again, the probability of getting a particular assignment is $(\varepsilon/(1 + 2\varepsilon))^m$, and there are N such

---

[2] Technically, it is one to one on all points except the realisation corresponding to x_j; that one has to be coupled to both $(0, 0.5)$ and $(0, -0.5)$ in FG_j.

assignments. Now they contribute values distinct from 1; still, the optimum is contributed by one of the new clauses, and then it will be defined by the realisation closest to $(0, 0)$. We prove this in Lemma 3.20. Assuming this statement is true, we need to find $\mathbb{E}[\min_{j\in[m]}(1 + \varepsilon'_j)]$ with ε'_j sampled uniformly from $(0, \varepsilon]$; we can rephrase this to $1 + \varepsilon \cdot \mathbb{E}[\min_{j\in[m]} u_j]$ with u_j sampled uniformly from $(0, 1]$. It is a standard result that the minimum now is geometrically distributed, so we get $\mathbb{E}[\min_{j\in[m]} u_j] = {}^{1}/_{(1+m)}$, and hence the expected contribution is $1 + {}^{\varepsilon}/_{(1+m)}$.

Lemma 3.20. *Consider some realisation* $\pi \Subset \mathrm{VC}$ *where each value can be interpreted either as* `True` *or* `False` *and the corresponding assignment satisfies the formula. Pick* j *such that the subcurve of* π *realising* VG_j *contains the point closest to* $(0, 0)$*, at location* $(0, 0.5 + \varepsilon')$ *or* $(0, -0.5 - \varepsilon')$ *for some* $\varepsilon' > 0$*. Then the optimal coupling establishes a coupling between* π *and* FG_j*, and the discrete Fréchet distance is* $d_{\mathrm{dF}}(\pi, \mathrm{CC}) = 1 + \varepsilon'$*.*

Proof. First of all, note that we still have to couple the synchronisation points and we cannot have the discrete Fréchet distance below 1. Thus, we need to consider only the couplings of π with the gadgets of CC. Note that if we couple FG_j to π, we get the discrete Fréchet distance of $1 + \varepsilon'$. Recall that we consider only satisfying assignments, so, if we consider an arbitrary subcurve ACG_i, then there is some variable x_ℓ that satisfies the corresponding clause, and so the realisation of that variable is $1 + \varepsilon''$ away from the corresponding assignment point. Therefore, such a coupling will yield the discrete Fréchet distance of $1 + \varepsilon'' \geq 1 + \varepsilon'$. Finally, it is easy to see that choosing some FG_k with $k \neq j$ will also yield some distance $1 + \varepsilon'' \geq 1 + \varepsilon'$. So, the statement of the lemma holds. □

We can bring the three cases together to find

$$\begin{aligned}
&d_{\mathrm{dF}}^{\mathbb{E}(\mathbb{U})}(\mathrm{VC}, \mathrm{CC}) \\
&\quad = 1 \cdot \left(1 - \left(\frac{2\varepsilon}{1 + 2\varepsilon}\right)^m\right) + 1 \cdot (2^m - N) \cdot \left(\frac{\varepsilon}{1 + 2\varepsilon}\right)^m \\
&\qquad + \left(1 + \frac{\varepsilon}{1 + m}\right) \cdot N \cdot \left(\frac{\varepsilon}{1 + 2\varepsilon}\right)^m \\
&\quad = 1 + N \cdot \frac{\varepsilon^{m+1}}{(1 + m) \cdot (1 + 2\varepsilon)^m} .
\end{aligned}$$

So, if we were to compute $d_{\mathrm{dF}}^{\mathbb{E}(\mathbb{U})}(\mathrm{VC}, \mathrm{CC}) = \mu$, then the number of satisfying assignments would be

$$N = (\mu - 1) \cdot \frac{(1 + m) \cdot (1 + 2\varepsilon)^m}{\varepsilon^{m+1}} .$$

This is easy to compute in polynomial time, and our construction can still be done in polynomial time; hence, the result follows.

Theorem 3.21. *The problem* Expected Discrete Fréchet *for imprecise curves modelled as line segments under the uniform distribution is #P-hard.*

We have stated the results for the uniform distribution; however, we conjecture that this construction could work for some other distributions. The requirements are that we need to be able to compute the probabilities of falling into each region; that all realisations are equiprobable, or we have some other way to compute the probability of getting a satisfying realisation; and that we can compute $\mathbb{E}[\min_{j \in [m]} u_j]$ under the appropriate distribution of u_j.

3.1.2 Lower Bound Fréchet Distance

In this section, we prove that computing the lower bound continuous Fréchet distance is NP-hard for uncertainty modelled with line segments. This contrasts with the algorithm for indecisive curves, given in Section 3.2.1, and with the algorithm previously suggested by Ahn el al. [18] for the discrete Fréchet distance. Unlike the upper bound proofs, this reduction uses the NP-hard problem Subset-Sum. We consider the following problems.

Problem 3.22. Lower Bound Continuous Fréchet: Given an uncertain curve $\mathcal{U}$ with m vertices, a polygonal curve σ with n vertices, and a threshold $\delta > 0$, decide if $d_{\mathrm{F}}^{\min}(\mathcal{U}, \sigma) \leq \delta$.

Problem 3.23. Subset-Sum: Given a set $S = \{s_1, \dots, s_n\}$ of n positive integers and a target integer τ, decide if there exists an index set I such that $\sum_{i \in I} s_i = \tau$.

As a polygonal curve is an uncertain curve, proving Problem 3.22 is NP-hard implies the corresponding problem with two uncertain curves is also NP-hard.

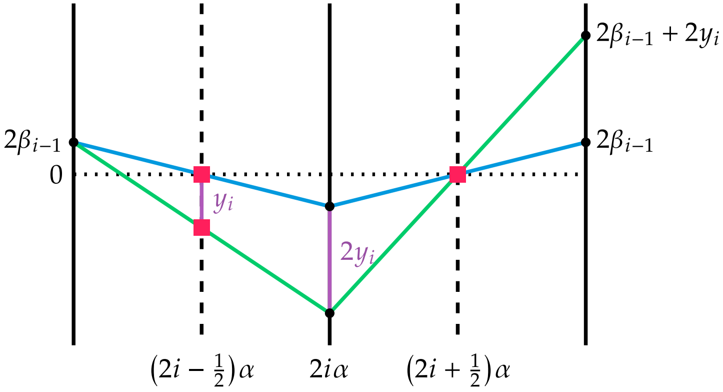


Figure 3.5. Passing through $\big((2i - {}^1\!/\!_2)\alpha, 0\big)$ does not change the height, and passing through $\big((2i - {}^1\!/\!_2)\alpha, -y_i\big)$ adds $2y_i$.

An Intermediate Problem

We start by reducing Subset-Sum to a more geometric intermediate curve-based problem.

Definition 3.24. Let $\alpha > 0$ be some value, and let $\pi = \langle \pi_1, \dots, \pi_{2n+1} \rangle$ be a polygonal curve. We call π an *α-regular curve* if for all $i \in [2n+1]$, the x-coordinate of π_i is $i \cdot \alpha$. Let $Y = \{y_1, \dots, y_n\}$ be a set of n positive integers. Call π a *Y-respecting curve* if:

1. For all $i \in [n]$, π passes through the point $\big((2i + {}^1\!/\!_2)\alpha, 0\big)$.
2. For all $i \in [n]$, π either passes through the point $\big((2i - {}^1\!/\!_2)\alpha, 0\big)$ or $\big((2i - {}^1\!/\!_2)\alpha, -y_i\big)$.

Intuitively, Definition 3.24 requires π to pass through $\big((2i + {}^1\!/\!_2)\alpha, 0\big)$ as it reflects the y-coordinate about the line $y = 0$ (see Figure 3.5). Thus, if the curve also passes through $\big((2i - {}^1\!/\!_2)\alpha, 0\big)$, the two reflections cancel each other. If it passes through $\big((2i - {}^1\!/\!_2)\alpha, -y_i\big)$, the lemma below argues that y_i shows up in the final vertex height.

Lemma 3.25. *Let π be a Y-respecting α-regular curve, and let I be the subset of indices such that π passes through $\big((2i - {}^1\!/\!_2)\alpha, -y_i\big)$ for all $i \in I$. If $\pi_1 = (\alpha, 0)$, then $\pi_{2n+1} = \big((2n+1)\alpha, 2\sum_{i \in I} y_i\big)$.*

Proof. For $j \in [n]$, let $I_j = \{i \in I \mid i \leq j\}$, and let $\beta_j = \sum_{i \in I_j} y_i$, where $\beta_0 = 0$. We argue by induction that $\pi_{2j+1} = \big((2j+1)\alpha, 2\beta_j\big)$, thus yielding the lemma statement when $j = n$. For the base case, $j = 0$, the statement becomes $\pi_1 = (\alpha, 0)$, which is true by assumption of the lemma.

Assume that $\pi_{2j-1} = \big((2j-1)\alpha, 2\beta_{j-1}\big)$. Suppose that $j \notin I$. In this case, since π is Y-respecting, it passes through points $\big((2j - 1/2)\alpha, 0\big)$ and $\big((2j + 1/2)\alpha, 0\big)$. This implies $\pi_{2j} = (2j\alpha, -2\beta_{j-1})$ and $\pi_{2j+1} = \big((2j + 1)\alpha, 2\beta_{j-1}\big) = \big((2j+1)\alpha, 2\beta_j\big)$. Now suppose that $j \in I$. In this case, it must pass through points $\big((2j - 1/2)\alpha, -y_j\big)$ and $\big((2j + 1/2)\alpha, 0\big)$. This implies $\pi_{2j} = \big(2j\alpha, 2\beta_{j-1} - 2 \cdot (2\beta_{j-1} + y_j)\big) = \big(2j\alpha, -2(\beta_{j-1} + y_j)\big)$ and $\pi_{2j+1} = \big((2j+1)\alpha, 2(\beta_{i-1} + y_j)\big) = \big((2j+1)\alpha, 2\beta_j\big)$. See Figure 3.5. □

The following corollary is needed in the next section, and follows from Lemma 3.25.

Corollary 3.26. *For a set $Y = \{y_1, \ldots, y_n\}$, let $M = \sum_{i=1}^{n} y_i$. For any vertex π_i of a Y-respecting α-regular curve, its y-coordinate is at most $2M$ and at least $-2M$.*

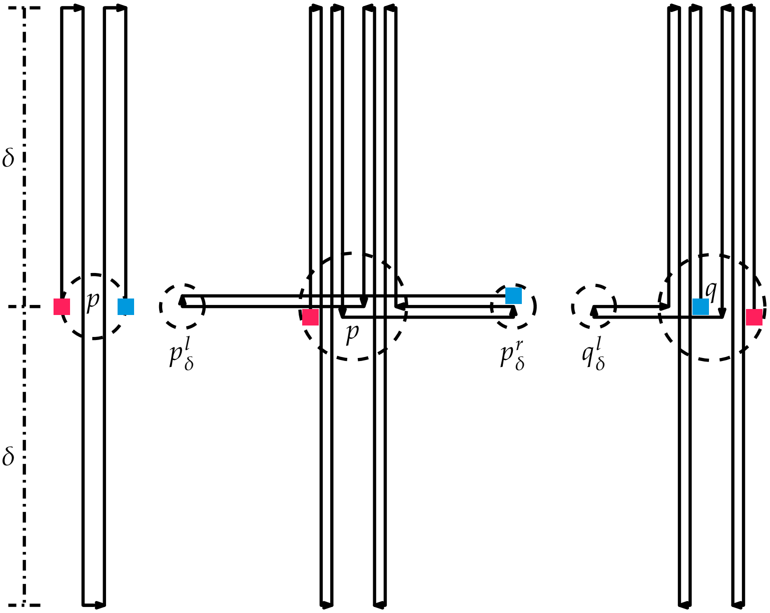
Problem 3.27. RR-Curve: Given a set $Y = \{y_1, \ldots, y_n\}$ of n positive integers, a value $\alpha = \alpha(Y) > 0$, and an integer τ, decide if there is a Y-respecting α-regular curve $\pi = \langle \pi_1, \ldots, \pi_{2n+1} \rangle$ such that $\pi_1 = (\alpha, 0)$ and $\pi_{2n+1} = \big((2n+1)\alpha, 2\tau\big)$.

By Lemma 3.25, Subset-Sum immediately reduces to this problem by setting $Y = S$. Note that for this reduction, it suffices to use any positive constant for α; however, we allow α to depend on Y, as this is ultimately required in our reduction to Problem 3.22.

Theorem 3.28. *For any $\alpha(Y) > 0$, RR-Curve is NP-hard.*

Reduction to Lower Bound Fréchet Distance

Let α, τ, $Y = \{y_1, \ldots, y_n\}$ be an instance of RR-Curve. In this section, we show how to reduce it to an instance $(\delta, \mathcal{U}, \sigma)$ of Problem 3.22, where the uncertain regions in $\mathcal{U}$ are vertical line segments. The main idea is to use $\mathcal{U}$ to define an α-regular curve, and to use σ to enforce that it is Y-respecting. Let $M = \sum_{i=1}^{n} y_i$. Then $\mathcal{U} = \langle V_1, \ldots, V_{2n+1} \rangle$, where V_i is a vertical segment whose horizontal coordinate is $i \cdot \alpha$ and whose vertical extent is the interval $[-2M, 2M]$. By Corollary 3.26, we have the following simple observation.



(a) $g_\delta(p)$ **(b)** $\mathrm{lcg}_\delta(p)$ **(c)** $\mathrm{ucg}_\delta(q)$

Figure 3.6. Depiction of the gadgets $g_\delta(p)$, $\mathrm{lcg}_\delta(p)$, and $\mathrm{ucg}_\delta(p)$. Dashed circles represent zero-area points; the red (blue) square represents the starting (ending) point.

Observation 3.29. *The set of all Y-respecting α-regular curves is a subset of* $\mathrm{Real}(\mathcal{U})$*.*

Thus, the main challenge is to define σ to enforce that the realisation is Y-respecting. To that end, we first describe a gadget forcing the realisation to pass through a specified point.

Definition 3.30. For any point $p = (x, y) \in \mathbb{R}^2$ and value $\delta > 0$, let the *δ-gadget* at p, denoted by $g_\delta(p)$, be the curve $(x, y) \sqcup (x, y + \delta) \sqcup (x, y - \delta) \sqcup (x, y + \delta) \sqcup (x, y)$. See Figure 3.6a.

Lemma 3.31. *Let $p = (x, y) \in \mathbb{R}^2$ be a point, and let S be any line segment. If $d_F(S, g_\delta(p)) \leq \delta$, then S must pass through p.*

Proof. In order, $g_\delta(p)$ visits the points $(x, y + \delta)$, $(x, y - \delta)$, and $(x, y + \delta)$. Let $a = (a_x, a_y)$, $b = (b_x, b_y)$, $c = (c_x, c_y)$ be the points from S which get aligned with these respective points under an optimal Fréchet alignment. If the Fréchet distance is at most δ, then $b_y \leq y \leq a_y, c_y$. If S has a positive slope w.r.t. the order along S, then also $c_y \geq b_y \geq a_y$, so we have $a_y = b_y$ and so $a = b$. However, if $a = b$, then this point must be p itself, as p is the only point with distance at most δ from both $(x, y + \delta)$ and $(x, y - \delta)$. If S has a negative slope, then $c_y \leq b_y \leq a_y$, so now $b_y = c_y$ and $b = c$, and again this must be point p. Finally, if S is horizontal, then $a = b = c = p$, as this is the only point on a horizontal segment aligned with both $(x, y + \delta)$ and $(x, y - \delta)$. $\square$

For our uncertain curve to be Y-respecting, it must pass through all the points of the form $\big((2i + {}^1\!/_2)\alpha, 0\big)$. This condition is satisfied by placing a δ-gadget at each such point, as follows from Lemma 3.31. The second condition for a curve to be Y-respecting is that it passes through $\big((2i - {}^1\!/_2)\alpha, 0\big)$ or $\big((2i - {}^1\!/_2)\alpha, -y_i\big)$. This condition is much harder to encode, and requires putting several δ-gadgets together to create a composite gadget.

Definition 3.32. For any point $p = (x, y) \in \mathbb{R}^2$ and value $\delta > 0$, let $p^l_\delta = (x - {}^\delta\!/_2, y)$ and $p^r_\delta = (x + {}^\delta\!/_2, y)$. Define the *$\delta$-lower composite gadget* at p, denoted $\mathrm{lcg}_\delta(p)$, to be the curve $\mathrm{g}_\delta(p) \sqcup p^r_\delta \sqcup \mathrm{g}_\delta(p) \sqcup p^l_\delta \sqcup p^r_\delta$. See Figure 3.6b. Define the *δ-upper composite gadget* at q, denoted $\mathrm{ucg}_\delta(q)$, to be the curve $\mathrm{g}_\delta(q) \sqcup q^l_\delta \sqcup \mathrm{g}_\delta(q)$. See Figure 3.6c. Define the *δ-composite gadget* of p and q, denoted $\mathrm{cg}_\delta(p, q)$, to be the curve $\mathrm{lcg}_\delta(p) \sqcup \mathrm{ucg}_\delta(q)$.

To use the composite gadget, we centre the lower gadget at height $-y_i$ and place the upper gadget directly above it at height zero. As the two gadgets are on top of each other, ultimately, we require our uncertain curve to go back and forth once between consecutive vertical line segments; we have the following key property.

Lemma 3.33. *Let $p = (p_x, -p_y)$ and $q = (p_x, 0)$ be points in $\mathbb{R}^2$. Let $\pi = \langle a, b, c, d\rangle$ be a three-segment curve such that $b_x > p_x + \delta$ and $c_x < p_x - \delta$. If $d_{\mathrm{F}}(\pi, \mathrm{cg}_\delta(p, q)) \leq \delta$, then:*

1. *the segment ab must pass through p,*
2. *the segment cd must pass through q, and*
3. *the segment bc must either pass through p or through q.*

Proof. Recall from Definition 3.32 that $\mathrm{cg}_\delta(p,q) = \mathrm{g}_\delta(p) \sqcup p^r_\delta \sqcup \mathrm{g}_\delta(p) \sqcup p^l_\delta \sqcup p^r_\delta \sqcup \mathrm{g}_\delta(q) \sqcup q^l_\delta \sqcup \mathrm{g}_\delta(q)$, and that the gadgets $\mathrm{g}_\delta(p)$ and $\mathrm{g}_\delta(q)$ lie entirely on the vertical line at $p_x = q_x$. Thus, as $b_x > p_x + \delta$ and $c_x < p_x - \delta$, each occurrence of $\mathrm{g}_\delta(p)$ or $\mathrm{g}_\delta(q)$ in $\mathrm{cg}_\delta(p,q)$ must map either entirely before or after b, and similarly entirely before or after c.

Moreover, as $\mathrm{cg}_\delta(p,q)$ starts with $\mathrm{g}_\delta(p)$ and $b_x > p_x + \delta$, this implies that a maps to p and $\mathrm{g}_\delta(p)$ maps to a subsegment of ab, which by Lemma 3.31 implies that ab passes through p. Similarly, as $\mathrm{cg}_\delta(p,q)$ ends with $\mathrm{g}_\delta(q)$ and $c_x < q_x - \delta$, cd passes through q.

Finally, the portion of $\mathrm{cg}_\delta(p,q)$ that maps to the segment bc must contain a point on the vertical line at $p_x = q_x$ (since $b_x > p_x + \delta$ and $c_x < p_x - \delta$). By the construction of $\mathrm{cg}_\delta(p,q)$, this point must lie on one of the (middle) $\mathrm{g}_\delta(p)$ or $\mathrm{g}_\delta(q)$ gadgets. As we already argued, such gadgets must map entirely to one side of b or c, so Lemma 3.31 implies that bc must pass through p or q. □

As bc shares an endpoint with ab and cd, the following corollary is immediate. It is used later to argue that while our uncertain curve goes back and forth between consecutive vertical lines, it defines an α-regular curve. (See Figure 3.7 used for Theorem 3.36.)

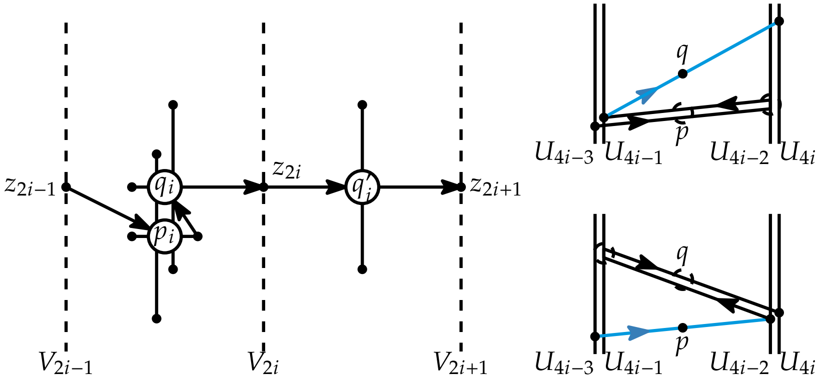
Corollary 3.34. *If $d_\mathrm{F}(\pi, \mathrm{cg}_\delta(p,q)) \le \delta$, then either ab and bc are on the same line, or cd and bc are on the same line.*

The following lemma acts as a rough converse of Lemma 3.33.

Lemma 3.35. *Let $p = (p_x, -p_y)$ and $q = (p_x, 0)$ be points in $\mathbb{R}^2$, with $p_y \le \delta/4$. Let $\pi = \langle p, b, c, q\rangle$ be a curve such that $p_x + \delta < b_x \le p_x + 1.1\delta$, $p_x - 1.1\delta \le c_x < p_x - \delta$, and $-\delta/2 \le b_y, c_y \le \delta/2$. If bc passes through either p or q, then $d_\mathrm{F}(\pi, \mathrm{cg}_\delta(p,q)) \le \delta$.*

Proof. Recall that $\mathrm{cg}_\delta(p,q) = \mathrm{g}_\delta(p) \sqcup p^r_\delta \sqcup \mathrm{g}_\delta(p) \sqcup p^l_\delta \sqcup p^r_\delta \sqcup \mathrm{g}_\delta(q) \sqcup q^l_\delta \sqcup \mathrm{g}_\delta(q)$. First, observe that all the points on the prefix $\mathrm{g}_\delta(p) \sqcup p^r_\delta$ of $\mathrm{cg}_\delta(p,q)$ are at most δ away from p, and thus can all be mapped to the starting point of π. Similarly, all points on the suffix $q^l_\delta \sqcup \mathrm{g}_\delta(q)$ of $\mathrm{cg}_\delta(p,q)$ are at most δ away from q, and thus can all be mapped to the ending point of π. Thus, it suffices to argue that $d_\mathrm{F}(\pi, \sigma) \le \delta$, where $\sigma = p^r_\delta \sqcup \mathrm{g}_\delta(p) \sqcup p^l_\delta \sqcup p^r_\delta \sqcup \mathrm{g}_\delta(q) \sqcup q^l_\delta$.

It is easiest to describe the rest of the matching in a similar manner, that is, as an alternating sequence of moves, where we stand still at a



(a) Pictorial representation of λ_i.

(b) The two solutions.

Figure 3.7. On the left, λ_i. On the right, the two possible solutions with the Fréchet distance at most δ. The top (resp. bottom) corresponds to an α-regular curve passing through q (resp. p).

single point on one curve while moving along a contiguous subcurve from the other curve, and then switching curves. We now describe this sequence, which differs based on whether bc passes through p or q. Ultimately, the matchings are valid, since for each move, all points on the subcurve have distance at most δ to the fixed point on the other curve. Thus, we now simply describe the moves without reiterating this property (distance at most δ) which is validating each move.

First suppose that bc passes through p, then $\pi = \langle p, b, p, c, q \rangle$. In this case, we first map the prefix $\langle p, b, p \rangle$ of π to p^r_δ. Next, we map the prefix $p^r_\delta \sqcup g_\delta(p) \sqcup p^l_\delta$ of σ to p. Then we map the suffix $\langle p, c, q \rangle$ of π to p^l_δ. Finally, we map the suffix $p^l_\delta \sqcup p^r_\delta \sqcup g_\delta(q) \sqcup q^l_\delta$ of σ to q.

Now suppose that bc passes through q, then $\pi = \langle p, b, q, c, q \rangle$. In this case, we first map the prefix $p^r_\delta \sqcup g_\delta(p) \sqcup p^l_\delta \sqcup p^r_\delta$ of σ to p. Next, we map the prefix $\langle p, b, q \rangle$ of π to p^r_δ. Then we map the suffix $p^r_\delta \sqcup g_\delta(q) \sqcup q^l_\delta$ of σ to q. Finally, we map the suffix $\langle q, c, q \rangle$ of π to q^l_δ. □

Theorem 3.36. *Lower Bound Continuous Fréchet (Problem 3.22) is NP-hard, even when the uncertain regions are all equal-length vertical segments with the same height and the same horizontal distance (to the left or right) between adjacent uncertain regions.*

Proof. To prove NP-hardness, we give a reduction from RR-Curve, which is NP-hard by Theorem 3.28. Let $\alpha(Y), \tau, Y = \{y_1, \ldots, y_n\}$ be an instance of RR-Curve. For the reduction we set $\delta = 4M$, where $M = \sum_{i=1}^{n} y_i$. Note that Theorem 3.28 allows us to choose how to set $\alpha(Y)$, and in particular we set $\alpha = 2.1\delta = 8.4M$. (More precisely, the properties we need are that $\alpha > 2\delta$ and $\delta \geq 4M$.) We now describe how to construct $\mathcal{U}$ and σ.

Let $V = \{V_1, \ldots, V_{2n+1}\}$ be a set of vertical line segments where all upper (resp. lower) endpoints of the segments have height $2M$ (resp. $-2M$), and for all i, the x-coordinate of V_i is $i\alpha$. Let $\mathcal{U} = \langle U_1, \ldots, U_{4n+1} \rangle$ be the uncertain curve such that $U_{4n+1} = V_{2n+1}$, and for all $i \in [n]$, $U_{4i-3} = V_{2i-1}$, $U_{4i-2} = V_{2i}$, $U_{4i-1} = V_{2i-1}$, and $U_{4i} = V_{2i}$. For $i \in [2n+1]$, define the points $z_i = (i\alpha, 0)$, and for $i \in [n]$, define $q_i = \big((2i - 1/2)\alpha, 0\big)$, $q'_i = \big((2i + 1/2)\alpha, 0\big)$, and $p_i = \big((2i - 1/2)\alpha, -y_i\big)$. For a given value $i \in [n]$, consider the curve $\lambda_i = z_{2i-1} \sqcup \mathrm{cg}_\delta(p_i, q_i) \sqcup z_{2i} \sqcup \mathrm{g}_\delta(q'_i)$ (see Figure 3.7a). Let $s = (\alpha, 0)$ and $t = \big((2n+1)\alpha, 2\tau\big)$. Then the curve σ is defined as

$$\sigma = \mathrm{g}_\delta(s) \sqcup \lambda_1 \sqcup \lambda_2 \sqcup \ldots \sqcup \lambda_n \sqcup \mathrm{g}_\delta(t)\,.$$

First, suppose there is a curve $\pi' = \langle \pi'_1, \ldots, \pi'_{4n+1} \rangle \Subset \mathcal{U}$ such that $d_{\mathrm{F}}(\pi', \sigma) \leq \delta$. Let $\pi = \langle \pi_1, \ldots, \pi_{2n+1} \rangle$ be the curve such that $\pi_{2n+1} = \pi'_{4n+1}$, and for all $i \in [n]$, $\pi_{2i-1} = \pi_{4i-3}$ and $\pi_{2i} = \pi_{4i}$. We argue that π is an α-regular Y-respecting curve with $\pi_1 = s$ and $\pi_{2n+1} = t$.

Observe that π is α-regular, as by the definition of $\mathcal{U}$, π_i is a point on the vertical segment V_i. Also, as σ begins (resp. ends) with $\mathrm{g}_\delta(s)$ (resp. $\mathrm{g}_\delta(t)$), by Lemma 3.31, $\pi_1 = \pi'_1 = s$ (resp. $\pi_{2n+1} = \pi'_{4n+1} = t$). Thus, it remains to argue that π is Y-respecting. To that end, consider the portion λ_i of σ for some i.

First, consider the gadget $\mathrm{g}_\delta(q'_i)$ from λ_i lying between z_{2i} and z_{2i+1}. By our choice of α, this gadget is strictly more than δ away from both V_{2i} and V_{2i+1}, and so the portion of π' aligned with $\mathrm{g}_\delta(q'_i)$ must lie between $\pi'_{4i} = \pi_{2i}$ and $\pi'_{4i+1} = \pi_{2i+1}$. Thus, by Lemma 3.31, π must pass through q'_i.

Now consider the gadget $\mathrm{cg}_\delta(p_i, q_i) = \mathrm{lcg}(p_i) \sqcup \mathrm{ucg}(q_i)$ from λ_i lying between z_{2i-1} and z_{2i}. This gadget is strictly more than δ away from both V_{2i-1} and V_{2i}, implying both that the portion of π' aligned with $\mathrm{cg}_\delta(p_i, q_i)$ lies between π'_{4i-3} and π'_{4i}, and that all three segments in the subcurve from π'_{4i-3} to π'_{4i} must in part map to $\mathrm{cg}_\delta(p_i, q_i)$. Thus, by Lemma 3.33, $\pi'_{4i-3}\pi'_{4i-2}$ passes through p_i, and $\pi'_{4i-1}\pi'_{4i}$ passes through

q_i. By Corollary 3.34, either $\pi'_{4i-2} = \pi'_{4i}$ or $\pi'_{4i-3} = \pi'_{4i-1}$, and thus $\pi'_{4i-3}\pi'_{4i} = \pi_{2i-1}\pi_{2i}$ passes through either p_i or q_i (see Figure 3.7b). Thus, π is Y-respecting.

Now suppose that there is an α-regular Y-respecting curve $\pi = \langle \pi_1, \ldots, \pi_{2n+1} \rangle$ such that $\pi_1 = s$ and $\pi_{2n+1} = t$. Let $\mathrm{int}(p_i)$ be the intersection with V_{2i} of the line through π_{2i-1} and p_i, and let $\mathrm{int}(q_i)$ be the intersection with V_{2i-1} of the line through π_{2i} and q_i. Let $\pi' = \langle \pi'_1, \ldots, \pi'_{4n+1} \rangle$ be the curve such that $\pi'_{4n+1} = \pi_{2n+1}$, and for all $i \in [n]$, $\pi'_{4i-3} = \pi_{2i-1}$, $\pi'_{4i-2} = \mathrm{int}(p_i)$, $\pi'_{4i-1} = \rho$, and $\pi'_{4i} = \pi_{2i}$, where $\rho = \pi_{2i-1}$ if π passes through q_i and $\rho = \mathrm{int}(q_i)$ if π passes through p_i. (See Figure 3.7b.)

Let $\mathrm{mid}(S)$ be the midpoint of a line segment S. By construction, $\mathrm{mid}(\pi'_{4i-3}\pi'_{4i-2}) = p_i$, $\mathrm{mid}(\pi'_{4i-1}\pi'_{4i}) = q_i$, and $\mathrm{mid}(\pi'_{4i-2}\pi'_{4i-1}) = p_i$ (resp. q_i) if π passed through q_i (resp. p_i). Let $\gamma_i = \langle p_i, \pi'_{4i-2}, \pi'_{4i-1}, q_i \rangle$, which by the previous argument is a subcurve of π'.

To argue that $d_{\mathrm{F}}(\pi', \sigma) \leq \delta$, we now describe how to walk along the curves π' and σ so that at all times the distance between the positions on the respective curves is at most δ. Note that γ_i satisfies the conditions of Lemma 3.35, implying that $d_{\mathrm{F}}(\mathrm{cg}_\delta(p_i, q_i), \gamma_i) \leq \delta$, and thus for all i, we can map $\mathrm{cg}_\delta(p_i, q_i)$ to γ_i. For the other parts of the curves, first observe that with the exception of the $\mathrm{cg}_\delta(p_i, q_i)$ gadgets, σ is x-monotone, i.e. as we walk along it, the x-coordinate never decreases. Moreover, with the exception of the γ_i portions, π' is x-monotone. Finally, observe that $\mathrm{cg}_\delta(p_i, q_i)$ and γ_i have the same starting and ending points, and π' and σ both start at s and end at t. Thus, with the exception of the $\mathrm{cg}_\delta(p_i, q_i)$ and γ_i portions, we can map all points from σ with a given x-coordinate to the point on π' with the same x-coordinate. It is easy to verify that this maps points between the curves that are at most δ apart. First, as π' is identical to π outside of the γ_i, and since π is Y-respecting, π' passes through s, t, and q'_i for all i. Thus, the matching stands still on π' at these respective points as σ executes the $\mathrm{g}_\delta(s)$, $\mathrm{g}_\delta(t)$, and $\mathrm{g}_\delta(q'_i)$ gadgets. The vertical distance elsewhere between the curves is at most $4M$ by Corollary 3.26, and by construction $4M \leq \delta$. □

3.2 Algorithms for Lower Bound Fréchet Distance

In the previous section, we have shown that the decision problem for $d_F^{\min}$ is hard, given an uncertain curve with line-segment-based imprecision model and a polygonal curve. Interestingly, the same problem is solvable in polynomial time for indecisive curves. This result highlights a distinction between $d_F^{\min}$ and $d_F^{\max}$ and between the different uncertainty models. To tackle $d_F^{\min}$ with general uncertain curves, we develop approximation algorithms.

3.2.1 Exact Solution for Indecisive Curves

The key idea is that we can use a dynamic programming approach similar to that for computing the Fréchet distance [23] and only keep track of realisations of the last indecisive point considered so far. (Note that one can also reduce the problem to the Fréchet distance between paths in *DAG complexes,* studied by Har-Peled and Raichel [137], but this yields a slower running time.) We present the approach for an indecisive and a precise curve, and then generalise it to two indecisive curves.

Indecisive and Precise

Consider the setting with an indecisive curve $\mathcal{U} = \langle U_1, \dots, U_m \rangle$ with m points and a precise curve $\sigma = \langle q_1, \dots, q_n \rangle$ with n points; each indecisive point has k possible realisations, $U_i = \{p_i^1, \dots, p_i^k\}$. We want to solve the decision problem *'Is the lower bound Fréchet distance between the curves at most some threshold δ?'*, so $d_F^{\min}(\mathcal{U}, \sigma) \leq \delta$?

In the free-space diagram for this problem, let $\mathcal{U}$ be positioned along the horizontal axis, and σ along the vertical axis. Just as for the precise curve Fréchet distance, we are interested in the reachable intervals on the cell boundary, since the free space in the cell interior is convex; however, now we care about the different realisations of the points, so we get a set of reachable boundaries instead of a single cell boundary. We can adapt the standard dynamic program to deal with this problem. We propagate reachability column by column. An important aspect is that we only need to make sure that a reachable point is reachable by a monotone path in the free-space diagram induced by *some* valid realisation; we do not need to remember which one, since we never return to the previous points on the indecisive curve, and we also do not care about the realisations that yield a distance higher than δ—a significant deviation from the upper bound Fréchet distance.

First of all, define $\mathrm{Feas}(i, \ell)$ to be the *feasibility column* for the realisation p_i^ℓ of U_i. This is a set of intervals on the vertical line containing a cell boundary in the free-space diagram, corresponding to the subintervals of one curve within distance δ from a point on the other curve. It is computed exactly the same way as for the precise Fréchet distance—it depends on the distance between a point and a line segment and gives a single interval on each vertical cell boundary. We can compute feasibility for the right boundary of all cells in a column for a given realisation, thus obtaining $\mathrm{Feas}(i, \ell)$.

Recall the standard dynamic program for computing the Fréchet distance on precise curves. Represent it so that it operates column by column, grouping propagation of reachable intervals between vertically aligned cells. Call that procedure $\mathrm{Prop}(R)$, where R is the *reachability column* for point i and the result is the reachability column for point $i+1$ on one of the curves. Again, the reachability column is a set of intervals on a vertical line, indicating the points in the free-space diagram that are reachable from the lower left corner with a monotone path.

Define $\mathrm{Reach}(i, s)$ to be the reachability column induced by p_i^s, where a point is in a reachability interval if it can be reached by a monotone path for some realisation of the previous points. Then we compute

$$\mathrm{Reach}(i+1, \ell) = \mathrm{Feas}(i+1, \ell) \cap \bigcup_{\ell' \in [k]} \mathrm{Prop}(\mathrm{Reach}(i, \ell'))\,.$$

So, we iterate over all the realisations of the previous column, thus getting precise cells, and simply propagate the reachable intervals as in the precise Fréchet distance algorithm. For the column corresponding to U_1, we set one reachable interval of a single point at the bottom for all realisations p_1^s for which $\|p_1^s - q_1\| \leq \delta$.

We now show correctness of this approach.

Lemma 3.37. *For all $i > 1$,*

$$\mathrm{Reach}(i, \ell) = \left\{ y \;\middle|\; \exists_{p_1^{\ell_1}, \dots, p_{i-1}^{\ell_{i-1}}} \left[d_{\mathrm{F}}\Big(\big(\bigsqcup_{j \in [i-1]} p_j^{\ell_j} \big) \sqcup p_i^\ell, \sigma[1 : \lfloor y \rfloor] \sqcup \sigma(y) \Big) \leq \delta \right] \right\}.$$

So, a point is inside a reachability interval if and only if there is a realisation that defines a free-space diagram and a monotone path through that diagram to this point.

Proof. We show this by induction on i. To compute $\text{Reach}(2, \ell)$ for any fixed $\ell \in [k]$, we start from a single point in the bottom left corner of the free space for the realisations of U_1 that are close enough to q_1 and we propagate the reachability through the resulting precise free-space column. Clearly, the statement holds in this case; if some realisation of U_1 is too far from q_1, then the reachability column is correctly empty.

Now assume the statement holds for $\text{Reach}(i, \ell')$ for all $\ell' \in [k]$. Note that all the values that we add to $\text{Reach}(i + 1, \ell)$ for some fixed ℓ are feasible, since we explicitly take the feasibility column and intersect it with the propagated reachability. Any point y in $\text{Reach}(i + 1, \ell)$ comes as a result of propagation from some $\text{Reach}(i, \ell')$ for some ℓ'. So, there is at least one point y' in the reachability column i for realisation $p_i^{\ell'}$ from which there is a monotone path to y. Since we know there was a realisation up to that point of the two curves that enables a monotone path from the start of the free space diagram to y'; and since point U_{i+1} is independent from the previous points; and since we have a fixed valid realisation for points U_i and U_{i+1} that enables the continuation of the monotone path from y' to y, we conclude that the statement holds for the column $i + 1$. □

Therefore, querying the upper boundary of all reachability intervals for U_m will give us the answer to the decision problem.

Now we analyse the complexity of the reachability column. A particular right cell boundary is entirely reachable if the bottom of the cell is reachable; combined with the feasibility interval, we get one reachability interval per cell. Furthermore, if a cell is only reachable from the left, since we consider monotone paths, each realisation of the previous points induces a reachable interval of $[y', 1]$ for some $0 \leq y' \leq 1$ if you assume the boundary coordinate range to be $[0, 1]$; therefore, taking a union of such intervals still gives us at most one reachability interval per cell. So, in the worst case we store $\Theta(mk)$ intervals. To propagate, we consider all combinations of the two successive indecisive points for all cells, yielding the running time of $\Theta(mnk^2)$.

Furthermore, observe that we can also store a realisation of the previous point on the indecisive curve with the interval that corresponds to the lowest reachable point on the current interval. If we then store all the reachability columns, we can later backtrack and find a specific curve that realises the Fréchet distance below the threshold δ. This increases the storage requirements to $\Theta(mnk)$; the running time stays the same. We summarise the results:

Theorem 3.38. *Given an indecisive curve $\mathcal{U} = \langle U_1, \dots, U_m \rangle$, where each indecisive point has k options, $U_i = \{p_i^1, \dots, p_i^k\}$, a precise curve $\sigma = \langle q_1, \dots, q_n \rangle$, and a threshold $\delta > 0$, we can decide if $d_{\mathrm{F}}^{\min}(\mathcal{U}, \sigma) \leq \delta$ in time $\Theta(mnk^2)$ in the worst case, using $\Theta(mk)$ space. We can also report the realisation of $\mathcal{U}$ realising the Fréchet distance at most δ, using $\Theta(mnk)$ space instead. Call the algorithm that solves the problem and reports a fitting realisation* Decider$(\delta, \mathcal{U}, \sigma)$.

Indecisive and Indecisive

Now consider the setting where instead of σ we are given curve $\mathcal{V} = \langle V_1, \dots, V_n \rangle$ with k options per indecisive point, $V_i = \{q_i^1, \dots, q_i^k\}$. We can adapt the algorithm of the previous section by propagating in column-major order, but cell by cell.

A cell boundary now depends on three indecisive points, since it corresponds to a segment on one curve and a point on the other curve, so there are k^3 options per boundary to consider. We now store the possibilities for $m-1$ right cell boundaries, k^3 realisations per boundary, and a single horizontal boundary, with also k^3 options. So, we use $\Theta(mk^3)$ storage.

Whenever we propagate to one further cell, we need to find the reachability for the top and the right boundary of the cell based on the left and the lower boundary of the cell. We again go over all the combinations of the realisations of the points that define the cell, yielding k^4 possible precise cells to consider. We aggregate the values as before, as for both the top and the right boundary only three points matter.

Since we solve the same problem as in the previous section and never have to revisit a previously considered point, it should be clear that this approach is correct. However, now we take $\Theta(k^4)$ time per cell, so in the worst case we need $\Theta(mnk^4)$ time to complete the propagation.

Theorem 3.39. *Given two indecisive curves $\mathcal{U} = \langle U_1, \dots, U_m \rangle$ and $\mathcal{V} = \langle V_1, \dots, V_n \rangle$, where each indecisive point has k options, $U_i = \{p_i^1, \dots, p_i^k\}$ and*

$V_i = \{q_i^1, \dots, q_i^k\}$, and a threshold $\delta > 0$, we can decide if $d_{\mathrm{F}}^{\min}(\mathcal{U}, \mathcal{V}) \le \delta$ in time $\Theta(mnk^4)$ in the worst case, using $\Theta(mk^3)$ space.

3.2.2 Approximation by Grids

Given a general uncertain curve $\mathcal{U}$ and a polygonal curve σ, in this section, we show how to find a curve $\pi \Subset \mathcal{U}$ such that $d_{\mathrm{F}}(\pi, \sigma) \le (1+\varepsilon) d_{\mathrm{F}}^{\min}(\mathcal{U}, \sigma)$. This is accomplished by carefully discretising the regions, in effect approximately reducing the problem to the indecisive case, for which we then can use Theorem 3.38.

For simplicity, assume the uncertain regions have constant complexity. Throughout the section, we assume $d_{\mathrm{F}}^{\min}(\mathcal{U}, \sigma) > 0$, justified by the following lemma.

Lemma 3.40. *Let $\mathcal{U}$ be an uncertain curve with m vertices, and σ a polygonal curve with n vertices. Then one can determine whether $d_{\mathrm{F}}^{\min}(\mathcal{U}, \sigma) = 0$ in $O(mn)$ time.*

Proof. Observe that if for some j, σ_j lies on the segment $\sigma_{j-1}\sigma_{j+1}$, then $d_{\mathrm{F}}(\sigma, \sigma') = 0$, where $\sigma' = \langle \sigma_1, \dots, \sigma_{j-1}, \sigma_{j+1}, \dots, \sigma_n \rangle$. So we can assume that no vertex of σ lies on the segment between its neighbours, as otherwise we can remove that vertex and get the same result in terms of the Fréchet distance. Thus, at every vertex σ turns, implying that if there exists $\pi \Subset \mathcal{U}$ such that $d_{\mathrm{F}}(\pi, \sigma) = 0$, then for all j, σ_j must be aligned with some π_i.

This observation leads to a simple decision procedure. Define

$$s(j) = \{ i \in [m] \mid d_{\mathrm{F}}(\pi[1:i], \sigma[1:j]) = 0 \},$$

so a set of indices on σ that yield the zero Fréchet distance between the corresponding prefix curves. Then we can go through σ one vertex at a time, maintaining $s(j)$, and ultimately $d_{\mathrm{F}}^{\min}(\mathcal{U}, \sigma) = 0$ if and only if $m \in s(n)$.

Initially, $s(1) = \{ i \in [m] \mid \forall k \in [i] : \sigma_1 \in U_k \}$, which is easy to test and compute. For $j > 1$, $s(j)$ can be computed from $s(j-1)$ as follows. Let $\mathrm{Stab}_j(k)$ be the set of indices $i > k$ such that there exist points $p_{k+1}, \dots, p_{i-1}$, appearing in order along $\sigma_{j-1}\sigma_j$, where $p_\ell \in U_\ell$ for all $k < \ell < i$. (Note that we always have $k+1 \in \mathrm{Stab}_j(k)$.) So, $\mathrm{Stab}_j(k)$ is the set of indices i of uncertainty regions, starting from $k+1$, such that all the regions between k and i are stabbed by the segment $\sigma_{j-1}\sigma_j$ in the correct order. Then we have

$$s(j) = \{i \mid \sigma_j \in U_i \wedge i \in \mathrm{Stab}_j(k) \text{ with } k = \max_{\ell < i}\{\ell \in s(j-1)\}\} .$$

From this definition of $s(j)$ it is easy to see that it can be computed in $O(m)$ time given $s(j-1)$, and thus the total time required is $O(mn)$. In particular, if $s(j-1)$ is non-empty, then let z be the minimum value in $s(j-1)$. We now incrementally loop over values of i, where initially $i = z+1$, and add i to $s(j)$ if $\sigma_j \in U_i$ and $i \in \mathrm{Stab}_j(z)$. Note that in constant time per iteration we can maintain sufficient information to determine if $i \in \mathrm{Stab}_j(z)$, as we describe further. If at any iteration $i = z' + 1$ for $z' \in s(j-1)$, we forget $\mathrm{Stab}_j(z)$ (as we no longer need to stab those regions) and start maintaining and checking $\mathrm{Stab}_j(z')$.

Note that the intersection of any U_ℓ with $\sigma_{j-1}\sigma_j$ is a constant number of intervals along $\sigma_{j-1}\sigma_j$. Then $\mathrm{Stab}_j(k)$ can be computed incrementally as follows. First, let p_{k+1} be the earliest point of $\sigma_{j-1}\sigma_j \cap U_{k+1}$. For some $i > k+1$, let p_i be the earliest point of $\sigma_{j-1}\sigma_j \cap U_i$, which is at least as far along $\sigma_{j-1}\sigma_j$ as p_{i-1} (if it exists). If such p_i exists, then we know that $i \in \mathrm{Stab}_j(k)$. Maintaining this information indeed takes constant time per iteration. ☐

Decision Procedure

An algorithm is a $(1+\varepsilon)$-*decider* for Problem 3.22, if when $d_{\mathrm{F}}^{\min}(\mathcal{U}, \sigma) \le \delta$, it returns a curve $\pi \Subset \mathcal{U}$ such that $d_{\mathrm{F}}(\pi, \sigma) \le (1+\varepsilon)\delta$, and when $d_{\mathrm{F}}^{\min}(\mathcal{U}, \sigma) > (1+\varepsilon)\delta$, it returns `False` (in between either answer is allowed). In this section, we present a $(1+\varepsilon)$-*decider* for Problem 3.22. We make use of the following standard observation.

Observation 3.41. *Given a curve $\pi = \langle \pi_1, \dots, \pi_n \rangle$, call a curve $\sigma = \langle \sigma_1, \dots, \sigma_n \rangle$ an r-perturbation of π if $\|\pi_i - \sigma_i\| \le r$ for all $i \in [n]$. Since $\|\pi_i - \sigma_i\|, \|\pi_{i+1} - \sigma_{i+1}\| \le r$, all points of the segment $\sigma_i\sigma_{i+1}$ are within distance r of $\pi_i\pi_{i+1}$. For segments this implies that $d_{\mathrm{F}}(\pi_i\pi_{i+1}, \sigma_i\sigma_{i+1}) \le r$, which implies that $d_{\mathrm{F}}(\pi, \sigma) \le r$ by composing the matchings for all i.*

The high-level idea is to replace $\mathcal{U}$ with the set of grid points it intersects; however, as our uncertainty regions may avoid the grid points, we need to include a slightly larger set of points.

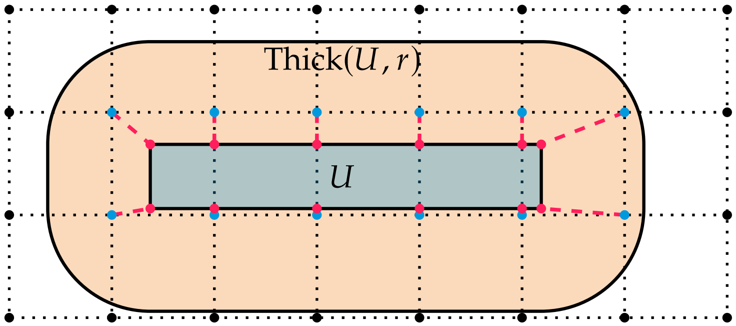


Figure 3.8. An example of the sets from Definition 3.42. The region U is shown in blue, and Thick(U, r) is in orange. The grid points of $GT_r(U)$ are in blue and the corresponding set of expanded r-grid points $\mathrm{EG}_r(U)$ are in red.

Definition 3.42. Let U be a compact subset of $\mathbb{R}^d$. We now define the set of points $\mathrm{EG}_r(U)$ which we call the *expanded r-grid points* of U (see Figure 3.8). Let $B(\sqrt{d}r)$ denote the ball of radius $\sqrt{d}r$, centred at the origin. Let $\mathrm{Thick}(U, r) = U \oplus B(\sqrt{d}r)$, where $\oplus$ denotes the Minkowski sum. Let G_r denote the regular grid of side length r, and let $GT_r(U)$ denote the subset of grid vertices from G_r that fall in Thick(U, r). Finally, we define

$$\mathrm{EG}_r(U) = \{p \mid p = \arg\min_{q \in U} \|q - x\| \text{ for } x \in GT_r(U)\} .$$

In the following observation we use the terms defined above.

Observation 3.43. *For any* $x \in U$*, there is a point* $p \in \mathrm{EG}_r(U)$ *such that* $\|p - x\| \leq 2\sqrt{d}r$.

Proof. For any point $x \in U$, let g be its nearest grid point in G_r. Since $\|x - g\| \leq \sqrt{d}r$, we know that $g \in \mathrm{Thick}(U, r) = U \oplus B(\sqrt{d}r)$. So let p be the point in U which is closest to g; thus, $p \in \mathrm{EG}_r(U)$. Therefore, $\|x - p\| \leq \|x - g\| + \|g - p\| \leq \sqrt{d}r + \sqrt{d}r = 2\sqrt{d}r$. □

Lemma 3.44. *There is a* $(1 + \varepsilon)$*-decider for Problem 3.22 in d dimensions with running time* $O(mn \cdot (1 + (\Delta/\varepsilon\delta)^{2d}))$*, for* $0 < \varepsilon \leq 1$ *and constant* d*, where* $\Delta = \max_{i \in [m]} \mathrm{diam}(U_i)$ *is the maximum diameter of an uncertain region.*

Proof. It helps with the analysis if $\varepsilon\delta < \Delta$. To ensure this, we first do the following. Select an arbitrary curve $x \Subset \mathcal{U}$. Now using the standard $O(mn)$-time exact decider for the Fréchet distance [23], query whether $d_{\mathrm{F}}(x, \sigma) \leq (1+\varepsilon)\delta$. If the decider returns $d_{\mathrm{F}}(x, \sigma) \leq (1+\varepsilon)\delta$, then we can return x as our solution. Otherwise, $d_{\mathrm{F}}(x, \sigma) > (1+\varepsilon)\delta$, and we next query whether $d_{\mathrm{F}}(x, \sigma) \leq \Delta + \delta$. By Observation 3.41 and the triangle inequality, $d_{\mathrm{F}}(x, \sigma) \leq \Delta + d_{\mathrm{F}}^{\min}(\mathcal{U}, \sigma)$. Thus, if the decider returns $\Delta + \delta < d_{\mathrm{F}}(x, \sigma)$, then $\delta < d_{\mathrm{F}}^{\min}(\mathcal{U}, \sigma)$, and so we return `False`. Otherwise, the two decider calls tell us that $(1+\varepsilon)\delta < d_{\mathrm{F}}(x, \sigma) \leq \Delta + \delta$, implying $\varepsilon\delta < \Delta$.

Let $r = {}^{\varepsilon\delta}/_{2\sqrt{d}}$, and for any U_i of $\mathcal{U}$, let $E_i = \mathrm{EG}_r(U_i)$ denote the expanded r-grid points of U_i, as defined in Definition 3.42. Consider the indecisive curve $\mathcal{U}' = \langle E_1, \ldots, E_m \rangle$. We call the algorithm $\textsc{Decider}((1+\varepsilon)\delta, \mathcal{U}', \sigma)$ of Theorem 3.38 and return whatever it returns, i.e. if it returns a curve, then we return that curve, and if it returns that $d_{\mathrm{F}}^{\min}(\mathcal{U}', \sigma) > (1+\varepsilon)\delta$, then we return that $d_{\mathrm{F}}^{\min}(\mathcal{U}, \sigma) > (1+\varepsilon)\delta$.

First, observe that $E_i \subseteq U_i$, and thus $d_{\mathrm{F}}^{\min}(\mathcal{U}, \sigma) \leq d_{\mathrm{F}}^{\min}(\mathcal{U}', \sigma)$. So if $d_{\mathrm{F}}^{\min}(\mathcal{U}, \sigma) > (1+\varepsilon)\delta$, then the decider must return $d_{\mathrm{F}}^{\min}(\mathcal{U}', \sigma) > (1+\varepsilon)\delta$, as desired. Now suppose that $d_{\mathrm{F}}^{\min}(\mathcal{U}, \sigma) \leq \delta$. In this case, we argue that our algorithm outputs a curve $\pi' \Subset \mathcal{U}$ such that $d_{\mathrm{F}}(\pi', \sigma) \leq (1+\varepsilon)\delta$. It suffices to argue that there exists some curve $\pi' \Subset \mathcal{U}'$ such that $d_{\mathrm{F}}(\pi', \sigma) \leq (1+\varepsilon)\delta$, as then Theorem 3.38 guarantees the decider outputs a curve (which is in $\mathrm{Real}(\mathcal{U})$, as it is a superset of $\mathrm{Real}(\mathcal{U}')$). So let $\pi = \langle \pi_1, \ldots, \pi_m \rangle$ be the curve in $\mathrm{Real}(\mathcal{U})$ realising the lower bound Fréchet distance to σ, that is, $d_{\mathrm{F}}(\pi, \sigma) = d_{\mathrm{F}}^{\min}(\mathcal{U}, \sigma)$. Let $\pi' = \langle \pi'_1, \ldots, \pi'_m \rangle$ be the curve such that $\pi'_i = \min_{x \in E_i} \|x - \pi_i\|$. Note that by Observation 3.43, we have $\|\pi_i - \pi'_i\| \leq 2\sqrt{d}r$ for all i. Thus, π' is a $2\sqrt{d}r$-perturbation of π as described in Observation 3.41, and so $d_{\mathrm{F}}(\pi, \pi') \leq 2\sqrt{d}r = \varepsilon\delta$. As the Fréchet distance satisfies the triangle inequality, we therefore have $d_{\mathrm{F}}(\pi', \sigma) \leq d_{\mathrm{F}}(\pi, \sigma) + d_{\mathrm{F}}(\pi, \pi') \leq \delta + \varepsilon\delta = (1+\varepsilon)\delta$. Thus, as $\pi' \Subset \mathcal{U}'$, when our algorithm calls $\textsc{Decider}((1+\varepsilon)\delta, \mathcal{U}', \sigma)$, it returns a curve.

For the running time, recall we first spent $O(mn)$ time to ensure $\varepsilon\delta < \Delta$, in which case we must bound the number of points in each E_i. By Definition 3.42, for all i, the number of points in E_i is bounded by the number of grid points in the region $\mathrm{Thick}(U_i, r)$. This region is the Minkowski sum of a compact set of diameter at most Δ with a radius $\sqrt{d}r$ ball, so its diameter is at most $\Delta + 2\sqrt{d}r$. Recall that d is a constant;

thus, the number of grid points and hence $|E_i|$ is

$$O\left(\left(\frac{\Delta + 2\sqrt{d}r}{r}\right)^d\right) = O\left(\left(\frac{2\sqrt{d}\Delta}{\varepsilon\delta} + 2\sqrt{d}\right)^d\right) = O\left(\left(\frac{\Delta}{\varepsilon\delta} + 1\right)^d\right) = O\left(\left(\frac{\Delta}{\varepsilon\delta}\right)^d\right).$$

Thus, by Theorem 3.38, the call to Decider takes time $O\big(mn(\Delta/\varepsilon\delta)^{2d}\big)$, which bounds the total time of our algorithm. □

Optimisation

Theorem 3.45. *Let $\mathcal{U}$ be an uncertain curve with m vertices, σ a polygonal curve with n vertices, and $\delta = d_F^{\min}(\mathcal{U}, \sigma)$. Then for any $0 < \varepsilon \leq 1$, there is an algorithm which returns a curve $\pi \Subset \mathcal{U}$ such that $d_F(\pi, \sigma) \leq (1+\varepsilon)\delta$, whose running time is $O\big(mn(\log(mn) + (\Delta/\varepsilon\delta)^{2d})\big)$ for constant d, where $\Delta = \max_{i\in[m]} \operatorname{diam}(U_i)$ is the maximum diameter of an uncertain region.*

Proof. Fix an arbitrary curve $x \Subset \mathcal{U}$. First, we compute the Fréchet distance between x and σ. If $d_F(x, \sigma) \geq \Delta + \Delta/\varepsilon$, then we return x as our solution. Intuitively, this means that the Fréchet distance is large when compared to the diameter of the uncertain regions, and so any realisation we can pick works as a $(1+\varepsilon)$-approximation. To see why this is valid, let $\hat{\pi} \Subset \mathcal{U}$ be an optimal solution, that is, $d_F(\hat{\pi}, \sigma) = d_F^{\min}(\mathcal{U}, \sigma)$. Note that x is a Δ-perturbation of $\hat{\pi}$, and thus by the triangle inequality and Observation 3.41,

$$d_F(x, \sigma) \leq d_F(x, \hat{\pi}) + d_F(\hat{\pi}, \sigma) \leq \Delta + d_F(\hat{\pi}, \sigma).$$

If $\Delta + \Delta/\varepsilon \leq d_F(x, \sigma)$, then plugging in the inequality above implies that $\Delta \leq \varepsilon \cdot d_F(\hat{\pi}, \sigma)$, which in turn implies that

$$d_F(x, \sigma) \leq \Delta + d_F(\hat{\pi}, \sigma) \leq (1+\varepsilon) \cdot d_F(\hat{\pi}, \sigma).$$

So suppose that $d_F(x, \sigma) < (1 + 1/\varepsilon)\Delta$, in which case

$$\begin{aligned} d_F^{\min}(\mathcal{U}, \sigma) &= d_F(\hat{\pi}, \sigma) \leq d_F(x, \sigma) + d_F(\hat{\pi}, x) \\ &< \left(1 + \frac{1}{\varepsilon}\right)\Delta + \Delta = \left(2 + \frac{1}{\varepsilon}\right)\Delta = \gamma. \end{aligned}$$

Let GridDecider$(\mathcal{U}, \sigma, \varepsilon', \delta)$ be the $(1+\varepsilon')$-decider of Lemma 3.44, which correctly returns either `False` (which implies $d_F^{\min}(\mathcal{U}, \sigma) > \delta$) or a curve in $\operatorname{Real}(\mathcal{U})$ with the Fréchet distance at most $(1+\varepsilon')\delta$ to

σ. We perform a decreasing exponential search using GridDecider. Specifically, starting at $i = 0$, we call GridDecider($\mathcal{U}, \sigma, {}^{\varepsilon}/_{4}, \gamma/(1 + {}^{\varepsilon}/_{4})^i$). If GridDecider returns a curve (i.e. `True`), we increment i by 1 and repeat, otherwise if GridDecider outputs `False`, we return the curve from iteration $i - 1$. (Note that GridDecider cannot return `False` when $i = 0$, as this would imply that $d_{\mathrm{F}}^{\min}(\mathcal{U}, \sigma) > \gamma$.)

Let j denote the index when the algorithm stops. So we know that GridDecider($\mathcal{U}, \sigma, {}^{\varepsilon}/_{4}, \gamma/(1 + {}^{\varepsilon}/_{4})^j$) returned `False`, and we also know that GridDecider($\mathcal{U}, \sigma, {}^{\varepsilon}/_{4}, \gamma/(1 + {}^{\varepsilon}/_{4})^{j-1}$) returned a curve $\pi \Subset \mathcal{U}$ such that $d_{\mathrm{F}}(\pi, \sigma) \leq (1 + {}^{\varepsilon}/_{4}) \cdot \gamma/(1 + {}^{\varepsilon}/_{4})^{j-1}$. Therefore,

$$\frac{\gamma}{(1 + {}^{\varepsilon}/_{4})^j} < d_{\mathrm{F}}^{\min}(\mathcal{U}, \sigma) \leq d_{\mathrm{F}}(\pi, \sigma) \leq (1 + {}^{\varepsilon}/_{4}) \frac{\gamma}{(1 + {}^{\varepsilon}/_{4})^{j-1}} = \frac{\gamma}{(1 + {}^{\varepsilon}/_{4})^{j-2}}\,,$$

which implies that

$$\begin{aligned} d_{\mathrm{F}}(\pi, \sigma) &\leq \left(1 + \frac{\varepsilon}{4}\right)^2 d_{\mathrm{F}}^{\min}(\mathcal{U}, \sigma) = \left(1 + \frac{\varepsilon}{2} + \frac{\varepsilon^2}{16}\right) \cdot d_{\mathrm{F}}^{\min}(\mathcal{U}, \sigma) \\ &< (1 + \varepsilon) \cdot d_{\mathrm{F}}^{\min}(\mathcal{U}, \sigma)\,. \end{aligned}$$

As for the running time, by Lemma 3.44, the time for the i-th call to GridDecider is

$$O\left(mn\left(\frac{(1 + {}^{\varepsilon}/_{4})^i \Delta}{\varepsilon\gamma}\right)^{2d}\right) = O\left(mn\left(\frac{(1 + {}^{\varepsilon}/_{4})^i \Delta}{\varepsilon(2 + {}^{1}/_{\varepsilon})\Delta}\right)^{2d}\right) = O\left(mn\left(1 + \frac{\varepsilon}{4}\right)^{2di}\right).$$

Recall that $\delta = d_{\mathrm{F}}^{\min}(\mathcal{U}, \sigma)$ and j is the index the last time GridDecider is called. By the argument above, $\delta \leq \gamma/(1 + {}^{\varepsilon}/_{4})^{j-2}$, which implies that $j - 2 \leq \log_{1+\varepsilon/4}({}^{\gamma}/_{\delta})$. Recall that d is a constant; as GridDecider is called $j+1$ times, and the running times for the calls to GridDecider form an increasing geometric series, the total time for all calls to GridDecider is

$$\begin{aligned} & O\left(mn\left(1 + \frac{\varepsilon}{4}\right)^{2d \cdot \left(3 + \log_{1+\varepsilon/4}({}^{\gamma}/_{\delta})\right)}\right) \\ &= O\left(mn\left(1 + \frac{\varepsilon}{4}\right)^{6d}\left(1 + \frac{\varepsilon}{4}\right)^{2d \cdot \log_{1+\varepsilon/4}({}^{\gamma}/_{\delta})}\right) \\ &= O\left(mn\left(1 + \frac{\varepsilon}{4}\right)^{2d \cdot \log_{1+\varepsilon/4}({}^{\gamma}/_{\delta})}\right) = O\left(mn\left(\frac{\gamma}{\delta}\right)^{2d \cdot \log_{1+\varepsilon/4}(1 + {}^{\varepsilon}/_{4})}\right) \\ &= O\left(mn\left(\frac{\gamma}{\delta}\right)^{2d}\right) = O\left(mn\left(\frac{(2 + {}^{1}/_{\varepsilon})\Delta}{\delta}\right)^{2d}\right) = O\left(mn\left(\frac{\Delta}{\varepsilon\delta}\right)^{2d}\right). \end{aligned}$$

As it takes $O(mn\log(mn))$ time to initially compute $d_\mathrm{F}(x,\sigma)$ using the algorithm of Alt and Godau [23], the total running time is $O\big(mn(\log(mn)+ ({}^{\Delta}/_{\varepsilon\delta})^{2d})\big)$. □

If the polygonal curve σ is replaced with an uncertain curve $\mathcal{V}$, it is easy to see that by discretising both $\mathcal{U}$ and $\mathcal{V}$, the same analysis gives an algorithm to compute $d_\mathrm{F}^{\min}(\mathcal{U},\mathcal{V})$. The only difference now is that we must use Theorem 3.39 instead of Theorem 3.38, yielding the following.

Corollary 3.46. *Let $\mathcal{U}$ and $\mathcal{V}$ be uncertain curves with m and n vertices, respectively, and $\delta = d_\mathrm{F}^{\min}(\mathcal{U},\mathcal{V})$. Then for any $0 < \varepsilon \leq 1$, there is an algorithm returning curves $\pi \Subset \mathcal{U}$ and $\sigma \Subset \mathcal{V}$ such that $d_\mathrm{F}(\pi,\sigma) \leq (1+\varepsilon)\delta$, whose running time is $O\big(mn(\log(mn) + ({}^{\Delta}/_{\varepsilon\delta})^{4d})\big)$ for constant d, where Δ is the maximum diameter of an uncertain region.*

3.2.3 Greedy Algorithm

Here we argue that there is a simple 3-decider as defined in Section 3.2.2 for Problem 3.22, running in near-linear time in the plane for separated regions. Roughly speaking, the idea is to greedily and iteratively pick $\pi_i \in U_i$ so as to allow us to get as far as possible along σ. Without any assumptions on $\mathcal{U}$, this greedy procedure may walk too far ahead and get stuck. Thus, in this section, we assume that consecutive U_i are separated, so as to ensure optimal solutions do not lag too far behind. Here we also assume that U_i are convex, i.e. imprecise, and have constant complexity, as it simplifies certain definitions. Throughout this section, let $\mathcal{U} = \langle U_1,\dots,U_m\rangle$ be an uncertain curve and let $\sigma = \langle\sigma_1,\dots,\sigma_n\rangle$ be a polygonal curve.

Definition 3.47. Call $\mathcal{U}$ *γ-separated* if for all $i \in [m-1]$, $\|U_i - U_{i+1}\| > \gamma$ and each U_i is convex. Define an *r-visit* of U_i to be any maximal-length contiguous portion of $\sigma \cap \big(U_i \oplus B(2r)\big)$ which intersects $U_i \oplus B(r)$, where $\oplus$ denotes the Minkowski sum. If $\mathcal{U}$ is γ-separated for $\gamma \geq 4r$, then any r-visit of U_i is disjoint from any r-visit of U_j for $i \neq j$, in which case define the *true r-visit* of U_i to be the first r-visit of U_i which occurs after the true r-visit of U_{i-1}. (For U_1 it is the first r-visit.)

Lemma 3.48. *If $\mathcal{U}$ is γ-separated for $\gamma \geq 4r$, then for any curve $\pi \Subset \mathcal{U}$ and any reparametrisations f and g such that $\mathrm{cost}_{f,g}(\pi, \sigma) \leq r$, π_i must map to a point on the true r-visit of U_i for all i.*

Proof. First, note that since $\mathrm{cost}_{f,g}(\pi, \sigma) \leq r$, π_i must map to a point in an r-visit of U_i, and thus we only need to prove it is the true r-visit.

We prove the claim by induction on i. For $i = 1$, the claim holds, as π_1 must map to σ_1, and σ_1 is in the first r-visit of U_1, which is its true r-visit.

Now suppose the claim holds for $i-1$. π_i must map to a point on an r-visit of U_i, and by the induction hypothesis, this visit must happen after the true r-visit of U_{i-1} on σ. Moreover, as $\mathcal{U}$ is $4r$-separated, the first point in $U_i \oplus B(r)$ of the first r-visit of U_i that occurs after the true r-visit of U_{i-1} (i.e. true r-visit of U_i) must map to a point x on $\pi_{i-1}\pi_i$. Note, however, that as both x and π_i map to points in $U_i \oplus B(r)$, the portion of σ that the segment $x\pi_i$ maps to must lie within $U_i \oplus B(2r)$, i.e. the same r-visit. Therefore, all of $x\pi_i$ is mapped to the true r-visit of U_i, completing the proof. $\square$

For two points α and β on σ, let $\alpha \leq \beta$ denote that α occurs before β, and for any points $\alpha \leq \beta$ let $\sigma(\alpha, \beta)$ denote the subcurve between α and β.

Definition 3.49. The *δ-greedy sequence* of σ with respect to $\mathcal{U}$, denoted $\mathrm{gs}(\mathcal{U}, \sigma, \delta)$, is the longest possible sequence $\alpha = \langle \alpha_1, \dots, \alpha_k \rangle$ of points on σ, where $\alpha_1 = \sigma_1$, and for any $i > 1$, α_i is the point furthest along σ such that $\|\alpha_i - U_i\| \leq \delta$ and $d_{\mathrm{F}}(\alpha_{i-1}\alpha_i, \sigma(\alpha_{i-1}, \alpha_i)) \leq 2\delta$.

Observation 3.50. *For any $i \leq k$, let $\alpha^i = \langle \alpha_1, \dots, \alpha_i \rangle$ be the i-th prefix of $\mathrm{gs}(\mathcal{U}, \sigma, \delta)$. Then $d_{\mathrm{F}}(\alpha^i, \sigma(\alpha_1, \alpha_i)) \leq 2\delta$, and $\alpha^i \Subset \mathcal{U}^i \oplus B(\delta)$, where $\mathcal{U}^i \oplus B(\delta) = \langle U_1 \oplus B(\delta), \dots, U_i \oplus B(\delta) \rangle$.*

Lemma 3.51. *If $\mathcal{U}$ is 10δ-separated and $d_{\mathrm{F}}^{\min}(\mathcal{U}, \sigma) \leq \delta$, then $\mathrm{gs}(\mathcal{U}, \sigma, \delta)$ has length m and $\alpha_m = \sigma_n$.*

Proof. Let $\mathrm{gs}(\mathcal{U}, \sigma, \delta) = \alpha = \langle \alpha_1, \dots, \alpha_k \rangle$. Let $\mathrm{opt} = \langle \mathrm{opt}_1, \dots, \mathrm{opt}_m \rangle$ be any curve in $\mathrm{Real}(\mathcal{U})$ such that $d_{\mathrm{F}}(\mathrm{opt}, \sigma) = d_{\mathrm{F}}^{\min}(\mathcal{U}, \sigma)$. Throughout this proof, we fix a matching realising $d_{\mathrm{F}}(\mathrm{opt}, \sigma)$ and let β_i be the point on σ which opt_i maps to under this matching. For the curve α, we fix the matching which is the composition of the reparametrisations realising $d_{\mathrm{F}}(\alpha_{i-1}\alpha_i, \sigma(\alpha_{i-1}, \alpha_i)) \leq 2\delta$, and, in particular, α_i on α maps to α_i on σ.

We prove by induction that for $i \leq m$, α_i exists and $\beta_i \leq \alpha_i$. For $i = 1$, we have $\alpha_1 = \beta_1 = \sigma_1$. So assume that α_{i-1} exists. By Observation 3.50, $\alpha^{i-1} \Subset \mathcal{U}^{i-1} \oplus B(\delta)$, and, moreover, $d_{\mathrm{F}}(\sigma(\alpha_1, \alpha_{i-1}), \alpha^{i-1}) \leq 2\delta$. Since $\mathcal{U}$ is 10δ-separated, $\mathcal{U}^{i-1} \oplus B(\delta)$ is 8δ-separated, and thus by Lemma 3.48, α_{i-1} is on the true 2δ-visit of $U_{i-1} \oplus B(\delta)$ by the prefix curve $\sigma(\alpha_1, \alpha_{i-1})$. Observe that the true 2δ-visit of $U_{i-1} \oplus B(\delta)$ by the prefix curve $\sigma(\alpha_1, \alpha_{i-1})$ is a subset of the true 2δ-visit of $U_{i-1} \oplus B(\delta)$ by σ, and thus α_{i-1} is on the true 2δ-visit of $U_{i-1} \oplus B(\delta)$ by σ. We also have that $\mathrm{opt} \Subset \mathcal{U} \oplus B(\delta)$, as $U_j \subset U_j \oplus B(\delta)$ for all j, so by Lemma 3.48, β_{i-1} and β_i are on the true 2δ-visit of $U_{i-1} \oplus B(\delta)$ and $U_i \oplus B(\delta)$. In particular, this implies that $\beta_{i-1} \leq \alpha_{i-1} \leq \beta_i$, as the true 2δ-visits of $U_{i-1} \oplus B(\delta)$ and $U_i \oplus B(\delta)$ are disjoint. Thus, some point x on the segment $\mathrm{opt}_{i-1}\mathrm{opt}_i$ must map to α_{i-1}. Note that $d_{\mathrm{F}}(x\mathrm{opt}_i, \sigma(\alpha_{i-1}, \beta_i)) \leq \delta$. As $\|x - \alpha_{i-1}\| \leq \delta$, $d_{\mathrm{F}}(x\mathrm{opt}_i, \alpha_{i-1}\mathrm{opt}_i) \leq \delta$, and so by the triangle inequality for the Fréchet distance, $d_{\mathrm{F}}(\alpha_{i-1}\mathrm{opt}_i, \sigma(\alpha_{i-1}, \beta_i)) \leq 2\delta$. Since $\|\beta_i - \mathrm{opt}_i\| \leq \delta$, β_i is a possible choice for α_i, and thus α_i exists and $\beta_i \leq \alpha_i$. Finally, since α_i exists for all $i \leq m$, $\alpha = \mathrm{gs}(\mathcal{U}, \sigma, \delta)$ has length m, and moreover, since $\beta_m \leq \alpha_m$ and $\beta_m = \sigma_n$, we conclude that $\alpha_m = \sigma_n$. ☐

The following lemma is the only place where we require the points to be in $\mathbb{R}^2$. The proof uses a result from Guibas et al. [132].

Lemma 3.52. *For $\mathcal{U}$ and σ in $\mathbb{R}^2$, where $\mathcal{U}$ is 10δ-separated, $\mathrm{gs}(\mathcal{U}, \sigma, \delta)$ is computable in time $O(m + n \log n)$.*

Proof. Given α_i from $\mathrm{gs}(\mathcal{U}, \sigma, \delta)$, we describe how to compute α_{i+1}, if it exists. Let σ_j be the smallest-index vertex such that $\alpha_i < \sigma_j$. Let $\langle D_j, \ldots, D_n \rangle$ be the sequence of 2δ-radius disks, where D_l is centred at σ_l. Observe that for α_{i+1} to be able to lie on $\sigma_z\sigma_{z+1}$, for any $z \geq j$, we first require that $d_{\mathrm{F}}(\alpha_i\alpha_{i+1}, \sigma(\alpha_i, \alpha_{i+1})) \leq 2\delta$, which occurs if and only if there exist points $p_j, \ldots, p_z$ that appear in order along $\alpha_i\alpha_{i+1}$ such that $p_l \in D_l$. Clearly, such points are necessary, but they are also sufficient, as $d_{\mathrm{F}}(p_l p_{l+1}, \sigma_l\sigma_{l+1}) \leq 2\delta$. (As α_i and α_{i+1} lie on σ, the same holds for $\alpha_i\sigma_j$ and $\sigma_z\alpha_{i+1}$.) $\mathrm{gs}(\mathcal{U}, \sigma, \delta)$ also requires that α_{i+1} lie within distance δ of U_{i+1}. This is equivalent to requiring that $\sigma_z\sigma_{z+1}$ intersects $U_{i+1} \oplus B(\delta)$. As both $\sigma_z\sigma_{z+1}$ and $U_{i+1} \oplus B(\delta)$ are convex regions, their intersection is convex, i.e. a single subsegment of $\sigma_z\sigma_{z+1}$. Let $S_{i+1}(z)$

denote this segment, which we can compute in constant time, as U_{i+1} is a constant-complexity convex region. Note that α_{i+1} may lie on the same segment of σ as α_i, i.e. $z = j - 1$, which is an easier case, as no disks need to be intersected and $d_F(\alpha_i \alpha_{i+1}, \sigma(\alpha_i, \alpha_{i+1})) \leq 2\delta$ holds.

Given a sequence of k equal-radius disks $\langle D_1, \ldots, D_k \rangle$, say that a line ℓ stabs the disks if for all $j \leq k$, there exists a point $p_j \in \ell \cap D_j$ such that the p_j appear in order along ℓ. Guibas et al. [132] give an $O(k \log k)$-time algorithm that determines the set of all stabbing lines. As follows from the description of our problem, their algorithm can be used to determine α_{i+1} given α_i by restricting the stabbing line to first pass through α_i and requiring it to intersect $S_{i+1}(k)$ at the end.

We now sketch the necessary changes. Their algorithm inserts the disks in order, maintaining three objects—the support hull, the limiting lines, and the line stabbing wedge. The support hull consists of a pair of upper and lower concave chains that all stabbers must pass between, and the limiting lines represent the largest and the smallest slope stabbers. The wedge is the set of all points p such that there is a stabber that passes through p after passing through the required points from the disks. To modify their approach for our setting, we require the stabber to initially pass through α_i. This actually simplifies the problem by joining and collapsing the chains of the support hull,[3] and thus we can focus on the wedge. After j insertions, the wedge boundary consists of $O(j)$ pieces from the disks, flanked by the limiting lines. These ordered boundary pieces are stored in a binary tree to facilitate logarithmic-time updates when a new disk is inserted, and we can simply reuse this structure to determine the intersection of the wedge with $S_{i+1}(j)$.

By Definition 3.49, the line segment $\sigma_z \sigma_{z+1}$ that α_{i+1} lies on must have z be as large as possible. Thus, we run the incremental procedure above, where in the j-th round we check for intersection with $S_{i+1}(j)$. If no such intersection is found before we reach the end of σ or the wedge becomes empty, then α_{i+1} does not exist. Otherwise, α_{i+1} is defined. However, the rounds which have intersection with $S_{i+1}(j)$ need not be contiguous; thus, care is needed to determine the last such intersection efficiently.

---

[3] Alternatively, one can enforce the condition by defining an initial zero-radius disk D_0 at α_i, and indeed the referenced work [132] considers stabbers for more general collections of convex objects.

Let k be the largest index such that α_k is defined. By Observation 3.50, for any $i \leq k$, we have $d_{\mathrm{F}}(\alpha^i, \sigma(\alpha_1, \alpha_i)) \leq 2\delta$ and $\alpha^i \Subset \mathcal{U}^i \oplus B(\delta)$. Since $\mathcal{U}$ is 10δ-separated, $\mathcal{U}^i \oplus B(\delta)$ is 8δ-separated, and so by Lemma 3.48, α_i must be in the true 2δ-visit of $U_i \oplus B(\delta)$ by $\sigma(\alpha_1, \alpha_k)$. Thus, when computing α_i, we only need to consider vertices from σ which occur after α_{i-1} and before the end of the true 2δ-visit of $U_i \oplus B(\delta)$. If n_i is the number of such vertices, it therefore takes $O(1 + n_i \log n_i)$ time to compute α_i with the algorithm above. Moreover, as the true 2δ-visits for $U_i \oplus B(\delta)$ and $U_j \oplus B(\delta)$ for $i \neq j \leq k$ are disjoint, any vertex of σ contributes to at most two counts n_i, as we have $\alpha_j \in U_j \oplus B(\delta)$, and we may process vertices from α_j to the end of $U_j \oplus B(\delta)$ twice; so $\sum_i n_i \leq 2n$. Thus, the total running time is $O(m + n \log n) + \sum_{i=1}^{k} O(1 + n_i \log n_i) = O(m + n \log n)$, where the leading $O(m + n \log n)$ term accounts for the time to determine if α_{k+1} does not exist for $k < m$. □

Theorem 3.53. *Let $\mathcal{U}$ be $10r$-separated for some $r > 0$. There is a 3-decider for Problem 3.22 in the plane with the running time $O(m + n \log n)$ that works for any query value $0 < \delta \leq r$.*

Proof. Compute $\mathrm{gs}(\mathcal{U}, \sigma, \delta)$. If it has length m, then let $\pi = \langle \pi_1, \ldots, \pi_m \rangle$ be any curve in $\mathrm{Real}(\mathcal{U})$ such that $\|\pi_i - \alpha_i\| \leq \delta$ for all i. If this occurs and if $\alpha_m = \sigma_n$, we output π as our solution, and otherwise we output `False`. Thus, the running time follows from Lemma 3.52.

Observe that if we output a curve π, then $d_{\mathrm{F}}(\pi, \sigma) \leq 3\delta$, using the triangle inequality:

$$d_{\mathrm{F}}(\pi, \sigma) \leq d_{\mathrm{F}}(\pi, \alpha) + d_{\mathrm{F}}(\alpha, \sigma) \leq \delta + 2\delta = 3\delta\,.$$

Thus, we only need to argue that when $d_{\mathrm{F}}^{\min}(\mathcal{U}, \sigma) \leq \delta$, a curve is produced, which is immediate from Lemma 3.51. □

It is also possible to turn this procedure into a 9-approximation algorithm for $d_{\mathrm{F}}^{\min}$. Suppose we are given a $10r$-separated uncertain curve. We can use decreasing exponential search with a factor of 3, starting with $\delta = r$. Suppose that for $\delta = r$, we get `True`; eventually, we switch to `False`. Let the last `True` value be x; then $3x$ must be `True`, and $x/3$ and $x/9$ must be `False`. Note that at most one value of δ can fall into the interval with the uncertain answer of the 3-decider. Then we know that $d_{\mathrm{F}}^{\min}(\mathcal{U}, \sigma) \leq 3x$ and $d_{\mathrm{F}}^{\min}(\mathcal{U}, \sigma) > 3 \cdot x/9 = x/3$. Let $\delta' = 3x$

be the returned distance, then $d_{\mathrm{F}}^{\min}(\mathcal{U}, \sigma) \leq \delta' < 9 d_{\mathrm{F}}^{\min}(\mathcal{U}, \sigma)$, so δ' is a 9-approximation to the lower bound Fréchet distance.

3.3 Algorithms for Upper Bound and Expected Fréchet Distance

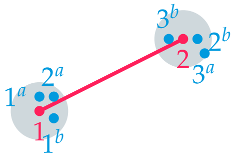
As shown in Section 3.1.1, finding the upper bound and the expected discrete and continuous Fréchet distance is hard even for simple uncertainty models. However, restricting the possible couplings or alignments between the curves makes the problem solvable in polynomial time. In this section, we use *indecisive* curves. Define a Sakoe–Chiba time band [194] in terms of reparametrisations of the curves: for a band of width w and all $t \in [0, 1]$, if $\phi_1(t) = x$, then $\phi_2(t) \in [x - w, x + w]$. In the discrete case, we can only couple point i on one curve to points $i \pm w$ on the other curve.

3.3.1 Upper Bound Discrete Fréchet Distance: Precise and Indecisive

First of all, let us discuss a simple setting. Suppose we are given a curve $\sigma = \langle q_1, \dots, q_n \rangle$ of n precise points and $\mathcal{U} = \langle U_1, \dots, U_n \rangle$ of n indecisive points, each of them having ℓ options, so for all $i \in [n]$, we have $U_i = \{p_i^1, \dots, p_i^\ell\}$. We would like to answer the following decision problem: *'If we restrict the couplings to a Sakoe–Chiba band of width w, is it true that $d_{\mathrm{dF}}^{\max}(\mathcal{U}, \sigma) \leq \delta$ for some given threshold $\delta > 0$?'* So, we want to solve the complement of the decision problem for the upper bound discrete Fréchet distance between a precise and an indecisive curve discussed previously, which is co-NP-hard in the unrestricted setting.

In a fully precise setting the discrete Fréchet distance can be computed using dynamic programming [100]. We create a table where the rows correspond to vertices of one curve, say σ, and columns correspond to vertices of the other curve, say π. Each table entry (i, j) then contains a `True` or `False` value indicating if there is a coupling between $\pi[1 : i]$ and $\sigma[1 : j]$ with maximum distance at most δ. We use a similar approach.

Suppose we position $\mathcal{U}$ to go horizontally along the table, and σ to go vertically. Consider an arbitrary column in the table and suppose that we fix the realisation of $\mathcal{U}$ up to the previous column. Then we can simply consider the new column ℓ times, each time picking a different realisation for the new point on $\mathcal{U}$, and compute the resulting



F F	F T	T T
T T	T F	F F

$$\begin{array}{|c|}\hline \mathrm{F} \\ \hline \mathrm{T} \\ \hline \end{array} \rightarrow \begin{array}{|c|c|}\hline \mathrm{F} & \mathrm{T} \\ \hline \mathrm{T} & \mathrm{F} \\ \hline \end{array} \rightarrow \begin{array}{|c|}\hline \mathrm{T} \\ \hline \mathrm{F} \\ \hline \end{array}$$

Figure 3.9. Left: An indecisive and a precise curve. Middle: Distance matrix. 'T T' in the bottom left cell means $\|1-1^a\| \le \delta$ and $\|1-1^b\| \le \delta$. Right: Computing reachability matrix, column by column. Note two reachability vectors for the second column.

reachability. As we do this for the entire column at once, we can ensure consistency of our choice of realisation. This procedure will give us a set of binary reachability vectors for the new column, each vector corresponding to a realisation. The *reachability vector* is a boolean vector that, for the cell (i, j) of the table, states whether for a particular realisation π of $\mathcal{U}[1:i]$ the discrete Fréchet distance between π and $\sigma[1:j]$ is below some threshold δ.

An important observation is that we do not need to distinguish between the realisations that give the same reachability vector: once we start filling out the next column, all we care about is the existence of some realisation leading to that particular reachability vector. So, we can keep a *set* of binary vectors corresponding to reachability in the column.

This procedure was suggested for a specific realisation. However, we can also repeat this for each previous reachability vector, only keeping the unique results. As all the realisation choices happen along $\mathcal{U}$, by treating the table column by column we ensure that we do not have issues with inconsistent choices. Therefore, repeating this procedure n times, we fill out the last column of the table. At that point, if any vector from the last column has False in the top cell, then there is some realisation $\pi \Subset \mathcal{U}$ such that $d_{\mathrm{dF}}(\pi, \sigma) > \delta$, and hence $d_{\mathrm{dF}}^{\max}(\mathcal{U}, \sigma) > \delta$.

In more detail, we use two tables: the distance matrix D, where cell (i, k, j) is True if and only if $\|p_i^k - q_j\| \le \delta$; and the dynamic program, referred to as the reachability matrix R. First of all, we initialise the distance matrix D and the reachability of the first column for all possible locations of U_1. Then we fill out R column by column. We take the reachability of the previous column and note that any cell can be reached either with a horizontal step or with a diagonal step. We need to consider various extensions of the curve $\mathcal{U}$ with one of the ℓ realisations of the current point; the distance matrix should allow the specific coupling. Assume we find that a certain cell is reachable; if allowed by the distance matrix, we can then go upwards, marking the cells above the current cell reachable, even if they are not directly reachable with a horizontal or a diagonal step. Then we just remember the newly computed vector; we make sure to only add distinct vectors. The computation is illustrated in Figure 3.9; the pseudocode is given in Algorithm 3.1.

Correctness. We use the following loop invariant to show correctness.

Lemma 3.54. *Consider column i. Every reachability vector of this column corresponds to at least one realisation of $\mathcal{U}[1 : i]$ and the discrete Fréchet distance between that realisation and $\sigma[1 : \min(n, i+w)]$; and every realisation corresponds to some reachability vector.*

Proof. The statement is trivial for the first column: we consider all ℓ possible realisations of U_1 and compute reachability of cells $(1, 1)$ to $(1, 1 + w)$ in a straightforward way.

Now suppose the statement holds for column i. As follows from the recurrence establishing the discrete Fréchet distance, the reachability of column $i + 1$ only depends on the distance matrix for column $i + 1$ and the reachability of column i. We consider every possible extension of $\mathcal{U}[1 : i]$ to $\mathcal{U}[1 : i + 1]$, as for every reachability vector of column i, we consider all ℓ options from the distance matrix for column $i + 1$. Thus, we only consider valid realisations for column $i + 1$, and we consider all of them from the point of view of reachability. □

Running time. First of all, populating the distance matrix takes time $\Theta(\ell n w)$. A call to PROPAGATE takes $\Theta(w)$ time, so initialisation of the first column of the reachability matrix takes $\Theta(\ell w)$ time. Note that, at any further point, we may have at most 2^{2w+1} distinct reachability vectors; for each of them, we make ℓ calls to PROPAGATE, taking $\Theta(4^w \ell w)$ time per column, so over all the columns we need $\Theta(4^w \ell w n)$ time. If we assume that adding an element to the set takes amortised constant time, then the previous value dominates. Finally, the check at the end takes $\Theta(4^w)$ time.

Algorithm 3.1. Finding the time-banded upper bound discrete Fréchet distance on an indecisive and a precise curve.

```
function TimeBandDFDIndPr($\mathcal{U}, \sigma, w, \delta$)
  ▷ Input constraint: $|\mathcal{U}| = |\sigma| = n$ and $0 \le w < n$
    Initialise matrix $D$ of size $n \times \ell \times (2w+1)$
    for all $i \in [n]$ do
        for all $k \in [\ell]$ do
            for all $j \in \{\max(1, i-w), \dots, \min(n, i+w)\}$ do
                $D_{i,k,j} := [d(p_i^k, q_j) \le \delta?]$
    Initialise matrix $R$ of size $n \times 2^{2w+1} \times (2w+1)$
    $R0 := \langle r_1 = \mathsf{True}, r_2 = \mathsf{False}, r_3 = \mathsf{False}, \dots, r_{w+1} = \mathsf{False} \rangle$
    for all $k \in [\ell]$ do
        $R_{1,k} :=$ Propagate($R0, D_{1,k}, 1, w, n$)
    for all $i \in [n] \setminus \{1\}$ do
        for all $A \in R_{i-1}$ do                    ▷ For each reachability vector
            $B := A \vee$ ($A$ shifted by 1)          ▷ Horizontal or diagonal step
            for all $k \in [\ell]$ do
                $C :=$ Propagate($B, D_{i,k}, i, w, n$)
                Add $C$ to set $R_i$
    $r := \mathsf{True}$
    for all $A \in R_n$ do
        $r := r \wedge A_n$
    return $r$

function Propagate($A, B, i, w, n$)
  ▷ Propagate the reachability upwards in a column
    $C := A \wedge B$                                 ▷ Step and distance matrix
    $r := \mathsf{False}$
    for all $j \in \{\max(1, i-w), \dots, \min(n, i+w)$ do
        if $B_j \wedge C_j$ then $r := \mathsf{True}$          ▷ Current cell already reachable
        else if $B_j \wedge \neg C_j \wedge r$ then $C_j := \mathsf{True}$        ▷ Vertical step
        else if $\neg B_j$ then $r := \mathsf{False}$
    return $C$
```

So, overall the algorithm runs in time $\Theta(4^w \ell n w)$. This agrees with our hardness result: for a small fixed-width time band, we get the running time of $\Theta(\ell n)$, whereas if we set $w = n - 1$ to compute the unrestricted distance, the algorithm runs in exponential time—$\Theta(4^n \ell n^2)$. We can also only store vectors that dominate in terms of `False` values, as we are interested in the worst case. This improvement reduces the running time by a factor of $\sqrt{w}$.

Theorem 3.55. *Problem* Upper Bound Discrete Fréchet *restricted to a Sakoe–Chiba time band of width w on a precise curve and an uncertain curve comprised of indecisive points with ℓ options, both of length n, can be decided in time $\Theta(4^w \ell n \sqrt{w})$ in the worst case.*

3.3.2 Upper Bound Discrete Fréchet Distance: Indecisive

Now we extend our previous result to the setting where both curves are indecisive, so instead of σ we have $\mathcal{V} = \langle V_1, \dots, V_n \rangle$, with, for each $j \in [n]$, $V_j = \{q_j^1, \dots, q_j^\ell\}$. Suppose we pick a realisation for curve $\mathcal{V}$—then we can apply the algorithm we just described. We cannot run it separately for every realisation; instead, note that the part of the realisation that matters for column i is the points from $i - w$ to $i + w$, since any previous or further points are outside the time band. So, we can fix these $2w + 1$ points and compute the column. We do so for each possible combination of these $2w + 1$ points.

Lemma 3.56. *Any reachability vector we store in column i corresponds to some realisation of the subcurves $\mathcal{U}[1 : i]$ and $\mathcal{V}[1 : \min(i + w, n)]$, and every such realisation has the resulting reachability vector stored in column i.*

Proof. First of all, consider the statement for column 1. Clearly, we consider all possible realisations of both subcurves, so the statement holds.

Now, as we move from column i to column $i+1$, we fix the realisation of points $i + 1 - w$ to $i + 1 + w$ on curve $\mathcal{V}$ and consider all the vectors stemming from the possible values of point $i - w$; as in Lemma 3.54, we cover all realisations of curve $\mathcal{U}$.

As for curve $\mathcal{V}$, note that we, again, only need the reachability from the previous column and the distance matrix from the current column, so the points before $i + 1 - w$ do not play a role for the consistency between the two, and thus they can be ignored.

So, we only get reachability vectors corresponding to valid realisations, and we do not miss any, as required. □

The running time is now $\Theta(4^w \ell^{2w+1} n w)$, as we consider all combinations of the $2w + 1$ relevant points on $\mathcal{V}$ with ℓ options per point. For small constants w and ℓ, we get $\Theta(n)$; for $w = n-1$, we get $\Theta(4^n n^2 \ell^{2n-1})$—exponential time in n. As in the previous algorithm, we can store the boolean vectors more efficiently, reducing the running time by a factor of $\sqrt{w}$.

Theorem 3.57. *Suppose we are given two indecisive curves of length n with ℓ options per indecisive point. Then we can decide whether the upper bound discrete Fréchet distance restricted to a Sakoe–Chiba band of width w is below the threshold in time $\Theta(4^w \ell^{2w+1} n \sqrt{w})$.*

3.3.3 Expected Discrete Fréchet Distance

To compute the expected discrete Fréchet distance with time bands, we need two observations:

1. For any two precise curves, there is a single threshold δ where the answer to the decision problem changes from `True` to `False`—a *critical value.* That threshold corresponds to the distance between some two points on the curves.
2. We can modify our algorithm to store associated counts with each reachability vector, obtaining the fraction of realisations that yield the answer `True` for a given threshold δ.

We can execute our algorithm for each critical value and get the cumulative distribution function $\mathbb{P}(d_{\text{dF}}(\pi, \sigma) > \delta)$ for $\pi, \sigma \Subset_{\text{U}} \mathcal{U}, \mathcal{V}$. As explained in the rest of this section, using the fact that the cumulative distribution function is a step function, we compute $d^{\mathbb{E}}_{\text{dF}}$.

Consider first the setting of one precise and one indecisive curve. Previously, we stored the reachability vectors in a set; instead, we can store a counter with each reachability vector, so that every time we get an element that is already stored, we increment the counter. We cannot use the improvement that would allow us to discard some vectors, as that would eschew the count, and we are not interested in the worst possible result now. We can implement a similar mechanism in the setting of two indecisive curves. Moreover, we can propagate the count

Algorithm 3.2. Finding the time-banded upper bound discrete Fréchet distance on two indecisive curves.

```
function TimeBandDFDIndInd($\mathcal{U}, \mathcal{V}, w, \delta$)
  ▷ Input constraint: $|\mathcal{U}| = |\mathcal{V}| = n$ and $0 \le w < n$
    Initialise matrix $D$ of size $n \times \ell \times (2w+1) \times \ell$
    for all $i \in [n]$ do
        for all $k \in [\ell]$ do
            for all $j \in \{\max(1, i-w), \dots, \min(n, i+w)\}$ do
                for all $s \in [\ell]$ do
                    $D_{i,k,j,s} := [d(p_i^k, q_j^s) \le \delta?]$
    Initialise matrix $R$ of size $n \times \ell^{2w+1} \times 2^{2w+1} \times 2w+1$
    $R0 := \langle r_1 = \mathsf{True}, r_2 = \mathsf{False}, r_3 = \mathsf{False}, \dots, r_{w+1} = \mathsf{False} \rangle$
    for all $s \in [\ell^{w+1}]$ do
        for all $k \in [\ell]$ do
            $R_{1,s,k} :=$ Propagate($R0, D_{1,k}[s], 1, w, n$)
    for all $i \in [n] \setminus \{1\}$ do
        for all $s \in [\ell^{2w+1}]$ do                         ▷ Or fewer in edge cases
          ▷ For each reachability vector with fixed realisation
            for all $A \in R_{i-1}[s]$ do
                $B := A \lor (A$ shifted by 1$)$
                for all $k \in [\ell]$ do
                    $C :=$ Propagate($B, D_{i,k}[s], i, w, n$)
                    Add $C$ to set $R_i[s]$
    $r := \mathsf{True}$
    for all $A \in R_n$ do
        for all $s \in [\ell^{2w+1}]$ do
            $r := r \land A_n[s]$
    return $r$

function Propagate($A, B, i, w, n$)
  ▷ Propagate the reachability upwards in a column
    $C := A \land B$
    $r := \mathsf{False}$
    for all $j \in \{\max(1, i-w), \dots, \min(n, i+w)$ do
        if $B_j \land C_j$ then $r := \mathsf{True}$
        else if $B_j \land \neg C_j \land r$ then $C_j := \mathsf{True}$
        else if $\neg B_j$ then $r := \mathsf{False}$
    return $C$
```

through the algorithm and in the end find the counts associated with answers `True` and `False` to the decision problem.

So, if we store the count of realisations that give us a certain reachability vector, we essentially obtain, for some value of δ,

$$\mathbb{P}(d_{\mathrm{dF}}(\pi, \sigma) > \delta) \quad \text{when } \pi, \sigma \Subset_{\mathbb{U}} \mathcal{U}, \mathcal{V}.$$

For any realisation, there is a specific value of δ—a *critical value*—that acts as a threshold between the answers `True` and `False` for that realisation, since if we fix the realisation, we just compute the regular discrete Fréchet distance. Note that that threshold must be a distance between some two points on different curves. In the case of a precise and an indecisive curve, there are $\ell n(2w + 1)$ such distances with the time band of width w; in the case of two indecisive curves, there are $\ell^2 n(2w+1)$ such distances. Therefore, if we run our algorithm for each of these critical values and record the counts of `True` and `False` for each threshold, we obtain the complete cumulative distribution function $\mathbb{P}(d_{\mathrm{dF}}(\pi, \sigma) > \delta)$ for $\pi, \sigma \Subset_{\mathbb{U}} \mathcal{U}, \mathcal{V}$.

Then we can simply find, under the time band restriction,

$$d_{\mathrm{dF}}^{\mathbb{E}(\mathbb{U})}(\mathcal{U}, \mathcal{V}) = \int_0^\infty \mathbb{P}_{\pi, \sigma \Subset_{\mathbb{U}} \mathcal{U}, \mathcal{V}}(d_{\mathrm{dF}}(\pi, \sigma) > \delta) \, \mathrm{d}\delta \, .$$

For any realisation the answer may change from `True` to `False` only at one of the critical values. So, the distribution of `True` and `False` only changes at a finite set of critical values and is constant between them; therefore, $\mathbb{P}(d_{\mathrm{dF}}(\pi, \sigma) > \delta)$ is a step function. Hence, finding the integral of interest amounts to multiplying the value of $\mathbb{P}(d_{\mathrm{dF}}(\pi, \sigma) > \delta)$ by the distance between two successive values of δ that match, and summing all the results, i.e. to finding the area under the step function by summing up the areas of the rectangles that make it up.

So, clearly, under the time band restriction, we can run one of our algorithms either $\ell n(2w + 1)$ or $\ell^2 n(2w + 1)$ times to obtain the expected discrete Fréchet distance. We show the details in Algorithm 3.3 for the two settings. We summarise this result as follows.

Theorem 3.58. *Suppose we are given an indecisive curve $\mathcal{U}$ and a precise curve σ of length n with ℓ options per indecisive point and we want to compute the expected discrete Fréchet distance constrained to a Sakoe–Chiba band of width w. Then we can run* ExpTimeBandDFDIndPr$(\mathcal{U}, \sigma, w)$ *to obtain the result in time $\Theta(4^w \ell^2 n^2 w^2)$ in the worst case.*

Algorithm 3.3. Finding the time-banded expected discrete Fréchet distance on an indecisive and a precise curve and two indecisive curves.

```
function ExpTimeBandDFDIndPr($\mathcal{U}, \sigma, w$)
  ▷ Input constraint: $|\mathcal{U}| = |\sigma| = n$ and $0 \leq w < n$
  Initialise sorted set $E$
  for all $i \in [n]$ do
    for all $k \in [\ell]$ do
      for all $j \in \{\max(1, i - w), \dots, \min(n, i + w)\}$ do
        Add $d(p_i^k, q_j)$ to sorted set $E$
  $s := E[1]$
  for $i := 1$ to $l(E) - 1$ do
    $\delta := E[i], \delta' := E[i + 1]$
    $p :=$ CntTimeBandDFDIndPr($\mathcal{U}, \sigma, w, \delta$)
    $s := s + (1 - p) \cdot (\delta' - \delta)$
  return $s$

function ExpTimeBandDFDIndInd($\mathcal{U}, \mathcal{V}, w$)
  ▷ Input constraint: $|\mathcal{U}| = |\mathcal{V}| = n$ and $0 \leq w < n$
  Initialise sorted set $E$
  for all $i \in [n]$ do
    for all $k \in [\ell]$ do
      for all $j \in \{\max(1, i - w), \dots, \min(n, i + w)\}$ do
        for all $s \in [\ell]$ do
          Add $d(p_i^k, q_j^s)$ to sorted set $E$
  $s := E[1]$
  for $i := 1$ to $l(E) - 1$ do
    $\delta := E[i], \delta' := E[i + 1]$
    $p :=$ CntTimeBandDFDIndInd($\mathcal{U}, \mathcal{V}, w, \delta$)
    $s := s + (1 - p) \cdot (\delta' - \delta)$
  return $s$

function CntTimeBandDFDIndPr($\mathcal{U}, \sigma, w, \delta$)
  ▷ Like TimeBandDFDIndPr, but returns the fraction of count of True over False for the final cell.

function CntTimeBandDFDIndInd($\mathcal{U}, \mathcal{V}, w, \delta$)
  ▷ Like TimeBandDFDIndInd, but returns the fraction of count of True over False for the final cell.
```

Proof. First of all, note that from the discussion above it immediately follows that the algorithm is correct. In the worst case, every δ that we have to add to E will be distinct, so we have $\ell n(2w + 1)$ insertions, taking in total $\Theta(\ell nw \log \ell nw)$ time. Then we run CntTimeBandDFDIndPr once per value in E, and its running time is the same as that of TimeBandDFDIndPr, so here we take time $\Theta(\ell nw \cdot 4^w \ell nw)$ in the worst case, as claimed. □

We can formalise the result similarly for the other setting.

Theorem 3.59. *Suppose we are given two indecisive curves $\mathcal{U}$ and $\mathcal{V}$ of length n with ℓ options per indecisive point and want to find the expected discrete Fréchet distance when constrained to a Sakoe–Chiba band of width w. Then we can run ExpTimeBandDFDIndInd$(\mathcal{U}, \mathcal{V}, w)$ to obtain the result in time $\Theta(4^w \ell^{2w+3} n^2 w^2)$ in the worst case.*

Proof. Again, note that from the discussion above it immediately follows that the algorithm is correct. In the worst case, we have $\ell^2 nw$ insertions, taking in total $\Theta(\ell^2 nw \log \ell nw)$ time. Then we run CntTimeBandDFDIndInd once per value in E, and its running time is the same as that of TimeBandDFDIndInd, so here we take time $\Theta(\ell^2 nw \cdot 4^w \ell^{2w+1} nw)$ in the worst case, as claimed. □

3.3.4 Upper Bound Continuous Fréchet Distance

We can adapt our time band algorithms to handle the continuous Fréchet distance. Instead of the boolean reachability vectors, we use vectors of *free space* cells, introduced by Alt and Godau [23, 117]. We now need to store reachability intervals on cell borders. The number of these intervals is limited: for any cell, the upper value of the interval is determined by the distance matrix, yielding at most ℓ^2 values; the lower value of the interval is determined by the distance matrix or by one of the cells from the same row, yielding exponential dependency on w. However, the algorithm is still polynomial-time in n.

In more detail, one could adapt the algorithms for the upper bound discrete Fréchet distance to the case when either both curves are indecisive or one is precise and one is indecisive, and we are interested in the decision problem for the Fréchet distance and not the discrete Fréchet

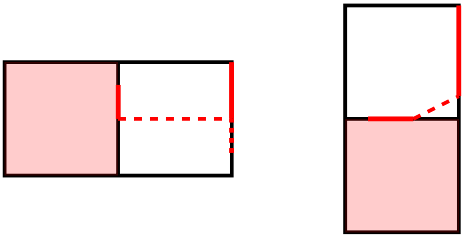


Figure 3.10. Reachability adjustments. Left: Although the dotted interval is free according to the distance matrix, only the solid interval is reachable from the cell on the left with a monotone path, assuming entire cell on the left is free. Right: The entire interval that is marked as free according to the distance matrix is reachable with a monotone path from the cell below, assuming the cell below is free.

distance. Since we are going column by column, we would need to store the reachability intervals on the vertical border of each cell.

It is simpler to see how this would work in the setting of a precise and an indecisive curve: each column now is a column of a free-space diagram, and we only need to store the intervals on the right side of the column. As we progress to the next column, we need to consider all the options from the previous column, so we need to run the same algorithm, except we store and process vectors of free-space intervals instead of True and False. One other distinction is that we do not consider diagonal steps—for the Fréchet distance doing so would not be meaningful, as the path is continuous, and the diagonal step is not distinguishable from a horizontal step followed by a vertical step, if such situation occurs.

In particular, we now take the intervals stored in the distance matrix and compute reachability based on the previous column: if a cell can be reached horizontally from the previous cell, then the lower bound of the interval in this cell may need to go up, since we can only use monotone paths. Propagate will now take the intervals that correspond to the distance matrix and the precomputed reachability and make the following adjustment: if a cell is reachable from below, then the entire interval on the right is actually reachable. See Figure 3.10 for an example of both cases.

Other than that, the algorithm is exactly the same; clearly, we can make the same adjustments to the algorithm handling two indecisive curves.

Notice that we now do not have at most 2^{2w+1} vectors per column, since we store intervals instead of boolean values, and they can be more varied. However, the number of values is still limited: for any cell, the upper value of the interval is determined by the distance matrix, so there can be at most ℓ or ℓ^2 values for the two settings. The lower value of the interval is determined by the distance matrix or by one of the cells from the same row; these may have at most ℓ or ℓ^2 values each, and there are at most $2w$ of them, so per cell we can have at most $\Theta(\ell w)$ or $\Theta(\ell^2 w)$ lower interval values and $\Theta(\ell)$ or $\Theta(\ell^2)$ upper interval values, instead of just two possible values in the discrete case. Note that for an interval, we only pick one of the possible lower bound values, and a lower bound value ultimately comes from the distance between some pair of points; and we pick one upper bound value, giving us $\Theta(\ell^2 w)$ and $\Theta(\ell^4 w)$ possible unique intervals. We also need to modify the set operations, e.g. by enumerating the possible boundaries and storing intervals as pairs of indices; adding a vector to a set would then take $O(w + \log \ell w)$ time. The running time changes accordingly, replacing 4^w with $(\ell^2 w)^{2w+1}$ and replacing $4^w \ell^{2w+1}$ with $(\ell^4 w)^{2w+1}$, but, importantly, we still have linear dependency on n, so the running time is polynomial for fixed w and ℓ.

3.3.5 Expected Continuous Fréchet Distance

We can, of course, again store the associated counts with the vectors of intervals in the algorithm. As we look at the final cell, we can sum up the counts associated with the cases where the upper right corner of this cell is reachable, and so we can find the proportion of `True` to `False` for a particular threshold δ.

We can find the critical values; now they follow in line with those discussed by Alt and Godau [23, 117]. The number of the critical values is different: case 1, where we look at the start and end points, now yields $\Theta(\ell^2)$ events; case 2, where we look at two neighbouring cells, so at the distance between a segment and a point, yields $\Theta(\ell^3 nw)$ events; and case 3, where we look at the distance between a segment and two points, yields $\Theta(\ell^4 nw^2)$ events.

Otherwise, we can run Algorithm 3.3 on the new critical values, calling instead the counting version for the continuous Fréchet distance.

This way we can compute the expected Fréchet distance restricted to a Sakoe–Chiba band in time polynomial in n for fixed w and ℓ.

Theorem 3.60. *Suppose we are given two indecisive curves of length n with ℓ options per indecisive point. Then we can decide the upper bound Fréchet distance and compute the expected Fréchet distance restricted to a Sakoe–Chiba band of fixed width w in time polynomial in n.*

3.4 Conclusions

In this chapter, we have studied the upper bound, the lower bound, and the expected Fréchet distance under various uncertainty models, namely, with uncertainty modelled as indecisive points, as line segments, and as disks. We conclude that deciding if the upper bound is above a given threshold is NP-hard in all the models we consider. This seems to translate to #P-hardness for computing the expected Fréchet distance under the uniform distribution. We do not have reason to believe that the variants of the expected Fréchet distance not covered here are easier. The lower bound problem presents an interesting trade-off, though: while the problem of deciding whether the lower bound is below a given threshold is still NP-hard for the continuous Fréchet distance for uncertain points modelled as line segments, the problem becomes tractable when either the uncertainty regions or the distance measure (or both) are discrete. We conjecture that the continuous Fréchet distance for uncertain points modelled as disks (or other continuous regions) is no easier than for line segments.

In Chapter 4, we continue our discussion on the topic. As it turns out, in one dimension, some problems remain NP-hard, while others become solvable in polynomial time.

CHAPTER 4

Similarity of Uncertain Curves in 1D

We have hopefully convinced the reader in Chapter 3 that it is important to consider the Fréchet distance under uncertainty. Unfortunately, as we have discovered, many variants turn out to be NP-hard or even #P-hard. In this chapter, we turn our attention to one-dimensional curves in order to find the source of the computational complexity of the problem. We present an efficient algorithm for computing the lower bound Fréchet distance with imprecision modelled as intervals, a setting akin to the one where the problem was NP-hard in two dimensions. We further generalise this approach to a framework applicable in higher dimensions and restricted settings; it does not guarantee polynomial-time solutions.

We also study the weak Fréchet distance, which, to our knowledge, has not been studied in the uncertain setting before. We give a polynomial-time algorithm that solves the lower bound problem in 1D. In contrast to that, we show that the problem is NP-hard in 2D, and that the discrete weak Fréchet distance is NP-hard already in 1D. We summarise these results in Table 4.1.

The table provides an interesting insight. First of all, it appears that for continuous distances the dimension matters, whereas for the discrete ones the results are the same both in 1D and 2D. Moreover, it

Table 4.1. Complexity results for the lower bound problems for uncertain curves.

	Fréchet distance		weak Fréchet distance	
	discrete	continuous	discrete	continuous
1D	polynomial [18]	polynomial	NP-hard	polynomial
2D	polynomial [18]	NP-hard (Ch. 3)	NP-hard	NP-hard

may be surprising that discretising the problem has a different effect: for the Fréchet distance it makes it easier, while for the weak Fréchet distance the problem becomes harder. We discuss the polynomial-time algorithm for the Fréchet distance in 1D in Section 4.2. We give the algorithm for the weak Fréchet distance in 1D in Section 4.4.1 and show NP-hardness for the weak (discrete) Fréchet distance in Section 4.4.2.

Finally, we also turn our attention to the problem of maximising the Fréchet distance, or finding the upper bound. We have shown that the problem is NP-hard in 2D for several uncertainty models, including indecisive points, both for the discrete and the continuous Fréchet distance (Chapter 3). We strengthen that result by presenting a similar construction that already shows NP-hardness in 1D. While the NP-hardness in the indecisive model supersedes the same result in 2D, other models are not directly comparable between one and two dimensions. The proof is given in Section 4.3.

Preliminaries. Building up on the general definitions of Chapter 2, observe that an uncertain point in one dimension is a set $u \subseteq \mathbb{R}$.[1] An *indecisive* point is a finite set of numbers $u = \{x_1, \ldots, x_\ell\}$. An *imprecise* point is a closed interval $u = [x_1, x_2]$. Note that a precise point is a special case of both types of points.

4.1 Lower Bound Fréchet Distance: General Approach

In this section, we consider the following decision problem.

Problem 4.1. Given two uncertain curves $\mathcal{U} = \langle u_1, \ldots, u_m \rangle$ and $\mathcal{V} = \langle v_1, \ldots, v_n \rangle$ in $\Upsilon = \mathbb{R}^d$ for some $d, m, n \in \mathbb{N}$ and a threshold $\delta > 0$, decide if $d_\mathrm{F}^{\min}(\mathcal{U}, \mathcal{V}) \leq \delta$.

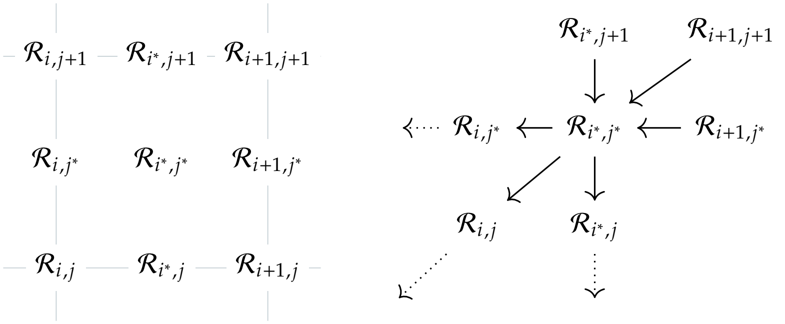
---

[1] To enable natural naming, we denote uncertain points with lowercase letters here.

Note that this problem formulation is general both in terms of the shape of uncertainty regions and the dimension of the problem. We propose an algorithmic framework that solves this problem. As shown previously (Chapter 3), the problem is NP-hard in 2D for vertical line segments as uncertainty regions, but admits a simple dynamic program for indecisive points in 2D. So, in many uncertainty models, especially in higher dimensions, the following approach will not result in an efficient algorithm. However, our approach is general in that it can be instantiated in restricted settings, e.g. in 2D assuming that the segments of the curves can only be horizontal or vertical. The inherent complexity of the problem appears to be related to the number of directions to consider, with the infinite number in 2D without restrictions and two directions in 1D. We conjecture that in this restricted setting the approach yields a polynomial-time algorithm; verifying this and making a more general statement delineating the hardness of restricted settings are both interesting open problems. Our approach shows a straightforward way to engineer an algorithm for various restricted settings in arbitrary dimension, but we cannot make any statements about its efficiency in most settings. To illustrate the approach, we instantiate it in 1D and analyse its efficiency in Section 4.2. The interested reader might refer to that section for a more intuitive explanation of the approach.

First we introduce some extra notation. Recall $Y = \mathbb{R}^d$. For $i \in [m]$, denote $\mathcal{U}_i = \langle u_1, \dots, u_i \rangle$ and $\mathcal{U}_i^* = \langle u_1, \dots, u_i, Y \rangle$. We call $\mathcal{U}_i$ and $\mathcal{U}_i^*$ the *subcurve* and the *free subcurve* of $\mathcal{U}$ at i, respectively. Intuitively, a realisation of $\mathcal{U}_i^*$ extends a realisation of $\mathcal{U}_i$ by a single edge whose final vertex position is unrestricted. Let $S := S^{d-1}$ be the unit $(d-1)$-sphere. Denote the *direction* of the ith edge $\pi[i : i+1]$ of a realisation π by $\mathbf{d}_i(\pi) \subseteq S$. For example, in 1D there are only two options, in 2D the directions can be picked from a unit circle, in 3D from a unit sphere, etc. In the degenerate case where the edge has length 0 (or π has no ith edge), let $\mathbf{d}_i(\pi) = S$.

We want to find realisations $\pi \Subset \mathcal{U}$ and $\sigma \Subset \mathcal{V}$ such that π and σ have Fréchet distance at most δ. Call such a pair (π, σ) a δ-realisation of $(\mathcal{U}, \mathcal{V})$. Intuitively, we want to create a dynamic program on the two curves that keeps track of the possible realisations of the current uncertain points and the allowed edge directions to extend the prefix curves, so that the entire prefix curves have the Fréchet distance at most δ. Recall that two polygonal curves $\pi\colon [1, i] \to Y$ and $\sigma\colon [1, j] \to$



(a) The sets on a cell of a regular free-space diagram.

(b) Dependencies of the dynamic program. $a \longrightarrow b$ means a depends on b.

Figure 4.1. Illustration for the dynamic program of Lemma 4.2.

Y have Fréchet distance $d_{\mathrm{F}}(\pi, \sigma)$ at most δ if and only if there exist *reparametrisations* (non-decreasing surjections) $\alpha\colon [0,1] \longrightarrow [1,i]$ and $\beta\colon [0,1] \longrightarrow [1,j]$ such that the path $(\pi \circ \alpha, \sigma \circ \beta)$ lies in the δ-free space, defined as $\mathcal{F}_\delta = \{(p,q) \in Y \times Y \mid \|p - q\| \le \delta\}$. For δ-close (free) subcurves of $\mathcal{U}$ at i and $\mathcal{V}$ at j, we capture their pairs of endpoints and final directions using $\mathcal{R}_{i,j}, \mathcal{R}_{i,j^*}, \mathcal{R}_{i^*,j}, \mathcal{R}_{i^*,j^*} \subseteq Y \times Y \times S \times S$:

$$\begin{aligned}
\mathcal{R}_{i,j} &= \Big\{(\pi(i), \sigma(j), s, t) &&\Big|\, \pi \Subset \mathcal{U}_i, \sigma \Subset \mathcal{V}_j,\\
&&& \quad s \in \mathbf{d}_i(\pi), t \in \mathbf{d}_j(\sigma), d_{\mathrm{F}}(\pi, \sigma) \le \delta\Big\},\\
\mathcal{R}_{i,j^*} &= \Big\{(\pi(i), \sigma(j+1), s, t) &&\Big|\, \pi \Subset \mathcal{U}_i, \sigma \Subset \mathcal{V}^*_j,\\
&&& \quad s \in \mathbf{d}_i(\pi), t \in \mathbf{d}_j(\sigma), d_{\mathrm{F}}(\pi, \sigma) \le \delta\Big\},\\
\mathcal{R}_{i^*,j} &= \Big\{(\pi(i+1), \sigma(j), s, t) &&\Big|\, \pi \Subset \mathcal{U}^*_i, \sigma \Subset \mathcal{V}_j,\\
&&& \quad s \in \mathbf{d}_i(\pi), t \in \mathbf{d}_j(\sigma), d_{\mathrm{F}}(\pi, \sigma) \le \delta\Big\},\\
\mathcal{R}_{i^*,j^*} &= \Big\{(\pi(i+1), \sigma(j+1), s, t) &&\Big|\, \pi \Subset \mathcal{U}^*_i, \sigma \Subset \mathcal{V}^*_j,\\
&&& \quad s \in \mathbf{d}_i(\pi), t \in \mathbf{d}_j(\sigma), d_{\mathrm{F}}(\pi, \sigma) \le \delta\Big\}.
\end{aligned}$$

Note that for $\pi \Subset \mathcal{U}_i$, i is the final vertex, so $\mathbf{d}_i(\pi) = S$. Therefore, $\mathcal{R}_{i,j}$ captures the reachable subset of $Y \times Y$ for the realisations of the last points of the prefixes, and the two other dimensions contain all points from S to capture that we may proceed in any allowed direction. The reachable subset of $Y \times Y$ here refers to the subset of $Y \times Y$ where the realisations of the last points may be, taking into account the entire prefixes of the curves. The set $\mathcal{R}_{i^*,j}$ captures the reachable subset of $Y \times Y$

for the point in the reparametrisation where we are between vertices i and $i+1$ on $\mathcal{U}$ and at j on $\mathcal{V}$; we have not restricted the range to u_{i+1} yet. The allowed directions for parameter s now depend on how we reached this point in the reparametrisation, since segments connecting realisations are straight line segments, and the direction needs to be kept consistent once chosen. From this description the reader can deduce what the other sets capture by symmetry. See also Figure 4.1a, where the sets are positioned as in a regular free-space diagram, replacing the edges, vertices, and cells.

To solve the decision problem, we must decide whether $\mathcal{R}_{m,n}$ is non-empty. If so, then there are realisations of $\mathcal{U}_m \equiv \mathcal{U}$ and $\mathcal{V}_n \equiv \mathcal{V}$ that, after placing all the previous realisations, result in curves with the Fréchet distance at most δ. We compute $\mathcal{R}_{\cdot,\cdot}$ using dynamic programming. We illustrate the propagation dependencies in Figure 4.1b and make them explicit in Lemma 4.2.

Lemma 4.2. *Define the set*

$$\star(A) := \left\{(p+\lambda s, q+\mu t, s, t) \,\middle|\, (p,q,s,t) \in A \text{ and } \lambda, \mu \geq 0\right\}.$$

We have

$$\begin{aligned}
&\mathcal{R}_{\cdot,0} = \mathcal{R}_{0,\cdot} = \emptyset\,,\\
&\mathcal{R}_{i+1,j^*} &&= \left\{(p,q,s,t) \in u_{i+1} \times \ Y \ \times S \times S \,\middle|\, (p,q,\cdot,t) \in \mathcal{R}_{i^*,j^*}\right\},\\
&\mathcal{R}_{i^*,j+1} &&= \left\{(p,q,s,t) \in \ Y \ \times v_{j+1} \times S \times S \,\middle|\, (p,q,s,\cdot) \in \mathcal{R}_{i^*,j^*}\right\},\\
&\mathcal{R}_{i+1,j+1} &&= \left\{(p,q,s,t) \in u_{i+1} \times v_{j+1} \times S \times S \,\middle|\, (p,q,\cdot,\cdot) \in \mathcal{R}_{i^*,j^*}\right\},\\
&\mathcal{R}_{0^*,0^*} &&= \mathcal{F}_\delta \times S \times S\,,\\
&\mathcal{R}_{i^*,j^*} &&= (\mathcal{F}_\delta \times S \times S) \cap \star(\mathcal{R}_{i,j} \cup \mathcal{R}_{i^*,j} \cup \mathcal{R}_{i,j^*}) \quad \textit{for } i > 0 \textit{ or } j > 0\,.
\end{aligned}$$

Proof. The first equation holds because the empty function has no reparametrisation, so the Fréchet distance of any pair of realisations is infinite. The equation for $\mathcal{R}_{i+1,j^*}$ holds because for $\pi \Subset \mathcal{U}_{i+1}$, $\mathbf{d}_{i+1}(\pi) = S$, and the only additional constraint that a realisation of $\mathcal{U}_{i+1}$ has over one of $\mathcal{U}_i^*$ is that the final vertex lies in u_{i+1}. Using symmetric properties on $\mathcal{V}$, we obtain the equations for $\mathcal{R}_{i^*,j+1}$ and $\mathcal{R}_{i+1,j+1}$. The equation for $\mathcal{R}_{0^*,0^*}$ concerns curves π and σ consisting of a single vertex, so $\mathbf{d}_0(\pi) = \mathbf{d}_0(\sigma) = S$, and $d_\mathrm{F}(\pi,\sigma) \leq \delta$ if and only if $(\pi(1), \sigma(1)) \in \mathcal{F}_\delta$. The equation for $\mathcal{R}_{i^*,j^*}$ remains. First we show that the right-hand side is contained in $\mathcal{R}_{i^*,j^*}$. Suppose that π and σ form a witness for $(p, q, s, t) \in \mathcal{R}_{i,j} \cup \mathcal{R}_{i^*,j} \cup \mathcal{R}_{i,j^*}$. We obtain realisations $\pi^* \Subset \mathcal{U}_i^*$ and $\sigma^* \Subset \mathcal{V}_j^*$ by extending the last edge of π and σ in the direction it is already going (or adding a new edge in an arbitrary direction if $\pi \Subset \mathcal{U}_i$ or $\sigma \Subset \mathcal{V}_j$), to $(p + \lambda s, q + \mu t)$. If $(p + \lambda s, q + \mu t) \in \mathcal{F}_\delta$, then, by convexity of $\mathcal{F}_\delta$, the extensions of the last edges have Fréchet distance at most δ (since the points at which the extension starts have distance at most δ), so $(p + \lambda s, q + \mu t, s, t) \in \mathcal{R}_{i^*,j^*}$. Conversely, we show that the right-hand side contains $\mathcal{R}_{i^*,j^*}$. Let $\pi^* \Subset \mathcal{U}_i^*$ and $\sigma^* \Subset \mathcal{V}_j^*$ together with reparametrisations $\alpha\colon [0, 1] \to [1, i + 1]$ and $\beta\colon [0, 1] \to [1, j + 1]$ form a witness that $(p, q, s, t) \in \mathcal{R}_{i^*,j^*}$. Then, for any $x \in [0, 1]$, the restrictions π_x of π^* and σ_x of σ^* to the domains $[1, \alpha(x)]$ and $[1, \beta(x)]$ have Fréchet distance at most δ. Because α and β are non-decreasing surjections, whenever $i > 0$ or $j > 0$, there exists some x such that

1. $\alpha(x) = i$ and $\beta(x) = j$, in which case $\pi_x \Subset \mathcal{U}_i$ and $\sigma_x \Subset \mathcal{V}_j$, or
2. $\alpha(x) > i$ and $\beta(x) = j$, in which case $\pi_x \Subset \mathcal{U}_i^*$ and $\sigma_x \Subset \mathcal{V}_j$, or
3. $\alpha(x) = i$ and $\beta(x) > j$, in which case $\pi_x \Subset \mathcal{U}_i$ and $\sigma_x \Subset \mathcal{V}_j^*$.

Note that if $i = 0$, only the second case applies, and if $j = 0$, only the third case applies. In each case, the last edge of π^* and σ^* extends the ith and jth edge of π_x and σ_x, respectively. So (π_x, σ_x) forms a witness that (p, q, s, t) is contained in the right-hand side. □

Simplifying the approach. Due to their dimension, the sets above can be impractical to work with. However, for the majority of these sets, at least one of the factors S carries no additional information, as formulated below. Denote by Pr_c the projection map of the c-th component, so that $\mathrm{Pr}_1\colon (p, q, s, t) \mapsto p$, and in general $\mathrm{Pr}_{c_1,\dots,c_k}(x) = (\mathrm{Pr}_{c_1}(x), \dots, \mathrm{Pr}_{c_k}(x))$. The equations of Lemma 4.2 imply the equivalences

$$\begin{aligned} (p, q, s, t) \in \mathcal{R}_{i,j} &\iff (p, q) \in \mathrm{Pr}_{1,2}(\mathcal{R}_{i,j})\,, \\ (p, q, s, t) \in \mathcal{R}_{i^*,j} &\iff (p, q, s) \in \mathrm{Pr}_{1,2,3}(\mathcal{R}_{i^*,j})\,, \\ (p, q, s, t) \in \mathcal{R}_{i,j^*} &\iff (p, q, t) \in \mathrm{Pr}_{1,2,4}(\mathcal{R}_{i,j^*})\,. \end{aligned}$$

Consequently, to find $\mathcal{R}_{i,j}$, $\mathcal{R}_{i^*,j}$, and $\mathcal{R}_{i,j^*}$, it suffices to compute the projections above. This simplifies the prior dependencies, see Figure 4.2.

$$\begin{tikzcd}
 & & \mathrm{Pr}_{1,2,3}(\mathcal{R}_{i^*,j+1}) \arrow[d] & \mathrm{Pr}_{1,2}(\mathcal{R}_{i+1,j+1}) \arrow[dl] \\
 {} & \mathrm{Pr}_{1,2,4}(\mathcal{R}_{i,j^*}) \arrow[l, dotted] & \mathcal{R}_{i^*,j^*} \arrow[l] \arrow[dl] \arrow[d] & \mathrm{Pr}_{1,2,4}(\mathcal{R}_{i+1,j^*}) \arrow[l] \\
 & \mathrm{Pr}_{1,2}(\mathcal{R}_{i,j}) \arrow[dl, dotted] & \mathrm{Pr}_{1,2,3}(\mathcal{R}_{i^*,j}) \arrow[d, dotted] & \\
 {} & & {} &
\end{tikzcd}$$

Figure 4.2. Simplified dependencies with projections following Lemma 4.2.

$$\begin{aligned}
\mathrm{Pr}_{1,2}(\mathcal{R}_{i+1,j+1}) &= (u_{i+1} \times v_{j+1}) \cap \mathrm{Pr}_{1,2}(\mathcal{R}_{i^*,j^*}) \\
&= (u_{i+1} \times \quad Y \quad) \cap \mathrm{Pr}_{1,2}(\mathcal{R}_{i^*,j+1}) \\
&= (\quad Y \quad \times v_{j+1}) \cap \mathrm{Pr}_{1,2}(\mathcal{R}_{i+1,j^*})\,.
\end{aligned}$$

Instantiating the approach. The dynamic program of Lemma 4.2 can naturally be adapted to constrained realisations whose edge directions are to be drawn from a subset $S' \subseteq S^{d-1}$, by replacing S by S', so the framework can be used for restricted settings in 2D. For $S = S^{d-1}$ the complexity of $\mathcal{R}_{i,j}$ can be exponential, so it can be useful to restrict the problem.

We can look at the construction used to prove NP-hardness of the problem in 2D (Section 3.1.2) as an example for our approach. There the curve $\mathcal{V}$ is precise, so each v_j is a single point and each t is predetermined, and curve $\mathcal{U}$ consists of uncertainty regions that are vertical line segments, so each u_i has a fixed x-coordinate and a range of y-coordinates. If we now exclude the fixed values from our propagation, we get to track pairs (y, s) of the feasible y-coordinates on the current interval and the directions. We start with a single region. The hardness construction uses gadgets on the precise curve to force the uncertain curve to go through certain points. In our approach, this means that we keep restricting the set of feasible directions while passing by vertices on $\mathcal{V}$, and eventually each point in the starting region gives rise to two disjoint reachable points on one of the following uncertainty regions. So we can use our algorithm to correctly track the feasible y-coordinates through the construction; however, we would need to keep track of regions of exponential complexity, which is, predictably, inefficient.

Therefore, it is important to analyse the complexity of the propagated regions to determine whether our approach gives rise to an efficient algorithm. To illustrate our approach, we use it in the 1D case to devise an efficient algorithm in Section 4.2.

4.2 Lower Bound Fréchet Distance: One Dimension

In this section, we instantiate the approach of Section 4.1 in 1D and analyse its efficiency. We first show the formal definitions that result from this process, and then give some intuition for how the resulting algorithm works in 1D.

We use S^0 for S, so there are only two directions: positive x-direction and negative x-direction. We make use of the projections interpretation and split the projections into two regions based on the value of the relevant direction; then all the regions we maintain are in $\mathbb{R}^2$ and have a geometric interpretation as feasible combinations of realisations of the last uncertain points on the prefixes of the curves. We omit $\mathcal{R}_{i,j}$ from our computations except for checking whether $\mathcal{R}_{m,n}$ is non-empty. As follows from the definition of the sets, $\mathcal{R}_{i,j} \subseteq \mathcal{R}_{i^*,j}$ and $\mathcal{R}_{i,j} \subseteq \mathcal{R}_{i,j^*}$, so we can simplify the computation of $\mathcal{R}_{i^*,j^*}$, and then we do not need the explicit computation of $\mathcal{R}_{i,j}$. Furthermore, we do not compute any of $\mathcal{R}_{i^*,j^*}$ explicitly, opting instead to substitute them into the relevant expressions. Therefore, we maintain the sets $\mathcal{R}_{i,j^*}$ and $\mathcal{R}_{i^*,j}$, splitting each into two based on the relevant direction. Based on our earlier free-space cell interpretation (see Figure 4.1a), call the directions along $\mathcal{U}$ *right* and *left* and call the directions along $\mathcal{V}$ *up* and *down*. We then have the following mapping from the regions of Section 4.1 to the simpler intuitive regions of this section.

$$\begin{aligned}
U_{i,j} &= \{(p,q) \mid (p,q,\cdot,t) \in \mathcal{R}_{i,j^*} \land t = 1\}\,,\\
D_{i,j} &= \{(p,q) \mid (p,q,\cdot,t) \in \mathcal{R}_{i,j^*} \land t = -1\}\,,\\
R_{i,j} &= \{(p,q) \mid (p,q,s,\cdot) \in \mathcal{R}_{i^*,j} \land s = 1\}\,,\\
L_{i,j} &= \{(p,q) \mid (p,q,s,\cdot) \in \mathcal{R}_{i^*,j} \land s = -1\}\,.
\end{aligned}$$

It is also easier to express the $\star$ operator of Lemma 4.2 in this setting. Depending on which of the directions we consider fixed because we already committed to a direction, the propagation through the cell interior works by adding either a quadrant or a half-plane to every point in the starting region; we can denote this with a Minkowski sum. Based on these considerations, we give the following simplified definition.

Formal definition. Denote $\mathbb{R}^- = \{x \in \mathbb{R} \mid x \leq 0\}$ and $\mathbb{R}^+ = \{x \in \mathbb{R} \mid x \geq 0\}$. Consider the space $\mathbb{R} \times \mathbb{R}$ of the coordinates of the two curves in 1D. We are interested in what is feasible within the *interval free space,* which in this space turns out to be a band around the line $y = x$ of width 2δ in L_1-distance called $\mathcal{F}_\delta$. For notational convenience, define the following regions (see Figure 4.3):

$$\mathcal{F}_\delta = \left\{(x, y) \in \mathbb{R}^2 \,\middle|\, |x - y| \leq \delta\right\},$$
$$I_i = (u_i \times \mathbb{R}) \cap \mathcal{F}_\delta\,, \qquad J_j = (\mathbb{R} \times v_j) \cap \mathcal{F}_\delta\,.$$

We use dynamic programming, similarly to the standard free-space diagram for the Fréchet distance; however, we propagate reachable subsets of uncertainty regions on the two curves. The propagation in the interval-free-space diagram consists of starting anywhere within the current region and going in restricted directions, since we need to distinguish between going in the positive and the negative x-direction along both curves. We introduce the notation for restricting the directions in the form of quadrants, half-planes, and slabs:

$$Q_{LD} = \mathbb{R}^- \times \mathbb{R}^-, Q_{LU} = \mathbb{R}^- \times \mathbb{R}^+, Q_{RD} = \mathbb{R}^+ \times \mathbb{R}^-, Q_{RU} = \mathbb{R}^+ \times \mathbb{R}^+,$$
$$H_L = \mathbb{R}^- \times \mathbb{R}\,, \quad H_R = \mathbb{R}^+ \times \mathbb{R}\,, \quad H_D = \mathbb{R} \times \mathbb{R}^-, \quad H_U = \mathbb{R} \times \mathbb{R}^+,$$
$$S_L = \mathbb{R}^- \times \{0\}\,, \ \ S_R = \mathbb{R}^+ \times \{0\}\,, \ \ S_D = \{0\} \times \mathbb{R}^-, \ \ S_U = \{0\} \times \mathbb{R}^+.$$

We introduce notation for propagating in these directions from a region by taking the appropriate Minkowski sum, denoted with $\oplus$. For $a, b \in \{L, R, U, D\}$ and a region X,

$$X^a = X \oplus H_a\,, \qquad X^{ab} = X \oplus Q_{ab}\,, \qquad X^{a0} = X \oplus S_a\,.$$

Now we can discuss the propagation. We start with the base case, where we compute the feasible combinations for the boundaries of the cells of a regular free-space diagram corresponding to the first vertex on one of the curves. For the sake of better intuition we do not use $(0, 0)$ as the base case here. We fix our position to the first vertex on $\mathcal{U}$ and see how far we can go along $\mathcal{V}$; and the other way around. As we are bound to the same vertex on $\mathcal{U}$, as we go along $\mathcal{V}$, we keep restricting

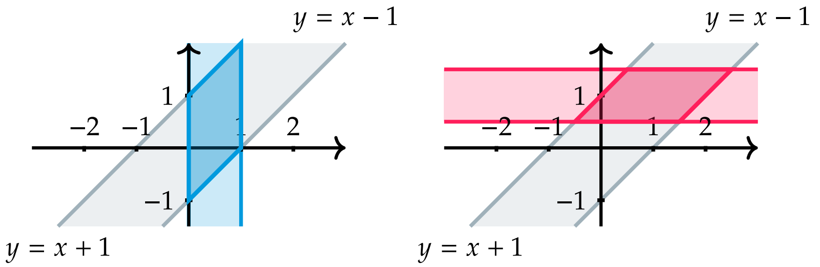


Figure 4.3. On the left, the filled region is $I_i = (u_i \times \mathbb{R}) \cap \mathcal{F}_\delta$ for $u_i = [0, 1]$. On the right, the filled region is $J_j = (\mathbb{R} \times v_j) \cap \mathcal{F}_\delta$ for $v_j = [0.5, 1.5]$. Here $\delta = 1$.

the feasible realisations of u_1. Thus, we cut off unreachable parts of the interval as we propagate along the other curve. We do not care about the direction we were going in after we cross a vertex on the curve where we move. So, if we stay at u_1 and we cross over v_j, then we are free to go both in the negative and the positive direction of the x-axis to reach a realisation of v_{j+1}. We get the following expressions, where $U_{i,j}$ denotes the propagation upwards from the pair of vertices u_i and v_j and propagation down, left, and right is defined similarly:

$$\begin{aligned}
U_{1,1} &= (I_1 \cap J_1)^{U0} \cap \mathcal{F}_\delta\,, \quad D_{1,1} = (I_1 \cap J_1)^{D0} \cap \mathcal{F}_\delta\,,\\
R_{1,1} &= (I_1 \cap J_1)^{R0} \cap \mathcal{F}_\delta\,, \quad L_{1,1} = (I_1 \cap J_1)^{L0} \cap \mathcal{F}_\delta\,,\\
U_{1,j+1} &= ((U_{1,j} \cup D_{1,j}) \cap J_{j+1})^{U0} \cap \mathcal{F}_\delta\,,\\
D_{1,j+1} &= ((U_{1,j} \cup D_{1,j}) \cap J_{j+1})^{D0} \cap \mathcal{F}_\delta\,,\\
R_{i+1,1} &= ((R_{i,1} \cup L_{i,1}) \cap I_{i+1})^{R0} \cap \mathcal{F}_\delta\,,\\
L_{i+1,1} &= ((R_{i,1} \cup L_{i,1}) \cap I_{i+1})^{L0} \cap \mathcal{F}_\delta\,.
\end{aligned}$$

Once the boundary regions are computed, we can propagate:

$$\begin{aligned}
U_{i+1,j} &= (U^{U}_{i,j} \cup R^{RU}_{i,j} \cup L^{LU}_{i,j}) \cap I_{i+1}\,,\ D_{i+1,j} = (D^{D}_{i,j} \cup R^{RD}_{i,j} \cup L^{LD}_{i,j}) \cap I_{i+1}\,,\\
R_{i,j+1} &= (R^{R}_{i,j} \cup U^{RU}_{i,j} \cup D^{RD}_{i,j}) \cap J_{j+1}\,, L_{i,j+1} = (L^{L}_{i,j} \cup U^{LU}_{i,j} \cup D^{LD}_{i,j}) \cap J_{j+1}\,.
\end{aligned}$$

Finally, we check if the last vertex combination is feasible:

$$\big((R_{m-1,n} \cup L_{m-1,n}) \cap I_m\big) \cup \big((U_{m,n-1} \cup D_{m,n-1}) \cap J_n\big) \neq \emptyset\,.$$

Intuition. If the consecutive regions are always disjoint, we do not need to consider the possible directions: we always know (in 1D) where the next region is, and thus what direction we take. However, if the regions may overlap, it may be that for different realisations of a curve a segment goes in the positive or in the negative direction. The propagation we compute is based on the parameter space where we look at whether we have reached a certain vertex on each curve yet, inspired by the traditional free-space diagram. It may be that we pass by several vertices on, say, $\mathcal{V}$ while moving along a single segment on $\mathcal{U}$. The direction we choose on $\mathcal{U}$ needs to be kept consistent as we compute the next regions, otherwise we might include realisations that are invalid as feasible solutions. Therefore, we need to keep track of the chosen direction, reflected by the pair (s, t) in the general approach and the separate sets in this section. Otherwise, these regions in 1D are simply the feasible pairs of realisations of the last vertices on the prefixes of the curves.

It is helpful to think of the approach in terms of interval-free-space diagrams. Consider a combination of specific vertices on the two curves, say, u_i and v_j, and suppose that we want to stay at u_i but move to v_{j+1}. Which realisations of u_i, v_j, and v_{j+1} can we pick that allow this move to stay within the 2δ-band?

Suppose the x-coordinate of the diagram corresponds to the x-coordinate of $\mathcal{U}$. Then we may pick a realisation for u_i anywhere in the vertical slab corresponding to the uncertainty interval for u_i, namely, in the slab $u_i \times \mathbb{R}$. The fixed realisation for u_i would then yield a vertical line. Now suppose the y-coordinate of the diagram corresponds to the x-coordinate of $\mathcal{V}$. For v_j, picking a realisation corresponds to picking a horizontal line from the slab $\mathbb{R} \times v_j$; for v_{j+1}, it corresponds to picking a horizontal line from $\mathbb{R} \times v_{j+1}$. Picking a realisation for the pair (u_i, v_j) thus corresponds to a point in $u_i \times v_j$.

Of course, we may only maintain the matching as long the distance between the matched points is at most δ. For a fixed point on $\mathcal{U}$, this corresponds to a 2δ-window for the coordinates along $\mathcal{V}$. Therefore, the allowed matchings are contained within the band defined by $y = x \pm \delta$, and when we pick the realisations for (u_i, v_j), we only pick points from $u_i \times v_j$ for which $|y - x| \leq \delta$ holds.

As we consider the propagation to v_{j+1}, note that we may not move within u_i, so the allowed realisations for the pair (u_i, v_{j+1}) are limited. In particular, we can find that region by taking the subset of $u_i \times v_{j+1}$ for which $|y - x| \leq \delta$ holds and restricting the x-coordinate further to be feasible for the pair (u_i, v_j). See Figure 4.4 for an illustration of this.

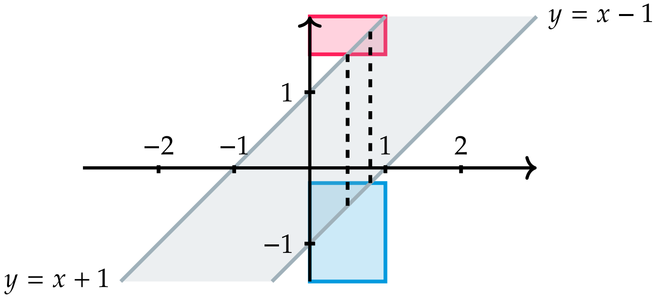


Figure 4.4. An interval-free-space diagram for $u_i = [0, 1]$, $v_j = [-1.5, -0.2]$, and $v_{j+1} = [1.5, 2]$ with $\delta = 1$. Note that the feasible realisations for u_i are $[0.5, 0.8]$.

In this figure, we know that v_{j+1} lies above v_j; if we did not know that, we would have to attempt propagation both upwards and downwards. For the second curve, the same holds.

Complexity. We now discuss the complexity of the regions we are propagating to analyse the efficiency of the algorithm presented above. We will perform the following steps.

1. Define complexity of the regions and establish the complexity of the base case.
2. Study the possible complex regions that can arise from all simple regions.
3. Study what happens to the complex regions as we propagate and conclude that the complexity is bounded by a constant.

The boundaries of the regions are always horizontal, vertical, or coincide with the boundaries of $\mathcal{F}_\delta$. A region can be thus represented as a union of (possibly unbounded) axis-aligned rectangular regions, further intersected with the interval free space. We define the *complexity* of a region as the minimal required number of such rectangular regions. Define a *simple* region as a region of complexity at most 1. Observe that a simple region is necessarily convex; and a non-simple region has to be non-convex. The illustration in Figure 4.5 shows the most general example of a simple region. An empty region is also a simple region. To enumerate the possible non-simple regions, we need to examine where higher region complexity may come from in our algorithm. To that aim, we first prove some simple statements about the propagation procedure.

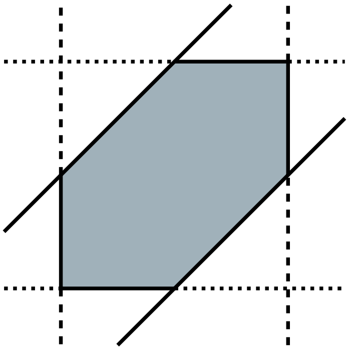


Figure 4.5. An example simple region. We get less general ones by setting any side length to 0.

First, we discuss the complexity of the regions we can get in the base case of the propagation.

Lemma 4.3. *For all $i \in [m-1]$ and $j \in [n-1]$, regions $U_{1,j}$, $D_{1,j}$, $R_{i,1}$, and $L_{i,1}$ are simple.*

Proof. Consider first the intersection $I_1 \cap J_1$. It is the intersection of a vertical slab, a horizontal slab, and the diagonal slab (interval free space). All three are convex sets, hence their intersection is also convex and uses only vertical, horizontal, and diagonal line segments, so the result is a simple region. To obtain $U_{1,1}$, $D_{1,1}$, $R_{1,1}$, and $L_{1,1}$, we take the Minkowski sum of the region with the corresponding half-slab. Both are convex, so the result again is convex; we then intersect it with the interval free space again, getting a simple region.

Now assume that $U_{1,j}$ is simple; we show that $U_{1,j+1}$ is simple. Note that for some region X, $U_{1,j} = X^{U0} \cap \mathcal{F}_\delta$ and $D_{1,j} = X^{D0} \cap \mathcal{F}_\delta$. Then

$$U_{1,j} \cup D_{1,j} = (X^{U0} \cup X^{D0}) \cap \mathcal{F}_\delta = (X \oplus (\{0\} \times \mathbb{R})) \cap \mathcal{F}_\delta\,.$$

So, we get a vertical slab the width of X, intersected with $\mathcal{F}_\delta$, so the result is convex. We then intersect the region with the simple region J_{j+1}; take Minkowski sum with a slab; and again intersect with the interval free space. Clearly, the result is convex and uses only the allowed boundaries, so we get a simple region.

The argument for $D_{1,j}$ is symmetric; the arguments for $R_{i,1}$ and $L_{i,1}$ are equally straightforward. Hence, all the regions we get in the base case are simple. □

To proceed, we need to make the relation in pairs (U, D) and (R, L) clear, so we know where the complexity may come from. Denote a half-plane with a vertical or horizontal boundary starting at coordinate s and going in direction X by H_s^X. For example, a half-plane bounded on the left by the line $x = 2$ is denoted H_2^R.

Lemma 4.4. *Take two imprecise curves $\mathcal{U}$ and $\mathcal{V}$ of lengths m and n, respectively, and let $i \in [m-1]$ and $j \in [n-1]$. Consider the pair $R_{i,j}$, $L_{i,j}$ and assume the regions are simple. Then exactly one of the following options holds:*

1. *$R_{i,j} = L_{i,j} = \emptyset$, so both regions are empty;*
2. *$R_{i,j} = J_j \cap H_s^R \neq \emptyset \wedge L_{i,j} = \emptyset$ for some s, so one region is empty and the other spawns the entire feasible range, except that it may be cut with a vertical line on the left;*
3. *$L_{i,j} = J_j \cap H_s^L \neq \emptyset \wedge R_{i,j} = \emptyset$ for some s, so one region is empty and the other spawns the entire feasible range, except that it may be cut with a vertical line on the right;*
4. *$L_{i,j} \cap R_{i,j} \neq \emptyset$, so both regions are non-empty, and they intersect.*

We can make the same statement for the pair $U_{i,j}$, $D_{i,j}$, replacing the half-planes with H_s^U and H_s^D.

Proof. We show the statement for the pair $R_{i,j}$, $L_{i,j}$. We prove the statement by induction on j. First of all, for $j = 1$, we know that either both regions are empty (case 1), or they are both non-empty and intersect (case 4), showing the claim. So let $j = j' + 1$ for the rest of the proof and assume that the lemma holds for the pair $R_{i,j'}$, $L_{i,j'}$.

We go over the possible combinations of the previous regions that are combined in the propagation and show that for any such combination, we end up in one of the cases. Recall that $R_{i,j} = R_{i,j'+1} = (R_{i,j'}^R \cup U_{i,j'}^{RU} \cup D_{i,j'}^{RD}) \cap J_{j'+1}$. Similarly, $L_{i,j} = (L_{i,j'}^L \cup U_{i,j'}^{LU} \cup D_{i,j'}^{LD}) \cap J_{j'+1}$. Consider the following cases.

- $U_{i,j'} \neq \emptyset$. Note that $U_{i,j'}^{LU} \cap U_{i,j'}^{RU} = U_{i,j'}^{U0}$, so a vertical half-slab from a lower boundary. If $U_{i,j'}^{U0} \cap J_{j'+1} \neq \emptyset$, then both $L_{i,j}$ and $R_{i,j}$ are non-empty and intersect, landing in case 4. Otherwise, suppose $U_{i,j'}^{U0} \cap J_{j'+1} = \emptyset$. This intersection can be empty due to two reasons. Firstly, $U_{i,j'}^{U0}$ may lie entirely above $J_{j'+1}$. Then $U_{i,j'}^{LU} \cap J_{j'+1} = U_{i,j'}^{RU} \cap J_{j'+1} = \emptyset$, so $U_{i,j'}$ does not contribute anything

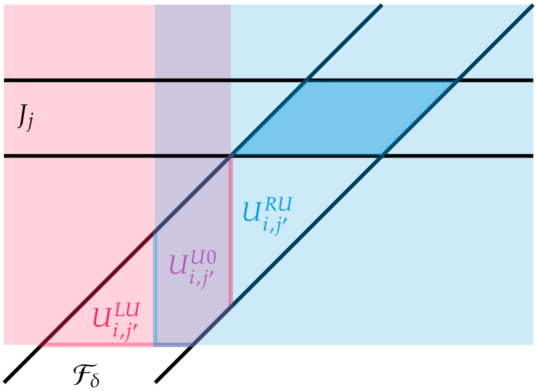


Figure 4.6. Propagation of $U_{i,j'}$ to $L_{i,j}$ and $R_{i,j}$. Note $J_j \subset \mathcal{F}_\delta$. Observe that $U^{LU}_{i,j'} \cap U^{RU}_{i,j'} = U^{U0}_{i,j'}$, and if $U^{U0}_{i,j'} \cap J_j = \emptyset$ but J_j does not lie below $U_{i,j'}$, then $U^{LU}_{i,j'} \cap J_j = \emptyset$ and $U^{RU}_{i,j'} \cap J_j = J_j$, so one of the regions is empty and the other covers the entire feasible region.

to either region; this case is considered below. Secondly, $U^{U0}_{i,j'}$ may lie entirely to the left of $J_{j'+1}$. Then we get the situation shown in Figure 4.6: it must be that $U^{LU}_{i,j} \cap J_{j'+1} = \emptyset$ and $U^{RU}_{i,j} \cap J_{j'+1} = J_{j'+1}$. This means, in particular, that $R_{i,j} = J_{j'+1} = J_j$. It might be that $L_{i,j}$ and $R_{i,j}$ are both non-empty; as $L_{i,j} \subseteq J_j$, they intersect, and so we end up in case 4. Otherwise, $L_{i,j}$ must be empty, ending up in case 2. So, whenever $U_{i,j'}$ contributes, we end up in one of the cases.

- $D_{i,j'} \neq \emptyset$. We can make arguments symmetric to the previous case, landing us in either case 4 or case 3. If $D_{i,j'}$ does not contribute to either region, we consider the next case.
- Neither $U_{i,j'}$ nor $D_{i,j'}$ contribute to $L_{i,j}$ or $R_{i,j}$, meaning we can simplify the expressions to $R_{i,j'+1} = R^R_{i,j'} \cap J_{j'+1}$ and $L_{i,j'+1} = L^L_{i,j'} \cap J_{j'+1}$. We use the induction hypothesis and distinguish between the cases for the pair $R_{i,j'}, L_{i,j'}$. Starting in case 1, we get that $L_{i,j} = R_{i,j} = \emptyset$, ending up in case 1. Starting in case 2, we get $L_{i,j} = \emptyset$, and $R_{i,j} = R_{i,j'+1} = J_{j'+1} \cap R^R_{i,j'}$. Observe that $R^R_{i,j'}$ is a half-plane that can be denoted by H^R_s for some appropriate s; depending on whether the intersection is empty, we end in either in case 1 or in case 2. Starting in case 3 is symmetric and lands us

in either case 1 or case 3. If we start in case 4, then the half-planes $R^R_{i,j'}$ and $L^L_{i,j'}$ intersect, and so for the pair $R_{i,j}, L_{i,j}$, we end up in case 4; or in case 2 or 3 if $L^R_{i,j'} \cap J_{j'+1}$ or $R^R_{i,j'} \cap J_{j'+1}$ is empty.

This covers all the cases. By induction, we conclude that the lemma holds. The proof for U, D is symmetric. ☐

Let us now introduce the higher-complexity regions.

Definition 4.5. A *staircase with k steps* is an otherwise simple region with k cut-outs on the same side of the region, each consisting of a single horizontal and a single vertical segment, introducing higher complexity. All the options for a staircase with one step (regions of complexity 2) are illustrated in Figure 4.7.

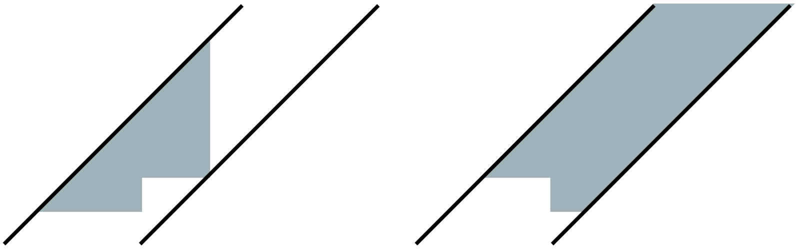
We should note that a staircase with k steps, when intersected with $\mathcal{F}_\delta$, can yield up to $k + 1$ disjoint simple regions. More specifically, every step that extends outside $\mathcal{F}_\delta$ splits a staircase of k steps into two staircases of at most $k - 1$ steps in total.

We make the following observation relating the regions in pairs $R_{i,j}$, $L_{i,j}$ and $U_{i,j}$, $D_{i,j}$.

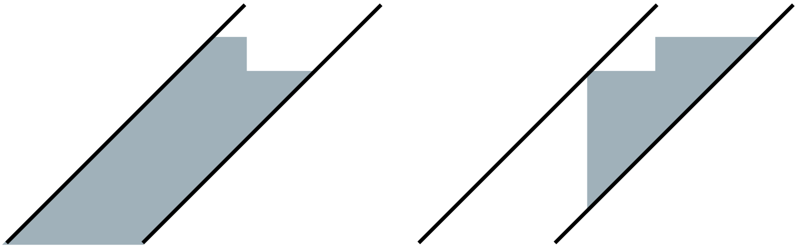
Lemma 4.6. *Take two imprecise curves $\mathcal{U}$ and $\mathcal{V}$ of lengths m and n, respectively, and let $i \in [m-1]$ and $j \in [n-1]$. Consider the pair $R_{i,j}, L_{i,j}$ and assume both regions are non-empty. If $j = 1$, the regions have the same y-coordinate for their lower and upper boundaries. If $j = j' + 1$ and the regions $U_{i,j'}, D_{i,j'}$ are simple, then the union $R_{i,j} \cup L_{i,j}$ is either simple or a staircase with one step. Furthermore, both $R_{i,j}$ and $L_{i,j}$ are either simple or staircases with one step.*

Proof. First of all, for $j = 1$, Lemma 4.3 implies that $R_{i,j}$ and $L_{i,j}$ are simple; furthermore, the propagation starts from the same region, so the y-range is the same and the statement holds. For the rest of the proof assume that $j = j' + 1$ and regions $U_{i,j'}$ and $D_{i,j'}$ are simple.

Consider the region $R_{i,j} = (R^R_{i,j'} \cup U^{RU}_{i,j'} \cup D^{RD}_{i,j'}) \cap J_j$. In principle, the union of the two quadrants may create a staircase with a single step. However, as the reader may verify, adding the half-plane to the union cannot add a step, since doing so would require a horizontal ray forming the top or the bottom boundary of the union of quadrants, which is impossible. Symmetrical arguments can be made for $L_{i,j}$. So, under the



(a) LU arrangement. **(b)** RU arrangement.



(c) LD arrangement. **(d)** RD arrangement.

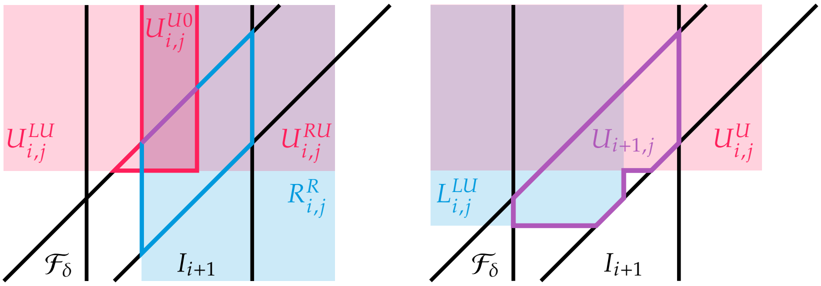
Figure 4.7. All possible combinations for a single-step staircase. Each can be further intersected by a vertical or horizontal slab (I_i or J_j) or shifted so that the boundary is affected by $\mathcal{F}_\delta$.

given assumptions the regions are always either simple or staircases with one step. Figures 4.8a and 4.8b show some examples.

Now consider the union of regions $R_{i,j} \cup L_{i,j}$:

$$R_{i,j} \cup L_{i,j} = (R^R_{i,j'} \cup L^L_{i,j'} \cup U^U_{i,j'} \cup D^D_{i,j'}) \cap J_j \,.$$

The only source of higher complexity is the union operator in the propagation. This is the union of four half-planes. If both $R_{i,j'}$ and $L_{i,j'}$ are non-empty, we know from Lemma 4.4 that they intersect, so $J_j \subseteq R^R_{i,j'} \cup L^L_{i,j'} = \mathbb{R}^2$. The same holds for the pair $U_{i,j'}$, $D_{i,j'}$. Now assume that at least one region from each pair is empty, say, $L_{i,j'}$ and $D_{i,j'}$. If one more region is empty, then one of $L_{i,j}$, $R_{i,j}$ is empty, which contradicts our assumption. Note that the union of two half-planes with perpendicular boundaries, intersected with a horizontal strip and the interval free space, can create a staircase with one step. In our particular setting, we get the staircase in the RU arrangement, shown in Figure 4.7b. Other choices for empty regions will give one of the other



(a) Taking the union of $R^{R}_{i,j}$ and $U^{RU}_{i,j}$ creates a simple region. The other region is also simple, but the union of resulting regions is a staircase.

(b) Taking the union of $L^{LU}_{i,j}$ and $U^{U}_{i,j}$ creates a staircase. Intersection with I_{i+1} preserves it: see the coloured outline of the resulting region for $U_{i+1,j}$.

Figure 4.8. Examples of staircase arrangements.

arrangements of Figure 4.7. There are no other options, so the statement about the union $R_{i,j} \cup L_{i,j}$ is proven. □

Now consider the propagation when we start from not necessarily simple regions or regions that do not match in their y-range (or x-range), as described in Lemma 4.6. Consider the complexity contribution when propagating across a cell—say, $U_{i,j}$ to $U_{i+1,j}$. To perform the propagation, we take

$$U^{U}_{i,j} = U_{i,j} \oplus H_U = U_{i,j} \oplus (\mathbb{R} \times \mathbb{R}^+)\,.$$

From the definition of the Minkowski sum, it is easy to see that for non-empty $U_{i,j}$, this results in an upper half-plane with respect to the lowest point in $U_{i,j}$. Therefore, when propagating a region across the cell, it either contributes nothing if it is empty, or it contributes a half-plane otherwise. Therefore, to establish if we can arrive at progressively more complex regions, we need to consider the other boundaries as source of complexity. This insight together with the previous results informs the following argument.

Lemma 4.7. *The regions that we propagate are either simple, or staircases with one step, so the regions have constant complexity.*

Proof. As shown in Lemma 4.3, the base regions are always simple. Consider the regions $R_{i,j}$, $L_{i,j}$ for some i and $j = j' + 1$. The proof for $U_{i,j}$, $D_{i,j}$ is symmetric. As we have just observed, the complexity of $R_{i,j'}$ and $L_{i,j'}$ is irrelevant for $R_{i,j}$, $L_{i,j}$, as they contribute a half-plane in the worst case. Furthermore, we have shown in Lemma 4.6 that if $U_{i,j'}$ and $D_{i,j'}$ are simple, then regions $R_{i,j}$, $L_{i,j}$ are at worst single-step staircases.

It remains to consider what happens as we propagate further from the regions obtained in Lemma 4.6. So suppose $R_{i,j}$, $L_{i,j}$ are obtained as in Lemma 4.6. Again, their complexity is irrelevant for the complexity of $R_{i,j+1}$, $L_{i,j+1}$, so it remains to answer the following question. Assuming no restrictions on $U_{i,j}$, $D_{i,j}$, what is the possible complexity of $U_{i+1,j}$, $D_{i+1,j}$? Consider the propagation for e.g. $U_{i+1,j} = (U^U_{i,j} \cup R^{RU}_{i,j} \cup L^{LU}_{i,j}) \cap I_{i+1}$. As follows from Lemma 4.6 and the mechanics of propagation, the region $L^{LU}_{i,j} \cup R^{RU}_{i,j}$ is either a simple region or a staircase with a single step, unbounded horizontally. Therefore, adding the half-plane of $U^U_{i,j}$ cannot increase the complexity. A symmetric argument holds for $D_{i+1,j}$. Hence, both $U_{i+1,j}$ and $D_{i+1,j}$ are again either simple or staircases with a single step.

Finally, consider the propagation through the next cell to the pair $R_{i+1,j+1}$, $L_{i+1,j+1}$. For the region $R_{i+1,j+1}$ we need to compute $U^{RU}_{i+1,j} \cup D^{RD}_{i+1,j}$. Note that

$$U_{i+1,j} \cup D_{i+1,j} = (U^U_{i,j} \cup D^D_{i,j} \cup R^R_{i,j} \cup L^L_{i,j}) \cap I_{i+1}\,,$$

and as both $R_{i,j}$ and $L_{i,j}$ are non-empty and intersect, as follows from Lemmas 4.4 and 4.6, we conclude $U_{i+1,j} \cup D_{i+1,j} = I_{i+1}$. Therefore, the region $R_{i+1,j+1}$ is formed with a union of two half-planes with parallel boundaries, and so the region is simple. The same holds for $L_{i+1,j+1}$. So, within two propagation steps we may go from simple regions to staircase regions with one step before returning to simple regions. As there are no other possibilities for the propagation, the statement of the lemma holds. □

The operations we use during propagation can be done in constant time for constant-complexity arguments. Using Lemma 4.7, we state the main result.

Theorem 4.8. *We can solve the decision problem for lower bound Fréchet distance on imprecise curves of lengths m and n in 1D in time $\Theta(mn)$.*

4.3 Upper Bound Fréchet Distance

Until this point, we have been discussing the lower bound Fréchet distance. We now turn our attention to the upper bound. The problem is known to be NP-hard in 2D in all variants we consider (Chapter 3); we show that this remains true even in 1D. Define the following problems for the discrete and continuous Fréchet distance.

Problem 4.9. UPPER BOUND (DISCRETE) FRÉCHET IN 1D: Given two uncertain trajectories $\mathcal{U}$ and $\mathcal{V}$ in 1D of lengths m and n, respectively, and a threshold $\delta > 0$, determine if $d_{\mathrm{F}}^{\max}(\mathcal{U}, \mathcal{V}) > \delta$ ($d_{\mathrm{dF}}^{\max}(\mathcal{U}, \mathcal{V}) > \delta$).

We show that these problems are NP-hard both for indecisive and imprecise models by giving a reduction from CNF-SAT. The construction we use is similar to that used in 2D; however, in 2D the desired alignment of subcurves is achieved by having one of the curves be close enough to $(0, 0)$ at all times. Here making a curve close to 0 will not work, so we need to add extra gadgets instead that can 'eat up' the alignment of the subcurves that we do not care about. We start by describing the construction and then show how it leads to the NP-hardness argument.

Suppose we are given a CNF-SAT formula C on n clauses and m variables, with $J_i \subseteq [m]$ and $K_i \subseteq [m] \setminus J_i$ for all $i \in [n]$:

$$C = \bigwedge_{i \in [n]} C_i\,, \qquad C_i = \bigvee_{j \in J_i} x_j \vee \bigvee_{k \in K_i} \neg x_k \quad \text{for all } i \in [n].$$

We define an *assignment* as a function $a \colon \{x_1, \dots, x_m\} \to \{\mathsf{True}, \mathsf{False}\}$ that assigns a value to each variable, $a(x_j) = \mathsf{True}$ or $a(x_j) = \mathsf{False}$ for any $j \in [m]$. $C[a]$ then denotes the result of substituting $x_j \mapsto a(x_j)$ in C for all $j \in [m]$. We construct two curves: curve $\mathcal{U}$ is an uncertain curve that represents the variables, and curve $\mathcal{V}$ is a precise curve that represents the structure of the formula.

Literal level. Define a *literal gadget* for curve $\mathcal{V}$:

$$\mathrm{LG}_{i,j} = \begin{cases} 0 \sqcup 1.5 & \text{if } x_j \text{ is a literal of } C_i, \\ -1.5 \sqcup 1.5 & \text{if } \neg x_j \text{ is a literal of } C_i, \\ -0.75 \sqcup 1.5 & \text{otherwise.} \end{cases}$$

Consider for now the indecisive uncertainty model. The curve $\mathcal{U}$ has an indecisive point per variable, each with two options, corresponding to True and False assignments. Define a *variable gadget* for curve $\mathcal{U}$:

$$\mathrm{VG}_j = \{-1.5, 0\} \sqcup 2.5\,.$$

Here the notation $\{-1.5, 0\}$ denotes an indecisive point with two possible locations -1.5 and 0. We interpret the position -1.5 as assigning $x_j = \mathsf{True}$ and the position 0 as assigning $x_j = \mathsf{False}$. Observe the relationship between $\mathrm{LG}_{i,j}$ and VG_j for any given $i \in [n]$: the distance between the first points of the gadgets is large if the given variable assignment turns the clause true. For instance, if a clause has the literal x_j, then the choice of $x_j = \mathsf{True}$ makes the distance between the first points $1.5 > 1$; if the literal is $\neg x_j$ and we make the same choice, then the distance is 0; and if the literal does not occur in C_i, then whichever realisation we pick, the distance is $0.75 < 1$.

Clause level. We now aggregate the literal gadgets into *clause gadgets.* Similarly, we aggregate the variable gadgets into the *variable section:*

$$\mathrm{CG}_i = 3.5 \sqcup \bigsqcup_{j\in[m]} \mathrm{LG}_{i,j}\,, \qquad \mathrm{VS} = 4.5 \sqcup \bigsqcup_{j\in[m]} \mathrm{VG}_j\,.$$

Suppose that we pick some realisation for all the variables with some function a. Pick a clause C_i. Suppose that $C_i[a] = \mathsf{True}$. This means there is at least one x_j assigned in a way that makes C_i turn true. In our construction, this means that there is at least one pair of $\mathrm{LG}_{i,j}$ and VG_j that gives a large distance between the first two points. If we are interested in just the Fréchet distance between CG_i and VS for some fixed i, we can state the following.

Lemma 4.10. *For some fixed $i \in [n]$, the (discrete) Fréchet distance between CG_i, corresponding to clause C_i, and a realisation $\pi \Subset \mathrm{VS}$, corresponding to an assignment a, is 1 iff $C_i[a] = \mathsf{False}$, and is 1.5 iff $C_i[a] = \mathsf{True}$, and there are no other possible values.*

Proof. First of all, note that the points 4.5 and 3.5 must be matched, yielding the distance of at least 1 between the curves. Furthermore, the only point within distance 1.5 of the point 2.5 that occurs at the end of every VG_j is the last point of every $\mathrm{LG}_{i,j}$, namely, 1.5. Observe that simply walking along both curves, matching point k on one curve to point k on the other curve for every k, gives us the (discrete) Fréchet distance of at most 1.5. Thus, an optimal matching will always match the point 2.5 to one of the points at 1.5. Furthermore, an optimal solution will always match the first point of $\mathrm{LG}_{i,j}$ to the indecisive point of VG_j, as the point at 2.5 is always too far. Therefore, both for the Fréchet and the discrete Fréchet distance, an optimal matching is one to one, i.e. we advance along both curves on every step. The initial synchronisation points yield the distance 1, as do the second points in the literal level gadgets; each indecisive point is matched at distance of either 0, 0.75, or 1.5. The latter case only occurs if the assignment of the variable makes the clause satisfied. So, indeed, we conclude that we can only get the distance of either 1 or 1.5, and the latter is only possible if some variable turns the clause to true, so if $C_i[a] = \mathtt{True}$. Otherwise, the clause is false, and the distance is 1. □

Formula level. We can now paste the clause gadgets together. Once we do that, we would like to have a way to freely choose a clause to align with the variable section: then, if there is a clause that is not satisfied, choosing that clause would yield a small overall distance; and if all clauses are satisfied, then any one of them will give a large distance, and so we can distinguish between whether the formula is satisfied or not. As a starting point, it is clear that we need to prepend and append something to the variable section that would catch the clauses that are not aligned with the variable section. We devise the following gadget for that:

$$\mathrm{abs} = 2.5 \sqcup \bigsqcup_{j \in [m]} (-0.5 \sqcup 0.5)\,.$$

We show that this gadget may indeed be satisfactorily aligned with any CG_i.

Lemma 4.11. *The (discrete) Fréchet distance between* abs *and any* CG_i *is* 1.

Proof. First of all, note that we must match the first synchronisation point of CG_i at 3.5 to some point on the other curve, and the only point in abs that is close enough is the point at 2.5 in the beginning. This establishes the lower bound of 1. Furthermore, we can always get the distance of 1 by walking step by step along both curves: the distance between any of -1.5, -0.75, and 0 is at most 1 to -0.5, and the distance between 1.5 and 0.5 is 1. Thus, the statement holds. □

We need as many of these gadgets as there may be misaligned clauses. In the worst case, we may align CG_1 or CG_n with VS, and so we need $n-1$ of the catch gadgets before and after VS. However, the new problem we get is that now the extra abs clauses need to be aligned with something. To that end, we devise the following gadget:

$$\mathrm{abs}^2 = 1.5 \sqcup 0.5\,.$$

Again, we show that it can perform its function.

Lemma 4.12. *The (discrete) Fréchet distance between* abs^2 *and* abs *is* 1.

Proof. First of all, note that we must match the first synchronisation point of abs at 2.5 to the point at 1.5 on abs^2, giving the lower bound of 1. Furthermore, we can always get the distance of 1 by stepping to the second point on both curves and staying at 0.5 on abs^2 while alternating between -0.5 and 0.5 on abs. Thus, the statement holds. □

Finally, we need to align these gadgets with something, but that is not too difficult, as they only have the length of 1. We define our final uncertain curves:

$$\begin{aligned} \mathcal{U} &= 1 \sqcup \Big(\bigsqcup_{i\in[n-1]} \mathrm{abs}\Big) \sqcup \mathrm{VS} \sqcup \Big(\bigsqcup_{i\in[n-1]} \mathrm{abs}\Big) \sqcup 1\,, \\ \mathcal{V} &= \Big(\bigsqcup_{i\in[n-1]} \mathrm{abs}^2\Big) \sqcup \Big(\bigsqcup_{i\in[n]} \mathrm{CG}_i\Big) \sqcup \Big(\bigsqcup_{i\in[n-1]} \mathrm{abs}^2\Big)\,. \end{aligned}$$

We illustrate the curves in Figure 4.9. With these definitions, we can show the following.

Theorem 4.13. *The problems* Upper Bound Fréchet in 1D *and* Upper Bound Discrete Fréchet in 1D *are NP-hard in the indecisive model.*

Proof. First of all, notice that in our construction the synchronisation points at the start of the clauses gadgets must be matched to the synchronisation points at the start of the variable section and the abs gadgets in the optimal matching, as hinted at in the proofs of the previous lemmas. Furthermore, note that any number of abs^2 at the

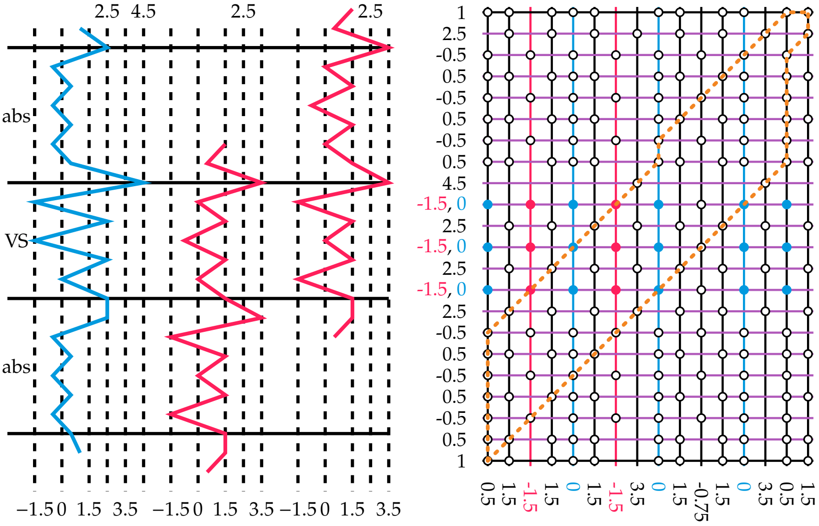


Figure 4.9. Left: The realisation of $\mathcal{U}$ for assignment x_1 = True, x_2 = True, x_3 = False and curve $\mathcal{V}$ for the formula $C = (x_1 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3)$. Note that C = True with this assignment, and that both feasible alignments give (discrete) Fréchet distance of 1.5. Right: The corresponding free space. White dots are accessible, spots without a dot are never accessible. Blue (red) dots are only accessible if the corresponding variable is set to True (False). Yellow dashed paths indicate potential paths through the free space; the goal is to determine if the variables can be set such that all potential paths are blocked.

start can be matched to the point at 1, and any number of abs^2 can be matched to the point at 1 at the end. Putting these observations together with Lemmas 4.10 to 4.12, it is easy to see the following. Choose some assignment a and consider the corresponding realisation $\pi \Subset \mathcal{U}$. Suppose that $C[a]$ = False; this means that there is at least one i for which $C_i[a]$ = False. Our construction allows us to consider the alignment where we match CG$_i$ to VS, and the rest of clauses to one of abs; the remaining abs are matched to the abs^2, and the remaining abs^2 are matched to 1. In this matching, the (discrete) Fréchet distance between the curves is 1, which is optimal, and the formula is not satisfied. Now suppose that $C[a]$ = True; this means that for all i, $C_i[a]$ = True. So, no matter which CG$_i$ we choose to align with VS, we get the distance of 1.5; therefore, the formula is satisfied, and the optimal distance is 1.5.

Recall that the upper bound distance takes the maximum distance over all realisations. Therefore, if the upper bound distance is 1, then all the realisations yield the distance 1, and so all assignments a yield $C[a] = \texttt{False}$, and the formula is not satisfiable. On the other hand, if the upper bound distance is 1.5, then there is some realisation that yields this distance, and it corresponds to an assignment a with $C[a] = \texttt{True}$, so the formula is satisfiable. Thus, our construction with the threshold $\delta = 1$ solves CNF-SAT. Curve $\mathcal{U}$ has length $2 + 2 \cdot (n - 1) \cdot (1 + 2m) + 1 + 2m = 2n + 4mn - 2m + 1$; curve $\mathcal{V}$ has length $4 \cdot (n - 1) + n \cdot (1 + 2m) = 5n + 2mn - 4$. Clearly, the construction takes polynomial time. Therefore, the problem both for discrete and continuous Fréchet distance is NP-hard. ◻

We can easily extend this result to the imprecise curves. We replace the indecisive points at $\{-1.5, 0\}$ with intervals $[-1.5, 0]$. The following observation is key.

Observation 4.14. *If an upper bound solution can be found as a certificate in the construction with the indecisive points, it can also be found in the imprecise construction.*

Furthermore, note that no realisation can yield a distance above 1.5 with an optimal matching. Thus, if the formula is satisfiable, the upper bound distance is still 1.5, and this distance cannot be obtained otherwise. We conclude that the problem is NP-hard.

Theorem 4.15. *The problems Upper Bound Fréchet in 1D and Upper Bound Discrete Fréchet in 1D are NP-hard in the imprecise model.*

4.4 Weak Fréchet Distance

In this section, we investigate the lower bound weak Fréchet distance for uncertain curves. In general, since weak matchings can revisit parts of the curve, the dynamic program for the regular Fréchet distance cannot easily be adapted, as it relies on the fact that only the realisation of the last few vertices is tracked. In particular, when computing the (lower bound) weak Fréchet distance for uncertain curves, one cannot simply forget the realisations of previously visited vertices, as the matching might revisit them. Surprisingly, we can show that for the continuous weak Fréchet distance between uncertain one-dimensional curves, we can still obtain a polynomial-time dynamic program, as shown in Section 4.4.1. One may expect that the discrete weak Fréchet distance for uncertain curves in 1D is also solvable in polynomial time; however, in Section 4.4.2, we show that this problem is NP-hard. We also show that computing the continuous weak Fréchet distance is NP-hard for uncertain curves in 2D.

4.4.1 Algorithm for Continuous Setting

We first introduce some definitions. Consider two polygonal one-dimensional curves $\pi\colon [1, m] \to \mathbb{R}$ and $\sigma\colon [1, n] \to \mathbb{R}$ with vertices at the integer parameters. Let π^{-1} denote the reversal of a polygonal curve π. Denote by $\pi|_{[a,b]}$ the restriction of π to the domain $[a, b]$. For integer values of a and b, note that $\pi|_{[a,b]} \equiv \pi[a : b]$. Finally, define the *image* of a curve as the set of points in $\mathbb{R}$ that belong to the curve, $\mathrm{Im}(\pi) \equiv \{\pi(x) \mid x \in [1, m]\}$ for $\pi\colon [1, m] \to \mathbb{R}$. For any polygonal curve π, define the *growing curve* $\vec{\pi}$ of π as the sequence of local minima and maxima of the sequence $\langle \pi(i) \mid \pi(i) \notin \mathrm{Im}(\pi|_{[1,i)})\rangle_{i=1}^{m}$. Thus, the vertices of a growing curve alternate between local minima and maxima, the subsequence of local maxima is strictly increasing, and the subsequence of local minima is strictly decreasing.

It has been shown that for precise one-dimensional curves, the weak Fréchet distance can be computed in linear time [60]. For uncertain curves, it is unclear how to use that linear-time algorithm; however, we can apply some of the underlying ideas. A *relaxed matching* between π and σ is defined by reparametrisations $\alpha\colon [0, 1] \to [1, m]$ and $\beta\colon [0, 1] \to [1, n]$ with $\alpha(0) = 1$, $\alpha(1) = x \in [m - 1, m]$ and $\beta(0) = 1$, $\beta(1) = y \in [n - 1, n]$. Observe that the final points of reparametrisations have to be on the last segments of the curves, but not necessarily at the endpoints of those segments. Moreover, define a relaxed matching (α, β) to be *cell-monotone* if for all $t \leq t'$, we have $\min(\lfloor\alpha(t)\rfloor, m - 1) \leq \alpha(t')$ and $\min(\lfloor\beta(t)\rfloor, n - 1) \leq \beta(t')$. In other words, once we pass by a vertex to the next segment on a curve, we do not allow going back to the previous segment; backtracking within a segment is allowed. Let $rm(\pi, \sigma)$ be the minimum matching cost over all cell-monotone relaxed matchings:

$$rm(\pi, \sigma) = \inf_{\text{cell-monotone relaxed matching } \mu} \operatorname{cost}_{\mu}(\pi, \sigma)\,.$$

It has been shown for precise curves [60] that

$$d_{\mathrm{wF}}(\pi, \sigma) = \max\bigl(rm(\vec{\pi}, \vec{\sigma}), rm(\overrightarrow{\pi^{-1}}, \overrightarrow{\sigma^{-1}})\bigr)\,.$$

Let $rm(\pi, \sigma)[i, j] \equiv rm\big(\pi[1:i], \sigma[1:j]\big)$. Then the value of $rm(\pi, \sigma)$ can be computed in quadratic time as $rm(\pi, \sigma)[m, n]$ using the following dynamic program:

$$\begin{aligned} rm(\pi,\sigma)[0,\cdot] &= \infty\,, \\ rm(\pi,\sigma)[\cdot,0] &= \infty\,, \\ rm(\pi,\sigma)[1,1] &= |\pi(1) - \sigma(1)|\,, \end{aligned}$$

and for $i > 0$ or $j > 0$,

$$rm(\pi,\sigma)[i+1,j+1] = \min \begin{cases} \max\Big(rm(\pi,\sigma)[i,j+1], \\ \qquad d\big(\pi(i), \mathrm{Im}(\sigma[j:j+1])\big)\Big)\,, \\ \max\Big(rm(\pi,\sigma)[i+1,j], \\ \qquad d\big(\sigma(j), \mathrm{Im}(\pi[i:i+1])\big)\Big)\,. \end{cases}$$

If π is a growing curve, we have $\mathrm{Im}(\pi[i:i+1]) = \mathrm{Im}(\pi[1:i+1])$, so the following dynamic program is equivalent if π and σ are growing curves:

$$\begin{aligned} r(\pi,\sigma)[0,\cdot] &= \infty\,, \\ r(\pi,\sigma)[\cdot,0] &= \infty\,, \\ r(\pi,\sigma)[1,1] &= |\pi(1) - \sigma(1)|\,, \end{aligned}$$

and for $i > 0$ or $j > 0$,

$$r(\pi,\sigma)[i+1,j+1] = \min \begin{cases} \max\Big(r(\pi,\sigma)[i,j+1], \\ \qquad d\big(\pi(i), \mathrm{Im}(\sigma[1:j+1])\big)\Big)\,, \\ \max\Big(r(\pi,\sigma)[i+1,j], \\ \qquad d\big(\sigma(j), \mathrm{Im}(\pi[1:i+1])\big)\Big)\,. \end{cases}$$

Let $r(\pi, \sigma) := r(\pi, \sigma)[m, n]$ when executing the dynamic program above for curves $\pi\colon [1, m] \to \mathbb{R}$ and $\sigma\colon [1, n] \to \mathbb{R}$. We have $rm(\vec{\pi}, \vec{\sigma}) = r(\vec{\pi}, \vec{\sigma})$. Moreover, observe that the final result of computing r is the same whether we apply it to the original or the growing curves. In other words, $r(\pi, \sigma) = r(\vec{\pi}, \vec{\sigma})$, so

$$\begin{aligned} d_{\mathrm{wF}}(\pi, \sigma) &= \max\bigl(rm(\vec{\pi}, \vec{\sigma}), rm(\overrightarrow{\pi^{-1}}, \overrightarrow{\sigma^{-1}})\bigr) \\ &= \max\bigl(r(\vec{\pi}, \vec{\sigma}), r(\overrightarrow{\pi^{-1}}, \overrightarrow{\sigma^{-1}})\bigr) \\ &= \max\bigl(r(\pi, \sigma), r(\pi^{-1}, \sigma^{-1})\bigr)\,. \end{aligned}$$

With regard to computing the lower bound weak Fréchet distance over realisations of uncertain curves, this roughly means that we only need to keep track of the image of the prefix (and the suffix) of π and σ. To formalise this, we split up the computation over the prefix and the suffix. Let $i_{\min}, i_{\max} \in [m]$, $j_{\min}, j_{\max} \in [n]$, $[x_{\min}, x_{\max}] \subseteq \mathbb{R}$, $[y_{\min}, y_{\max}] \subseteq \mathbb{R}$. Abbreviate the pairs $I := (i_{\min}, i_{\max})$, $J := (j_{\min}, j_{\max})$ and the intervals $X := [x_{\min}, x_{\max}]$, $Y := [y_{\min}, y_{\max}]$, and call a realisation π of an uncertain curve *I-respecting* if $\pi(i_{\min})$ is a global minimum of π and $\pi(i_{\max})$ is a global maximum of π. Say that π is *(I, X)-respecting* if additionally $\pi(i_{\min}) = x_{\min}$ and $\pi(i_{\max}) = x_{\max}$. Denote some I- and (I, X)-respecting realisations of an uncertain curve $\mathcal{U}$ by $\pi' \Subset \mathcal{U}_I$ and $\pi'' \Subset \mathcal{U}_I^X$, respectively. Consider the lower bound weak Fréchet distance between (I, X)- and (J, Y)-respecting realisations $\pi \Subset \mathcal{U}_I^X$ and $\sigma \Subset \mathcal{V}_J^Y$:

$$\begin{aligned} d_{\mathrm{wF}}^{\min}(\mathcal{U}_I^X, \mathcal{V}_J^Y) &\equiv \min_{\pi \Subset \mathcal{U}_I^X, \sigma \Subset \mathcal{V}_J^Y} d_{\mathrm{wF}}(\pi, \sigma) \\ &= \min_{\pi \Subset \mathcal{U}_I^X, \sigma \Subset \mathcal{V}_J^Y} \max\bigl(r(\pi, \sigma), r(\pi^{-1}, \sigma^{-1})\bigr)\,. \end{aligned}$$

Lemma 4.16. *Among (I, X)- and (J, Y)-respecting realisations, the prefix and the suffix are independent:*

$$d_{\mathrm{wF}}^{\min}(\mathcal{U}_I^X, \mathcal{V}_J^Y) = \max \begin{cases} \min_{\pi \Subset \mathcal{U}_I^X, \sigma \Subset \mathcal{V}_J^Y} r(\pi, \sigma)\,, \\ \min_{\pi' \Subset \mathcal{U}_I^X, \sigma' \Subset \mathcal{V}_J^Y} r(\pi'^{-1}, \sigma'^{-1})\,. \end{cases}$$

Proof. If we take $\pi = \pi'$ and $\sigma = \sigma'$, the right-hand side becomes a lower bound on $d_{\mathrm{wF}}^{\min}(\mathcal{U}_I^X, \mathcal{V}_J^Y)$. To show that it is also an upper bound, consider (I, X)-respecting realisations π and π', and define π_c as the prefix of π up to $i_{\min}$ concatenated with the suffix of π' starting from $i_{\min}$. Then π_c is an (I, X)-respecting realisation of $\mathcal{U}$. Moreover, the growing curves $\vec{\pi}$ and $\overrightarrow{\pi_c}$ are the same (this is obvious if $i_{\min} > i_{\max}$, and follows from the fact that the value of the $i_{\max}$th vertex is $x_{\max}$ otherwise).

Symmetrically, $\overrightarrow{\pi'^{-1}} = \overrightarrow{\pi_c^{-1}}$. We can similarly define a (J, Y)-respecting realisation σ_c of $\mathcal{V}$ based on some σ and σ'. Since $\vec{\pi} = \overrightarrow{\pi_c}$ and $\vec{\sigma} = \overrightarrow{\sigma_c}$, we have $r(\pi, \sigma) = r(\pi_c, \sigma_c)$, and symmetrically, $r(\pi'^{-1}, \sigma'^{-1}) = r(\pi_c^{-1}, \sigma_c^{-1})$. We can therefore use $\pi_c \Subset \mathcal{U}_I^X$ and $\sigma_c \Subset \mathcal{V}_J^Y$ in the definition of $d_{\mathrm{wF}}(\mathcal{U}_I^X, \mathcal{V}_J^Y)$ to obtain the desired upper bound. □

The remainder of this section is guided by observations based on Lemma 4.16.

1. Computing $\min_{\pi \Subset \mathcal{U}_I^X, \sigma \Subset \mathcal{V}_J^Y} r(\pi, \sigma)$ lets us compute $d_{\mathrm{wF}}^{\min}(\mathcal{U}_I^X, \mathcal{V}_J^Y)$.
2. To compute $d_{\mathrm{wF}}^{\min}(\mathcal{U}_I, \mathcal{V}_J)$, we must find an optimal pair of images X and Y for π and σ.
3. We can compute $d_{\mathrm{wF}}^{\min}(\mathcal{U}, \mathcal{V})$ by computing $d_{\mathrm{wF}}^{\min}(\mathcal{U}_I, \mathcal{V}_J)$ for all $O(m^2 n^2)$ values for (I, J).

Instead of computing $\min_{\pi \Subset \mathcal{U}_I^X, \sigma \Subset \mathcal{V}_J^Y} r(\pi, \sigma)$ for a specific value of (X, Y), we compute the function $(X, Y) \mapsto \min_{\pi \Subset \mathcal{U}_I^X, \sigma \Subset \mathcal{V}_J^Y} r(\pi, \sigma)$ using a dynamic program that effectively simulates the dynamic program $r(\pi, \sigma)$ for all I- and J-respecting realisations simultaneously. So let

$$\begin{aligned}
R_{I,J}[i, j](x, y, X, Y) &:= \inf_{\substack{\pi \Subset \mathcal{U}_I, \operatorname{Im}(\pi[1:i]) = X, \pi(i) = x \\ \sigma \Subset \mathcal{V}_J, \operatorname{Im}(\sigma[1:j]) = Y, \sigma(j) = y}} r(\pi, \sigma)[i, j]\,, \quad \text{then} \\
R_{I,J}[m, n](x, y, X, Y) &= \inf_{\substack{\pi \Subset \mathcal{U}_I^X, \pi(m) = x \\ \sigma \Subset \mathcal{V}_J^Y, \sigma(n) = y}} r(\pi, \sigma)\,.
\end{aligned}$$

We derive

$$\begin{aligned}
R_{I,J}[0, \cdot](x, y, X, Y) &= R_{I,J}[\cdot, 0](x, y, X, Y) = \infty, \\
R_{I,J}[1, 1](x, y, X, Y) &= \inf_{\substack{\pi \Subset \mathcal{U}_I, \{x\} = X, \pi(1) = x \\ \sigma \Subset \mathcal{V}_J, \{y\} = Y, \sigma(1) = y}} |\pi(1) - \sigma(1)|,
\end{aligned}$$

and for $(i, j) \neq (1, 1)$,

$$
\begin{aligned}
&R_{I,J}[i, j](x, y, X, Y) \\
&\quad= \inf_{\substack{\pi \in \mathcal{U}_I, \mathrm{Im}(\pi[1:i]) = X, \pi(i) = x \\ \sigma \in \mathcal{V}_J, \mathrm{Im}(\sigma[1:j]) = Y, \sigma(j) = y}} \min \begin{cases} \max\{r(\pi, \sigma)[i-1, j], d(\pi(i-1), Y)\}, \\ \max\{r(\pi, \sigma)[i, j-1], d(\sigma(j-1), X)\} \end{cases} \\
&\quad= \min \begin{cases} \inf_{\substack{\pi \in \mathcal{U}_I, \mathrm{Im}(\pi[1:i]) = X, \pi(i) = x \\ \sigma \in \mathcal{V}_J, \mathrm{Im}(\sigma[1:j]) = Y, \sigma(j) = y \\ \pi(i-1) = x'}} \max\{r(\pi, \sigma)[i-1, j], d(x', Y)\}, \\ \inf_{\substack{\pi \in \mathcal{U}_I, \mathrm{Im}(\pi[1:i]) = X, \pi(i) = x \\ \sigma \in \mathcal{V}_J, \mathrm{Im}(\sigma[1:j]) = Y, \sigma(j) = y \\ \sigma(j-1) = y'}} \max\{r(\pi, \sigma)[i, j-1], d(y', X)\}, \end{cases} \\
&\quad= \min \begin{cases} \inf_{\substack{\pi \in \mathcal{U}_I, \mathrm{Im}(\pi[1:i]) = X, \\ \mathrm{Im}(\pi[1:i-1]) = X', \\ \pi(i) = x, \pi(i-1) = x'}} \max\{R_{I,J}[i-1, j](x', y, X', Y), d(x', Y)\}, \\ \inf_{\substack{\sigma \in \mathcal{V}_J, \mathrm{Im}(\sigma[1:j]) = Y, \\ \mathrm{Im}(\sigma[1:j-1]) = Y', \\ \sigma(j) = y, \sigma(j-1) = y'}} \max\{R_{I,J}[i, j-1](x, y', X, Y'), d(y', X)\}, \end{cases}
\end{aligned}
$$

where, crucially, the conditions on x', y', X', and Y' can be checked purely in terms of $\mathcal{U}_I$ and $\mathcal{V}_J$, so the recurrence does not depend on any particular π or σ. This yields a dynamic program that constructs the function $R_{I,J}[i, j]$ based on the functions $R_{I,J}[i-1, j]$ and $R_{I,J}[i, j-1]$.

The recurrence has parameters I, J, i, j, x, y, X, and Y. The first four are easy to handle, since $i \in [m]$, $j \in [n]$, $I \in [m]^2$, and $J \in [n]^2$. The other parameters are continuous. X can be represented by $x_{\min}$, $x_{\max}$, and Y by $y_{\min}$, $y_{\max}$. To prove that we can solve the recurrence in polynomial time, it is sufficient to prove that we can restrict the computation to a polynomial number of different $x_{\min}$, $x_{\max}$, $y_{\min}$, $y_{\max}$, x, and y.

We assume that each of the u_i and v_j is given as a set of intervals. This includes the cases of uncertain curves with imprecise vertices (where each of these is just one interval) and with indecisive vertices (where each interval is just a point; but in this case, we get by definition only a polynomial number of different values for the parameters).

Consider the realisations $\pi = \langle p_1, \dots, p_m \rangle$ and $\sigma = \langle q_1, \dots, q_n \rangle$ of the curves that attain $d_{\mathrm{wF}}^{\min}(\mathcal{U}, \mathcal{V}) =: \delta$. In these realisations, we need to have a sequence of vertices $r_1 \leq r_2 \leq \cdots \leq r_\ell$ with the r_k alternately from the set of p_i and the set of q_j such that r_1 is at a right interval endpoint, r_ℓ is at a left interval endpoint, and $r_{k+1} - r_k = \delta$. Since $1 \leq \ell \leq m + n$, this implies that there are only $O(N^2 \cdot (m + n))$ candidates for δ, where

N is the total number of interval endpoints. We can compute these candidates in time $O(N^2 \cdot (m+n))$.

Now assume that we have chosen π and σ such that none of the p_i or q_j can be increased (i.e. moved to the right) without increasing the weak Fréchet distance. Then for every p_i (and likewise q_j), there is a sequence $r_1 \leq r_2 \leq \cdots \leq r_\ell = p_i$, where r_1 is the endpoint of an interval and $r_{k+1} - r_k = \delta$. There are $O(N)$ possibilities for r_1, $O(m+n)$ possibilities for ℓ, and $O(N^2 \cdot (m+n))$ possibilities for δ, thus the total number of positions to consider for p_i is polynomial.

Theorem 4.17. *The lower bound continuous weak Fréchet distance between uncertain one-dimensional curves can be computed in polynomial time.*

4.4.2 Hardness of Discrete Setting

In this section, we prove that minimising the discrete weak Fréchet distance is NP-hard, already in one-dimensional space. We show this both for indecisive and imprecise points. In the constructions in this section, the lower bound Fréchet distance is never smaller than 1. The goal is to determine whether it is equal to 1 or greater than 1.

Indecisive points. We reduce from 3SAT. Consider an instance with n variables and m clauses. We assign each variable a unique *height:* variable x_i gets assigned height $10i + 5$. We use slightly larger heights ($10i + 6$ and $10i + 7$) to interact with the positive state of the variable, and slightly smaller heights to interact with the negative state.

We construct two uncertain curves, one representing the variables and one representing the clauses. The first curve, $\mathcal{U}$, consists of $n + 2$ vertices. The first and last vertices are precise points at height 0. The remaining vertices are uncertain points, with two possible heights each:

$$\mathcal{U} = \langle 0, \{14, 16\}, \{24, 26\}, \ldots, \{10n + 4, 10n + 6\}, 0 \rangle .$$

The second curve, $\mathcal{V}$, consists of $nm + n + m + 2$ vertices. For a clause $c_j = \ell_a \vee \ell_b \vee \ell_c$, let C_j be the set $\{10a + 3/7, 10b + 3/7, 10c + 3/7\}$, where for each literal we choose $+7$ if $\ell_i = x_i$ or $+3$ if $\ell_i = \neg x_i$. Let S be the set $S = \{15, 25, \ldots, 10n + 5\}$ of 'neutral' variable heights. Define $\mathcal{V}$ as the curve that starts and ends at 0, has a vertex for each C_j, and has

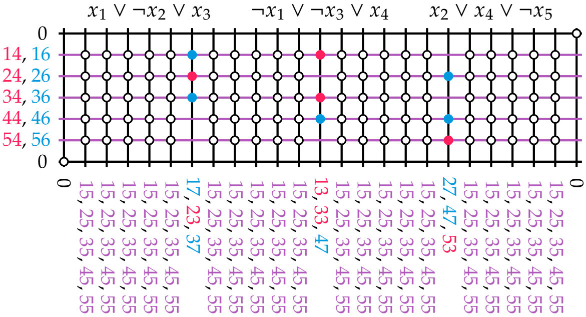

$x_1 \vee \neg x_2 \vee x_3$
$\neg x_1 \vee \neg x_3 \vee x_4$
$x_2 \vee x_4 \vee \neg x_5$
0
14, 16
24, 26
34, 36
44, 46
54, 56
0
0
15, 25, 35, 45, 55
17, 23, 37
13, 33, 47
27, 47, 53
0

Figure 4.10. An example with five variables and three clauses. White dots are always accessible, no matter the state of the variables (however, note that only one white dot per column can be used). Red / blue dots are accessible only if the corresponding variable is set to False / True. Spots without a dot are never accessible.

sufficiently many copies of S between them:

$$\mathcal{V} = \langle 0, S, \ldots, S, C_1, S, \ldots, S, C_2, S, \ldots, S, \ldots\ldots, C_m, 0 \rangle .$$

Consider the free-space diagram, with a 'spot' (i, j) corresponding to a pair of vertices u_i and v_j. The discrete weak Fréchet distance is equal to 1 if and only if there is an assignment to each uncertain vertex such that the there is a path from the bottom left to the top right of the diagram that uses only accessible spots, where a spot is accessible if the assigned heights of the corresponding row and column are within distance 1. Figure 4.10 shows an example.

We can only cross a column corresponding to clause c_j if at least one of the corresponding literals is set to true. The remaining columns can always be crossed at any row. Note that the repetition is necessary: although all spots are in principle reachable, only one spot in each column can be reachable at the same time. If we have at least n columns between each pair of clauses, this will always be possible.

Theorem 4.18. *Given two uncertain curves $\mathcal{U}$ and $\mathcal{V}$, each given by a sequence of values and sets of values in $\mathbb{R}$, the problem of choosing a realisation of $\mathcal{U}$ and $\mathcal{V}$ such that the weak discrete Fréchet distance between $\mathcal{U}$ and $\mathcal{V}$ is minimised is NP-hard.*

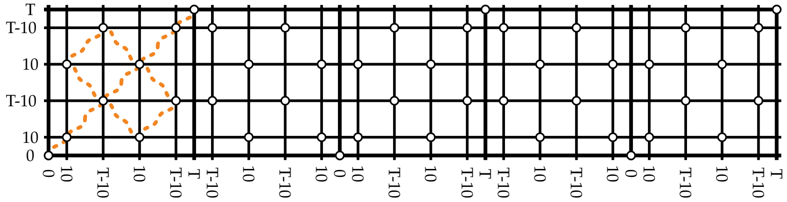


Figure 4.11. The global frame. White dots are accessible, spots without a dot are never accessible. Within each block, there are three potential paths between its two accessible corners.

Imprecise points. The construction above relies on the ability to select arbitrary sets of values as uncertainty regions. We now show that this is not required. We strengthen the proof in two ways: we restrict the uncertainty regions to intervals and we use uncertainty in only one curve.

The main idea of the adaptation is to encode clauses not by a single uncertain vertex, but by sets of globally distinct paths through the free-space diagram. To facilitate this, we need a global *frame* to guide the possible solution paths, and we need more copies of the variable vertices (though only one copy will be uncertain) to facilitate the paths.

Let $T = 10(n+2)$. We build a frame for the construction using four unique heights: $0, 10, T-10$, and T.[2] Let $S = \langle 0, 10, ?, T-10, ?, 10, ?, T-10, T \rangle$ be a partial sequence—the question marks indicate the gaps where we will insert other vertices later. Globally, the curves have the structure $\mathcal{U} = S$ and $\mathcal{V} = S \sqcup S^{-1} \sqcup S \sqcup S^{-1} \sqcup \ldots \sqcup S$: one copy or reversed copy of S for each clause (if the number of clauses is even, simply add a trivial clause). In the free-space diagram, this creates a frame that every path needs to adhere to. The frame consists of one block per clause, and inside each block, there are three potential paths from the bottom left to top right corner (or from the top left to the bottom right corner for reversed blocks). See Figure 4.11.

Next, we fill in the gaps. Let $\mathcal{U} = \langle 0 \rangle \sqcup \mathcal{U}_1 \sqcup \mathcal{U}_2^{-1} \sqcup \mathcal{U}_1 \sqcup \langle T \rangle$, where

$$\mathcal{U}_1 = \langle 10, 14, 16, 24, 26, \ldots, 10n+4, 10n+6, T-10 \rangle \,,$$
$$\mathcal{U}_2 = \langle 10, [14, 16], [24, 26], \ldots, [10n+4, 10n+6], T-10 \rangle \,.$$

---

[2] The actual values are, in fact, irrelevant for the construction—they simply need to be unique numbers sufficiently removed from the values that we use for encoding the variables.

Let $\mathcal{V} = \bigsqcup_{1 \leq j \leq m} C_j^{(-1)^{j-1}}$ be the concatenation of clause sequences, where every even clause sequence is reversed. For a clause $c_j = \ell_a \vee \ell_b \vee \ell_c$, the sequence C_j is of the form

$$C_j = \langle 0, 10 \rangle \sqcup L_a \sqcup \langle T - 10 \rangle \sqcup L_b^{-1} \sqcup \langle 10 \rangle \sqcup L_c \sqcup \langle T - 10, T \rangle\,,$$

where the literal sequences L_i corresponding to $\ell_i = x_i$ (positive literals), $\ell_i = \neg x_i$ (negative literals) are

$$L_i = \langle 15, \dots, 10(i-1)+5, 10i+5, 10i+7, 10(i+1)+5, \dots, 10n+5 \rangle\,,$$
$$L_i = \langle 15, \dots, 10(i-1)+5, 10i+3, 10i+5, 10(i+1)+5, \dots, 10n+5 \rangle\,,$$

respectively. See Figure 4.12 for an example of the resulting free-space diagram.

The construction relies on the following observation.

Observation 4.19. *L_i can always be matched to $\mathcal{U}_1$. L_i can be matched to $\mathcal{U}_2$ if and only if $\ell_i = x_i$ and x_i is set to* `True`*, or $\ell_i = \neg x_i$ and x_i is set to* `False`*.*

Theorem 4.20. *Given an uncertain curve $\mathcal{U}$, described by a sequence of values and intervals in $\mathbb{R}$, and a precise curve $\mathcal{V}$, described by a sequence of values in $\mathbb{R}$, it is NP-hard to choose a realisation of $\mathcal{U}$ such that the weak discrete Fréchet distance between $\mathcal{U}$ and $\mathcal{V}$ is minimised.*

Continuous weak Fréchet distance in $\mathbb{R}^2$. Finally, we mention that the results in this section carry over to the continuous weak Fréchet distance in one dimension higher. We simply construct the same curves as described above on the x-axis, and intersperse each curve with the point at $(0, \infty)$.

Corollary 4.21. *Given an uncertain curve $\mathcal{U}$, described by a sequence of points and regions in $\mathbb{R}^2$, and a precise curve $\mathcal{V}$, described by a sequence of points in $\mathbb{R}^2$, it is NP-hard to choose a realisation of $\mathcal{U}$ such that the weak Fréchet distance between $\mathcal{U}$ and $\mathcal{V}$ is minimised.*

4.5 Conclusions

In this chapter, we have considered the standard extremal questions about the (discrete) (weak) Fréchet distance on uncertain curves in one dimension. We conclude that deciding if the upper bound is above a

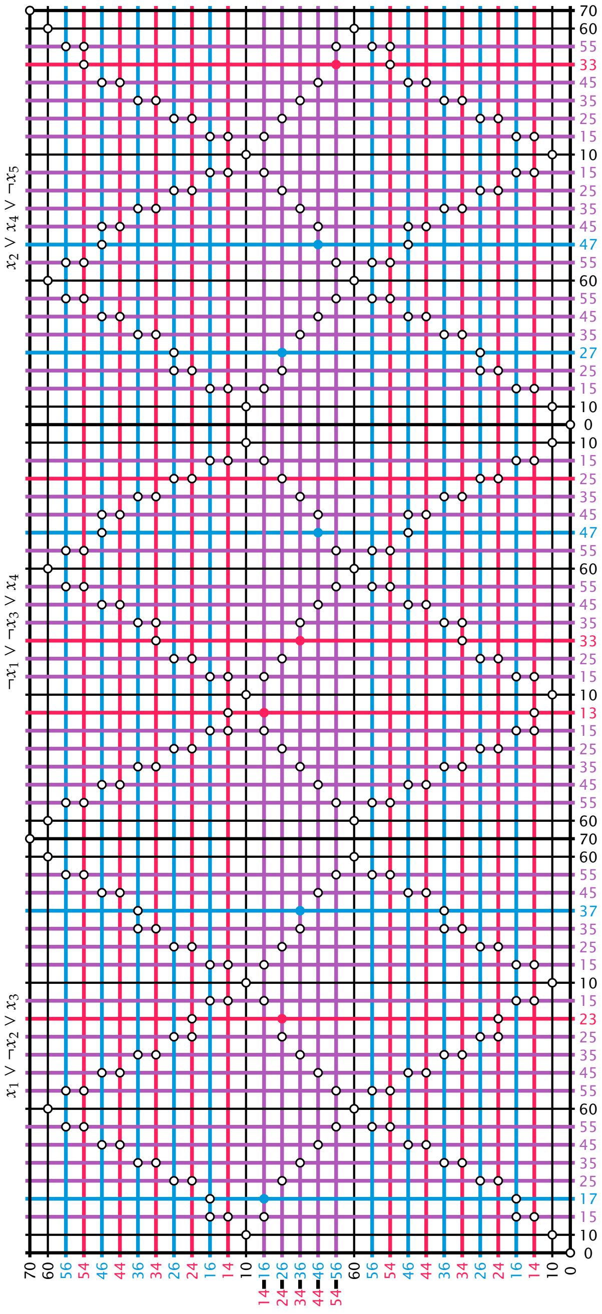


Figure 4.12. An example with five variables and three clauses. White dots are always accessible, no matter the state of the variables. Red / blue dots are accessible only if the corresponding variable is set to False / True. Spots without a dot are never accessible.

given threshold remains NP-hard for indecisive points, and is NP-hard for intervals. It appears hopeless to find a variant of the Fréchet distance where the upper bound problem is not NP-hard, although it is an interesting open problem to complete the study of the upper bound (discrete) weak Fréchet distance.

The lower bound problem, like in Chapter 3, turns out to be solvable in polynomial time in some settings and NP-hard in other settings. The dichotomy for the weak Fréchet distance, interestingly, is the opposite of that for the regular Fréchet distance: discretising the problem makes it harder in this case, not easier.

In future work, it would be interesting to complete filling in the gaps by studying the upper bound weak Fréchet distance, as well as various variants of the Hausdorff distance on uncertain point sets—the two distances share some structure, so it would be interesting to see exactly which variants turn out to be hard. Finally, for the lower bound Fréchet distance, it is possible that the problem is fixed-parameter tractable in the number of directions that the edges of the curve may take; investigating this further would be an interesting line of research, potentially making it more practical to work with the Fréchet distance under uncertainty.

CHAPTER 5

Uncertain Curve Simplification

In this chapter, we investigate the topic of curve simplification under uncertainty. There are many classical algorithms dealing with curve simplification with different distance metrics, discussed in Section 1.2; however, it is typically assumed that the locations of the points making up the curves are known precisely, which is not ideal when modelling noisy data. Curve simplification is frequently used as a first step to reduce the noise-to-signal ratio in the trajectory data before applying other algorithms or when storing large amounts of data. In both cases, modelling uncertainty could reduce the error introduced by simplifying imprecise measurements while maintaining a short, efficient representation of the data.

Among the approaches to simplifying polygonal curves, we would like to highlight the one by Imai and Iri [145]. The basic approach involves computing the *shortcut graph*, which captures all the possible simplifications of a curve, and then finding a shortest path through the graph in terms of the number of edges from the start node to the end node, thus finding the simplification with fewest edges. We adapt this approach in our work to simplification under uncertainty, which, to our knowledge, has not been studied before.

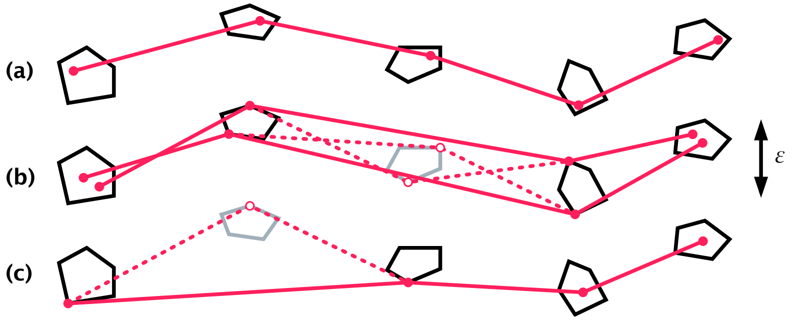


Figure 5.1. **(a)** An uncertain curve modelled with convex polygons and a potential realisation. **(b)** A valid simplification under the Hausdorff distance with the threshold ε: for every realisation, the subsequence is within Hausdorff distance ε from the full sequence. **(c)** An invalid simplification under the Hausdorff distance with the threshold ε: there is a realisation for which the subsequence is not within Hausdorff distance ε from the full sequence.

As in Chapters 3 and 4, we use indecisive and imprecise models. Recall that an uncertain curve is a sequence of uncertain points of the same kind, and a realisation of an uncertain curve is a precise polygonal curve obtained by taking one point from each uncertain point.

In this chapter, we solve the following problem.

Problem 5.1. Given an uncertain curve as a sequence of n uncertain points, find the shortest subsequence of the uncertain points of the curve such that for any realisation of the curve, the corresponding realisation of the subsequence is a valid simplification of that realisation.

We present a family of efficient algorithms for this problem under both the Hausdorff and the Fréchet distance, with the uncertain points modelled as indecisive points, as disks, as line segments, and as convex polygons, shown in Table 5.1. See also Figure 5.1.

Using the notation introduced in Chapter 2, we discuss the following problem: given an uncertain curve $\mathcal{U} = \langle U_1, \dots, U_n \rangle$ with $n \in \mathbb{N}$, $n \geq 3$, and $U_i \subset \mathbb{R}^2$ for all $i \in [n]$, and the threshold $\varepsilon \in \mathbb{R}^+$, find a minimal-length subsequence $\mathcal{U}' = \langle U_{i_1}, \dots, U_{i_\ell} \rangle$ of $\mathcal{U}$ with $\ell \leq n$, such that for any realisation $\pi \Subset \mathcal{U}$, the corresponding realisation $\pi' \Subset \mathcal{U}'$ forms an ε-simplification of π under some distance measure δ. We solve this problem both for the Hausdorff and the Fréchet distance for

Table 5.1. Running time of our approach in each setting. For indecisive points, k is the number of options per point. For convex polygons, k is the number of vertices.

	indecisive	disks	line segments	convex polygons
Hausdorff	$O(n^3k^3)$	$O(n^3)$	$O(n^3)$	$O(n^3k^3)$
Fréchet	$O(n^3k^3)$	$O(n^3)$	$O(n^3)$	$O(n^3k^3)$

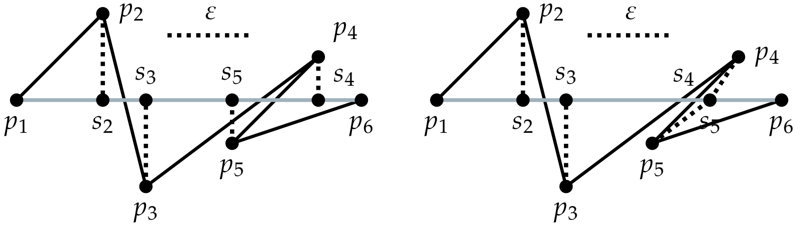


Figure 5.2. Left: Alignment for the Hausdorff distance. Right: Alignment for the Fréchet distance. In both cases, the alignment is described as the sequence $\langle s_1 := p_1, s_2, s_3, s_4, s_5, s_6 := p_6 \rangle$.

uncertainty modelled with indecisive points, line segments, disks, and convex polygons.

Preliminaries. Suppose we are given a threshold $\varepsilon \in \mathbb{R}^+$, a polygonal curve $\pi = \langle p_1, \dots, p_n \rangle$, and a curve built on the subsequence of vertices of π for some set $I = \{i_1, \dots, i_\ell\} \subseteq [n]$, i.e. $\sigma = \langle p_{i_1}, \dots, p_{i_\ell} \rangle$ with $i_j < i_{j+1}$ for all $j \in [\ell - 1]$ and $\ell \leq n$. We call σ an *ε-simplification* of π if for each segment $\langle p_{i_j}, p_{i_{j+1}} \rangle$, we have $\delta(\langle p_{i_j}, p_{i_{j+1}} \rangle, \pi[i_j : i_{j+1}]) \leq \varepsilon$, where δ denotes some distance measure, e.g. the Hausdorff or the Fréchet distance. The *Hausdorff distance* between two sets $P, Q \subset \mathbb{R}^2$ is

$$d_{\mathrm{H}}(P, Q) \overset{\text{def}}{=} \max\Bigl\{\sup_{p \in P} \inf_{q \in Q} \|p - q\|, \sup_{q \in Q} \inf_{p \in P} \|p - q\|\Bigr\}\,.$$

For two polygonal curves π and σ in $\mathbb{R}^2$, since π and σ are closed and bounded, we get

$$d_{\mathrm{H}}(\pi, \sigma) = \max\Bigl\{\max_{p \in \pi} \min_{q \in \sigma} \|p - q\|, \max_{q \in \sigma} \min_{p \in \pi} \|p - q\|\Bigr\}\,.$$

Recall the definition of the Fréchet distance using reparametrisations. In this chapter, we often consider the Fréchet distance between a curve

$\pi = \langle p_1, \ldots, p_n \rangle$ and a line segment p_1p_n, for some $n \in \mathbb{N}$, $n \geq 3$. In this setting, the alignment can be described in a more intuitive way; see also Figure 5.2. It can be described as a sequence of locations on the line segment to which the vertices of the curves are matched, $\langle s_2, \ldots, s_{n-1} \rangle$, where $s_i \in [1, 2]$ for all $i \in \{2, \ldots, n-1\}$ and $s_i \leq s_{i+1}$ for all $i \in \{2, \ldots, n-2\}$. To see that, assign $s_1 := 1$ and $s_n := 2$ and construct a helper reparametrisation $\phi : [0, 1] \to [1, n]$, defined as $\phi(t) = (n-1) \cdot t + 1$ for any $t \in [0, 1]$. Construct another reparametrisation $\psi : [1, n] \to [1, 2]$, defined as

$$\psi(t) = \begin{cases} s_{\lfloor t \rfloor} \cdot (1 - t + \lfloor t \rfloor) + s_{\lfloor t \rfloor + 1} \cdot (t - \lfloor t \rfloor) & \text{if } t \in [1, n), \\ s_n & \text{if } t = n. \end{cases}$$

Note that ϕ and $\psi \circ \phi$ satisfy the definition of reparametrisations for π and p_1p_n, respectively.

We also define an *alignment* between a curve and a line segment for the Hausdorff distance (see Figure 5.2). It represents the map from the curve to the line segment, where each point on the curve is mapped to the closest point on the line segment. It is given by a sequence $\langle s_1, \ldots, s_n \rangle$, where $s_i \in [1, 2]$ for all $i \in [n]$, such that $p_1p_n(s_i) = \arg\min_{p' \in p_1p_n} \|p' - p_i\|$. In other words, $p_1p_n(s_i)$ is the closest point to p_i for all $i \in [n]$; as we show in Section 5.2.1, the Hausdorff distance is realised as the distance between p_i and $p_1p_n(s_i)$ for some $i \in [n]$. Therefore, establishing such an alignment and checking that $\|p_1p_n(s_i) - p_i\| \leq \varepsilon$ for all $i \in [n]$ allows us to check that $d_{\mathrm{H}}(\pi, p_1p_n) \leq \varepsilon$ for some $\varepsilon \in \mathbb{R}^+$.

5.1 Overview of the Approach

In this section, we present the short description of our approach in different settings. We work out the details and show correctness in Sections 5.2 to 5.4.

On the highest level, we use the *shortcut graph*. Each uncertain point of an uncertain curve $\mathcal{U} = \langle U_1, \ldots, U_n \rangle$ corresponds to a vertex. An edge connects two vertices i and j if and only if the distance between any realisation of $\mathcal{U}[i : j]$ and the corresponding line segment from U_i to U_j is below the threshold. The path with the least edges from vertex 1 to vertex n then corresponds to the simplification using least uncertain points. So, we construct the shortcut graph and find the shortest path between two vertices. The key idea is that we find shortcuts that are valid for *all* realisations, so any sequence of shortcuts can be chosen. We discuss this in Section 5.4.

In order to construct the shortcut graph, we need to check whether an edge should be added to the graph, i.e. whether a shortcut is *valid*. The approach is different for the Hausdorff and the Fréchet distance and for each uncertainty model. For the first and the last uncertain point of the shortcut, we state in Section 5.3 that there are several critical pairs of realisations that need to be tested explicitly, and then for any other pair of realisations, we know that the distance is also below the threshold. Testing each pair corresponds to finding the distance between a precise line segment and any realisation of an uncertain curve; we show the simple procedures to do this in Section 5.2.

5.2 Shortcut Testing: Intermediate Points

In this section, we discuss testing a single shortcut where we fix the realisations of the first and the last uncertain point. We start by showing some basic facts about the Hausdorff and the Fréchet distance in the precise setting, and then we use them to design simple algorithms for testing shortcuts in the uncertain settings. We answer the following problem.

Problem 5.2. Given an uncertain curve $\mathcal{U} = \langle U_1, \dots, U_n \rangle$ on $n \in \mathbb{N}$, $n \geq 3$ uncertain points in $\mathbb{R}^2$, as well as realisations $p_1 \in U_1$, $p_n \in U_n$, check if the largest Hausdorff or Fréchet distance between $\mathcal{U}$ and its one-segment simplification is below a threshold $\varepsilon \in \mathbb{R}^+$ for any realisation with the fixed start and end points, i.e. for $\delta := d_{\mathrm{H}}$ or $\delta := d_{\mathrm{F}}$, verify

$$\max_{\pi \Subset \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} \delta(\pi, p_1 p_n) \leq \varepsilon\,.$$

5.2.1 Hausdorff Distance

We start by showing some useful facts about the Hausdorff distance in the precise setting. We then solve Problem 5.2 for $\delta := d_{\mathrm{H}}$.

Lemma 5.3. *Given $n \in \mathbb{N}$ and a precise curve $\pi = \langle p_1, \dots, p_n \rangle$ with $p_i \in \mathbb{R}^2$ for all $i \in [n]$, we have that for any $q \in p_1 p_n$ with $q \prec p_n$, there is some*

$i \in [n-1]$ such that $s := \arg\min_{r \in p_1p_n} \|p_i - r\|$, $t := \arg\min_{r \in p_1p_n} \|p_{i+1} - r\|$, and $s \preccurlyeq q \prec t$.

Proof. Assume this is not the case and pick a point $q \in p_1p_n \setminus p_n$ that forms a counterexample. We now have, for all $i \in [n-1]$ and the definitions of s and t given above, that $s \preccurlyeq q \implies t \preccurlyeq q$. Clearly, for $i = 1$, we have $s \equiv p_1$ and so $s \preccurlyeq q$. By induction on i, we can conclude that for all $i \in [n]$, $\arg\min_{r \in p_1p_n} \|p_i - r\| \preccurlyeq q$. In particular, as $\arg\min_{r \in p_1p_n} \|p_n - r\| = p_n$, this means that $p_n \preccurlyeq q$. However, we picked $q \prec p_n$. This is a contradiction, so the lemma holds. □

Lemma 5.4. *Given four points $a, b, c, d \in \mathbb{R}^2$ forming segments ab and cd, the largest distance from one segment to the other is achieved at an endpoint:*

$$\max_{p \in ab} d(p, cd) = \max\left\{d(a, cd), d(b, cd)\right\}.$$

Proof. Trivially, $\max_{p \in ab} d(p, cd) \geq \max\{d(a, cd), d(b, cd)\}$, since $a, b \in ab$, so it remains to show that $\max_{p \in ab} d(p, cd) \leq \max\{d(a, cd), d(b, cd)\}$. Consider two sets $S_1 := \{p \mid \|p\| \leq \varepsilon\}$ and $S_2 := cd$, with $\varepsilon := \max\{d(a, cd), d(b, cd)\}$. Take their Minkowski sum:

$$\begin{aligned} S &:= \{p + q \mid p \in S_1, q \in S_2\} \\ &\overset{\text{def}}{=} \{p + q \mid \|p\| \leq \varepsilon, q \in cd\} \\ &= \{r \mid \|r - q\| \leq \varepsilon, q \in cd\} \\ &= \{r \mid \min_{q \in cd} \|r - q\| \leq \varepsilon\} \\ &\overset{\text{def}}{=} \{r \mid d(r, cd) \leq \varepsilon\}. \end{aligned}$$

Note that both sets are convex: S_1 is a disk and S_2 is a line segment. Then their Minkowski sum S is also convex. By definition of S and ε, we have $a, b \in S$. By definition of a convex set, we conclude that $ab \in S$, so $\max_{p \in ab} d(p, cd) \leq \max\{d(a, cd), d(b, cd)\}$, and the statement of the lemma holds. □

Lemma 5.5. *Given $n \in \mathbb{N}$, for any precise curve $\pi = \langle p_1, \ldots, p_n \rangle$ with $p_i \in \mathbb{R}^2$ for all $i \in [n]$, we have*

$$d_{\mathrm{H}}(\pi, p_1p_n) = \max_{i \in [n]} d(p_i, p_1p_n).$$

Proof. Recall the definition of Hausdorff distance in this setting:

$$d_{\mathrm{H}}(\pi, p_1p_n) = \max\left\{\max_{p\in\pi}\min_{q\in p_1p_n}\|p-q\|, \max_{q\in p_1p_n}\min_{p\in\pi}\|p-q\|\right\}.$$

We first show that $\max_{q\in p_1p_n}\min_{p\in\pi}\|p-q\| \le \max_{p\in\pi}\min_{q\in p_1p_n}\|p-q\| =: \varepsilon$. We do a case distinction on $q \in p_1p_n$ and show that for all q, we have $\min_{p\in\pi}\|p-q\| \le \varepsilon$.

- $q \equiv p_n$. Note $p_n \in \pi$, so $\min_{p\in\pi}\|p-q\| = 0 \le \varepsilon$.
- $q \prec p_n$. Using Lemma 5.3, we find $i \in [n-1]$ and the corresponding s and t such that $s \preccurlyeq q \prec t$. As $\max_{p\in\pi} d(p, p_1p_n) \overset{\text{def}}{=} \varepsilon$, we know $d(p_i, p_1p_n) = \|p_i - s\| \le \varepsilon$ and $d(p_{i+1}, p_1p_n) = \|p_{i+1} - t\| \le \varepsilon$. But then $d(s, p_ip_{i+1}) \le \|s - p_i\| \le \varepsilon$ and $d(t, p_ip_{i+1}) \le \|t - p_{i+1}\| \le \varepsilon$. By Lemma 5.4, we conclude that $\max_{r\in st} d(r, p_ip_{i+1}) \le \varepsilon$. As $s \preccurlyeq q \prec t$, we have $q \in st$, so $d(q, p_ip_{i+1}) \le \varepsilon$.

This covers all cases, so indeed for all $q \in p_1p_n$, $\min_{p\in\pi}\|p-q\| \le \varepsilon$, and hence we conclude $\max_{q\in p_1p_n}\min_{p\in\pi}\|p-q\| \le \max_{p\in\pi}\min_{q\in p_1p_n}\|p-q\|$. We can derive

$$\begin{aligned}
d_{\mathrm{H}}(\pi, p_1p_n) &= \max\left\{\max_{p\in\pi}\min_{q\in p_1p_n}\|p-q\|, \max_{q\in p_1p_n}\min_{p\in\pi}\|p-q\|\right\}\\
&= \max_{p\in\pi}\min_{q\in p_1p_n}\|p-q\|\\
&\overset{\text{def}}{=} \max_{p\in\pi} d(p, p_1p_n)\\
&\overset{\text{def}}{=} \max_{i\in[n-1]}\max_{p\in p_ip_{i+1}} d(p, p_1p_n)\\
&\quad \{\text{Lemma 5.4}\}\\
&= \max_{i\in[n-1]}\max\left\{d(p_i, p_1p_n), d(p_{i+1}, p_1p_n)\right\}\\
&\overset{\text{def}}{=} \max_{i\in[n]} d(p_i, p_1p_n),
\end{aligned}$$

as was to be shown. □

Indecisive points. We are now ready to generalise the setting to include imprecision. We first show that the straightforward setting with indecisive points permits an easy solution using Lemma 5.5.

Lemma 5.6. *Given $n, k \in \mathbb{N}$, $n \geq 3$, for any indecisive curve $\mathcal{U} = \langle U_1, \dots, U_n \rangle$ with $U_i = \{p_i^1, \dots, p_i^k\}$ for all $i \in [n]$ and $p_i^j \in \mathbb{R}^2$ for all $i \in [n]$, $j \in [k]$, and given some $p_1 \in U_1$ and $p_n \in U_n$, we have*

$$\max_{\pi \Subset \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} d_{\mathrm{H}}(\pi, p_1 p_n) = \max_{i \in \{2, \dots, n-1\}} \max_{j \in [k]} d(p_i^j, p_1 p_n) \, .$$

Proof. Assume the setting of the lemma statement. Derive

$$\begin{aligned}
& \max_{\pi \Subset \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} d_{\mathrm{H}}(\pi, p_1 p_n) \\
& \quad \{\text{Lemma 5.5}\} \\
= & \max_{\pi \Subset \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} \max_{i \in [n]} d(\pi(i), p_1 p_n) \\
& \quad \{\text{Def. } \Subset, \, d(p_1, p_1 p_n) = d(p_n, p_1 p_n) = 0\} \\
= & \max_{i \in \{2, \dots, n-1\}} \max_{p \in U_i} d(p, p_1 p_n) \\
& \quad \{\text{Def. indecisive point}\} \\
= & \max_{i \in \{2, \dots, n-1\}} \max_{j \in [k]} d(p_i^j, p_1 p_n) \, ,
\end{aligned}$$

as was to be shown. ☐

Note that this means that when the start and end realisations are fixed, we can test that a shortcut is valid using the lemma above in time $O(nk)$ for a shortcut of length n.

Disks. We proceed to present the way to test shortcuts for fixed realisations of the first and the last points when the imprecision is modelled using disks. In the next arguments the following well-known form of a triangle inequality is useful.

Lemma 5.7. *Given a metric space (X, d) and a non-empty subset $S \subset X$, $S \neq \emptyset$, for any $x, y \in X$,*

$$d(x, S) \leq d(x, y) + d(y, S) \, .$$

Proof. Pick some $z' \in S$ and $x, y \in X$. By the triangle inequality, $d(x, z') \leq d(x, y) + d(y, z')$, so

$$d(x, S) \stackrel{\text{def}}{=} \inf_{z \in S} d(x, z) \leq d(x, z') \leq d(x, y) + d(y, z') \, ,$$

and this holds for *any* choice of z'. Therefore, we conclude

$$d(x, S) \leq d(x, y) + \inf_{z \in S} d(y, z) \overset{\text{def}}{=} d(x, y) + d(y, S)\,. \qquad \square$$

Corollary 5.8. *For any $p, q \in \mathbb{R}^2$ and a line segment ab on $a, b \in \mathbb{R}^2$,*

$$d(p, ab) \leq \|p - q\| + d(q, ab)\,.$$

We now state the result for disks.

Lemma 5.9. *Given $n \in \mathbb{N}$, $n \geq 3$, for any imprecise curve modelled with disks $\mathcal{U} = \langle U_1, \dots, U_n \rangle$ with $U_i = D(c_i, r_i)$ for all $i \in [n]$ and $c_i \in \mathbb{R}^2$, $r_i \in \mathbb{R}^{\geq 0}$ for all $i \in [n]$, and given some $p_1 \in U_1$ and $p_n \in U_n$, we have*

$$\max_{\pi \Subset \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} d_{\mathrm{H}}(\pi, p_1 p_n) = \max_{i \in \{2, \dots, n-1\}} \big(d(c_i, p_1 p_n) + r_i\big)\,.$$

Proof. Assume the setting of the lemma. As before, we derive

$$\begin{aligned}
&\max_{\pi \Subset \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} d_{\mathrm{H}}(\pi, p_1 p_n)\\
&\quad \{\text{Lemma 5.5}\}\\
={}& \max_{\pi \Subset \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} \max_{i \in [n]} d(\pi(i), p_1 p_n)\\
&\quad \{\text{Def. } \Subset,\ d(p_1, p_1 p_n) = d(p_n, p_1 p_n) = 0\}\\
={}& \max_{i \in \{2, \dots, n-1\}} \max_{p \in U_i} d(p, p_1 p_n)\,.
\end{aligned}$$

It remains to show that $\max_{p \in U_i} d(p, p_1 p_n) = d(c_i, p_1 p_n) + r_i$ for any $i \in \{2, \dots, n-1\}$.

Firstly, pick $p' := \arg\max_{p \in U_i} d(p, p_1 p_n)$. Note that by Corollary 5.8, $d(p', p_1 p_n) \leq \|p' - c_i\| + d(c_i, p_1 p_n)$. Furthermore, as $p' \in U_i$, by definition of U_i, we have $\|p' - c_i\| \leq r_i$. Thus, $\max_{p \in U_i} d(p, p_1 p_n) \leq d(c_i, p_1 p_n) + r_i$, and it remains to show the inequality in the other direction.

Now pick a point $q' := \arg\min_{q \in p_1 p_n} \|q - c_i\|$, so that $d(c_i, p_1 p_n) = \|q' - c_i\|$. Draw the line through c_i and q' and pick the point p' on that line on the boundary of U_i on the opposite side of q w.r.t. c_i. Clearly, $\|p' - c_i\| = r_i$ and $q' = \arg\min_{q \in p_1 p_n} \|q - p'\|$. Thus,

$$d(p', p_1 p_n) = \|p' - q'\| = \|q' - c_i\| + \|p' - c_i\| = d(c_i, p_1 p_n) + r_i\,.$$

Note that $p' \in U_i$, so we conclude $\max_{p \in U_i} d(p, p_1 p_n) \geq d(c_i, p_1 p_n) + r_i$. Hence, the statement of the lemma holds. $\square$

Once again, note that this lemma allows us to test a shortcut in a straightforward manner, in time $\mathcal{O}(n)$ for a shortcut of length n.

Polygonal closed convex sets (PCCSs). We show the following lemma.

Lemma 5.10. *Given $n, k \in \mathbb{N}$, $n \geq 3$, for any imprecise curve modelled with PCCSs $\mathcal{U} = \langle U_1, \dots, U_n \rangle$ with $U_i \subset \mathbb{R}^2$ and $V(U_i) = \{p_i^1, \dots, p_i^k\}$ for all $i \in [n]$, and given some $p_1 \in U_1$ and $p_n \in U_n$, we have*

$$\max_{\pi \Subset \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} d_{\mathrm{H}}(\pi, p_1 p_n) = \max_{i \in \{2, \dots, n-1\}} \max_{v \in V(U_i)} d(v, p_1 p_n)\,.$$

Proof. Assume the setting of the lemma. As before, derive

$$\begin{aligned} & \max_{\pi \Subset \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} d_{\mathrm{H}}(\pi, p_1 p_n) \\ & \quad \{\text{Lemma 5.5}\} \\ = & \max_{\pi \Subset \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} \max_{i \in [n]} d(\pi(i), p_1 p_n) \\ & \quad \{\text{Def. } \Subset, d(p_1, p_1 p_n) = d(p_n, p_1 p_n) = 0\} \\ = & \max_{i \in \{2, \dots, n-1\}} \max_{p \in U_i} d(p, p_1 p_n)\,. \end{aligned}$$

To show that the claim holds, it remains to show that for any PCCS U and a line segment ab, it holds that $\max_{p \in U} d(p, ab) = \max_{v \in V(U)} d(v, ab)$. Firstly, as $V(U) \subset U$, immediately $\max_{p \in U} d(p, ab) \geq \max_{v \in V(U)} d(v, ab)$. Consider any $p \in U$. We show that there is some $v \in V(U)$ such that $d(v, ab) \geq d(p, ab)$, completing the proof. We do a case distinction on p.

- $p \in V(U)$. Then pick $v := p$, and we are done.
- $p \notin V(U)$, but p is on the boundary of U. Consider the vertices $v, w \in V(U)$ with $p \in vw$. Using Lemma 5.4, we note

  $$\max_{q \in vw} d(q, ab) = \max\big\{d(v, ab), d(w, ab)\big\}\,.$$

  W.l.o.g. suppose $d(v, ab) \geq d(w, ab)$. Then for v, indeed we have $d(v, ab) \geq d(p, ab)$.
- p is in the interior of U (cannot occur for line segments). Find the point $q' := \arg\min_{q \in ab} \|p - q\|$, so $d(p, ab) = \|p - q'\|$. Draw the line through p and q'; let p' be the point on that line on

Algorithm 5.1. Testing a shortcut on a precise curve with the Fréchet distance.

```
▷ Input constraint: $\pi = \langle p_1, \dots, p_n \rangle, n \in \mathbb{N}, \forall i \in [n] : p_i \in \mathbb{R}^2, \varepsilon \in \mathbb{R}^+$
function CheckFréchetPrecise($\pi, n, \varepsilon$)
    $s_1 := 1$
    for all $i \in \{2, \dots, n-1\}$ do
        $S_i := \{t \in [s_{i-1}, 2] \mid \|p_i - p_1p_n(t)\| \le \varepsilon\}$
        if $S_i = \emptyset$ then
            return False
        $s_i := \min S_i$
    return True
```

the boundary of U on the opposite side of q' w.r.t. p. Clearly, $q' = \arg\min_{q \in ab} \|p' - q\|$, so $d(p', ab) > d(p, ab)$. Then we can find a vertex $v \in V(U)$ as in the previous cases, yielding $d(v, ab) \ge d(p', ab) > d(p, ab)$.

This covers all the cases, so the statement holds. □

As before, this lemma gives us a simple way to test the shortcut with fixed realisations of the first and the last points in time $O(nk)$ for a shortcut of length n and PCCSs with k vertices.

5.2.2 Fréchet Distance

We now turn our attention to the Fréchet distance. In this section, we do not show results for the Fréchet distance in the precise setting. For extra intuition, see Algorithm 5.1, which follows from a well-known fact shown e.g. by Guibas et al. [132, Lemma 8]; it can also be seen as specialisation of the indecisive case to $k = 1$ or of the disk case to $r = 0$.

Indecisive points. The idea is that in the precise case we can always align greedily as we move along the line segment. In this case, we also need to find the realisation for each indecisive point that makes for the 'worst' greedy choice.

Lemma 5.11. *Given $n, k \in \mathbb{N}$ and $\varepsilon \in \mathbb{R}^+$, for any indecisive curve $\mathcal{U} = \langle U_1, \dots, U_n \rangle$ with $U_i = \{p_i^1, \dots, p_i^k\}$ for all $i \in [n]$ and $p_i^j \in \mathbb{R}^2$ for all $i \in [n]$, $j \in [k]$, and given some $p_1 \in U_1$ and $p_n \in U_n$, we have, using*

Algorithm 5.2,

$$\max_{\pi \Subset \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} d_{\mathrm{F}}(\pi, p_1 p_n) \leq \varepsilon$$

if and only if

$$\textsc{CheckFréchetInd}(\mathcal{U}, p_1, p_n, n, k, \varepsilon) = \mathsf{True}\,.$$

Proof. First, assume that $\max_{\pi \Subset \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} d_{\mathrm{F}}(\pi, p_1 p_n) \leq \varepsilon$. In the algorithm, we compute some set S_i^j for each p_i^j and then pick one value from it and add it to T_i; from T_i we then pick a single value as s_i. So, $s_i \in S_i^j$ for some $j_i \in [k]$, on every iteration $i \in \{2, \ldots, n-1\}$. Consider a realisation $\pi \Subset \mathcal{U}$ with $\pi(1) \equiv p_1$, $\pi(n) \equiv p_n$, and $\pi(i) \equiv p_i^{j_i}$ for every $i \in \{2, \ldots, n-1\}$, where j_i is chosen as the value corresponding to s_i. Then we know $d_{\mathrm{F}}(\pi, p_1 p_n) \leq \varepsilon$. So, there is an alignment that can be given as a sequence of n positions, $t_i \in [1, 2]$, such that $\|\pi(i) - p_1 p_n(t_i)\| \leq \varepsilon$ and $t_i \leq t_{i+1}$ for all i. The alignment is established by interpolating between the consecutive points on the curves, as discussed in Chapter 2.

We now show by induction that $s_i \leq t_i$ for all i. For $i = 2$, we get, for the chosen j_2, $s_2 := \min\{t \in [1, 2] \mid \|p_2^{j_2} - p_1 p_n(t)\| \leq \varepsilon\}$. As we have $t_2 \in \{t \in [1, 2] \mid \|p_2^{j_i} - p_1 p_n(t)\| \leq \varepsilon\}$, we get $s_2 \leq t_2$. Now assume the statement holds for some i, then for $i + 1$, we get $s_{i+1} := \min\{t \in [s_i, 2] \mid \|p_{i+1}^{j_{i+1}} - p_1 p_n(t)\| \leq \varepsilon\}$; we can rephrase this so that

$$s_{i+1} \stackrel{\text{def}}{=} \min\bigl(\{t \in [1, 2] \mid \|p_{i+1}^{j_{i+1}} - p_1 p_n(t)\| \leq \varepsilon\} \cap [s_i, 2]\bigr)\,.$$

So, there are two options.

- $s_{i+1} = s_i$. Then we know $s_{i+1} = s_i \leq t_i \leq t_{i+1}$.
- $s_{i+1} > s_i$. Then we can use the same argument as for $i = 2$ to find that $s_{i+1} \leq t_{i+1}$.

Now we know that for every i, $t_i \in S_i^{j_i}$ for the choice of j_i described above. Therefore, for any $p_{i+1}^{j_{i+1}}$, there is always a realisation prefix such that any valid alignment has $t_{i+1} \geq s_i$; as we know that there is a valid alignment for every realisation, we conclude that every S_i^j is non-empty. Thus, the algorithm returns True.

Now assume that the algorithm returns True. Consider any realisation $\pi \Subset \mathcal{U}$. We claim that there is a valid alignment, described with a

Algorithm 5.2. Testing a shortcut on an indecisive curve with the Fréchet distance.

```
▷ Input constraint: $\mathcal{U} = \langle U_1, \dots, U_n \rangle$, $n, k \in \mathbb{N}$, $\forall i \in [n] : U_i = \{p_i^1, \dots, p_i^k\}$, $\forall i \in [n], j \in [k] : p_i^j \in \mathbb{R}^2$, $\varepsilon \in \mathbb{R}^+$, $p_1 \in U_1$, $p_n \in U_n$
function CheckFréchetInd($\mathcal{U}, p_1, p_n, n, k, \varepsilon$)
    $s_1 := 1$
    for all $i \in \{2, \dots, n-1\}$ do
        $T_i := \emptyset$
        for all $j \in [k]$ do
            $S_i^j := \{t \in [s_{i-1}, 2] \mid \|p_i^j - p_1p_n(t)\| \leq \varepsilon\}$
            if $S_i^j = \emptyset$ then
                return False
            $T_i := T_i \cup \min S_i^j$
        $s_i := \max T_i$
    return True
```

sequence of $t_i \in [1,2]$ for $i \in \{2, \dots, n-1\}$, such that $s_{i-1} \leq t_i \leq s_i$ and $\|p_1p_n(t_i) - \pi(i)\| \leq \varepsilon$. Denote realisation $\pi \overset{\text{def}}{=} \langle p_1, p_2^{j_2}, p_3^{j_3}, \dots, p_{n-1}^{j_{n-1}}, p_n \rangle$, so the sequence $\langle j_2, \dots, j_{n-1} \rangle$ describes the choices of the realisation. Consider the set $S_i^{j_i}$ for any $i \in \{2, \dots, n-1\}$. We know that it is non-empty, otherwise the algorithm would have returned False. We claim that we can pick $t_i = \min S_i^{j_i}$ for every i. By definition, $S_i^{j_i} \subseteq [1,2]$ and $\|p_1p_n(t_i) - \pi(i)\| \leq \varepsilon$. We also trivially get that $s_{i-1} \leq t_i$. Finally, note that $t_i \in T_i$, and $s_i := \max T_i$, so $t_i \leq s_i$.

This argument shows that $t_i \leq t_{i+1}$ for every i, and $\|p_1p_n(t_i) - \pi(i)\| \leq \varepsilon$. Thus, $d_{\mathrm{F}}(\pi, p_1p_n) \leq \varepsilon$. As it holds for any realisation with $\pi(1) \equiv p_1$ and $\pi(n) \equiv p_n$, we conclude $\max_{\pi \Subset \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} d_{\mathrm{F}}(\pi, p_1p_n) \leq \varepsilon$. □

Disks. To show the generalisation to disks, it is helpful to reframe the problem as that of disk stabbing for appropriate disks. We demonstrate some useful facts first.

Lemma 5.12. *Given a disk* $D_1 := D(c, r)$ *with* $c \in \mathbb{R}^2$, $r \in \mathbb{R}^+$, *a threshold* $\varepsilon \in \mathbb{R}^+$, *and a point* $p \in \mathbb{R}^2$, *define* $D_2 := D(c, \varepsilon - r)$. *We have*

$$\max_{p' \in D_1} \|p - p'\| \leq \varepsilon \iff p \in D_2 \,.$$

Proof. First, assume $p \in D_2 \overset{\text{def}}{=} \{s \in \mathbb{R}^2 \mid \|s - c\| \leq \varepsilon - r\}$; thus, we know $\|p - c\| \leq \varepsilon - r$. Take $q := \arg\max_{p' \in D_1} \|p - p'\|$. Then $q \in D_1 \overset{\text{def}}{=} \{s \in \mathbb{R}^2 \mid \|s - c\| \leq r\}$, so $\|q - c\| \leq r$. Then by the triangle inequality,

$$\|p - q\| \leq \|p - c\| + \|q - c\| \leq \varepsilon - r + r = \varepsilon\,.$$

Now assume that $p \notin D_2 \overset{\text{def}}{=} \{s \in \mathbb{R}^2 \mid \|s - c\| \leq \varepsilon - r\}$. Then $\|p - c\| > \varepsilon - r$. Consider a point q on the line pc on the boundary of D_1, so that c is between p and q on the line. Note that $q \in D_1$, so

$$\max_{p' \in D_1} \|p - p'\| \geq \|p - q\| = \|p - c\| + \|q - c\| > \varepsilon - r + r = \varepsilon\,,$$

completing the proof. □

We can now generalise the previous statement to talk about distance to line segments.

Lemma 5.13. *Given a disk $D_1 := D(c, r)$ with $c \in \mathbb{R}^2$, $r \in \mathbb{R}^+$, a threshold $\varepsilon \in \mathbb{R}^+$, and a line segment pq with $p, q \in \mathbb{R}^2$, define $D_2 := D(c, \varepsilon - r)$. We have*

$$\max_{p' \in D_1} d(p', pq) \leq \varepsilon \iff pq \cap D_2 \neq \emptyset\,.$$

Proof. First, assume $pq \cap D_2 \neq \emptyset$. Take $t \in pq \cap D_2$. Consider an arbitrary point $s \in D_1$. By Lemma 5.12, we know that $\|t - s\| \leq \varepsilon$; so also $d(s, pq) \overset{\text{def}}{=} \min_{q' \in pq} \|q' - s\| \leq \|t - s\| \leq \varepsilon$. As this holds for arbitrary $s \in D_1$, we conclude $\max_{p' \in D_1} \min_{q' \in pq} \|p' - q'\| \leq \varepsilon$.

Now assume that $\max_{p' \in D_1} d(p', pq) \leq \varepsilon$. In disks, it is easy to see that the furthest point of a disk from a line segment is positioned in a way that the centre of the disk is on the line through the point of the disk and the closest point of the line segment, so in our case $c \in st$, where $s := \arg\max_{p' \in D_1} \min_{q' \in pq} \|p' - q'\|$ and $t := \arg\min_{q' \in pq} \|s - q'\|$. Then $\|t - c\| = \|t - s\| - \|s - c\| \leq \varepsilon - r$, so indeed $t \in D_2$, and $pq \cap D_2 \neq \emptyset$. □

Lemma 5.14. *Given $n \in \mathbb{N}$ and $\varepsilon \in \mathbb{R}^+$, for any imprecise curve modelled with disks $\mathcal{U} = \langle U_1, \ldots, U_n \rangle$ with $U_i = D(c_i, r_i)$ for all $i \in [n]$ and $c_i \in \mathbb{R}^2$, $r_i \in \mathbb{R}^+$ for all $i \in [n]$, and given some $p_1 \in U_1$ and $p_n \in U_n$, we have, using Algorithm 5.3,*

$$\max_{\pi \Subset \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} d_F(\pi, p_1 p_n) \leq \varepsilon$$

Algorithm 5.3. Testing a shortcut on an imprecise curve modelled with disks with the Fréchet distance.

```
▷ Input constraint: $\mathcal{U} = \langle U_1, \dots, U_n \rangle, n \in \mathbb{N}, \varepsilon \in \mathbb{R}^+, \forall i \in [n] : U_i = D(c_i, r_i)$,
  $\forall i \in [n] : c_i \in \mathbb{R}^2, r_i \in \mathbb{R}^+, p_1 \in U_1, p_n \in U_n$
function CheckFréchetDisks($\mathcal{U}, p_1, p_n, n, \varepsilon$)
    $s_1 := 1$
    for all $i \in \{2, \dots, n-1\}$ do
        $S_i := \{t \in [s_{i-1}, 2] \mid \|c_i - p_1p_n(t)\| \leq \varepsilon - r_i\}$
        if $S_i = \emptyset$ then
            return False
        $s_i := \min S_i$
    return True
```

if and only if

$$\textsc{CheckFréchetDisks}(\mathcal{U}, p_1, p_n, n, \varepsilon) = \mathit{True}\,.$$

Proof. It is convenient to use Lemma 5.13 to change the problem: rather than establishing an alignment that comes in the correct order and satisfies the distance constraints, we can do disk stabbing and pick the stabbing points in the correct order. Therefore, we have $\max_{\pi \Subset \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} d_{\mathrm{F}}(\pi, p_1p_n) \leq \varepsilon$ if and only if there exists a sequence of points $p'_i \in p_1p_n \cap D(c_i, \varepsilon - r_i)$ for all $i \in \{2, \dots, n-1\}$ such that $p'_i \preccurlyeq p'_{i+1}$ along p_1p_n for all $i \in \{2, \dots, n-2\}$. It remains to show that this is exactly what Algorithm 5.3 computes.

Assume the algorithm returns True. We claim that in this case, the alignment obtained by $p'_i := p_1p_n(s_i)$ satisfies the conditions. First, by definition, $s_i \in S_i \overset{\text{def}}{=} \{t \in [s_{i-1}, 2] \mid \|c_i - p_1p_n(t)\| \leq \varepsilon - r_i\}$, so we have $\|c_i - p'_i\| \leq \varepsilon - r_i$, so indeed $p'_i \in p_1p_n \cap D(c_i, \varepsilon - r_i)$. Furthermore, by construction, $s_i \in [s_{i-1}, 2]$, so $s_{i-1} \leq s_i$, and hence $p'_{i-1} \preccurlyeq p'_i$.

Now assume that the conditions hold, so there is some valid alignment, represented by a sequence of points p'_i. We claim that for all $i \in \{2, \dots, n-1\}$, we have $p_1p_n(s_i) \preccurlyeq p'_i$. For $i = 2$, this clearly holds, as $p_1p_n(s_2)$ is the first point that falls into $p_1p_n \cap D(c_2, \varepsilon - r_2)$. Now assume this holds for some i, and we will show that it holds for iteration $i + 1$. On iteration $i + 1$, there are two possibilities:

- $s_i > s_{i-1}$; then we are in the same situation as for $i = 2$, so $p_1p_n(s_i) \preccurlyeq p'_i$.

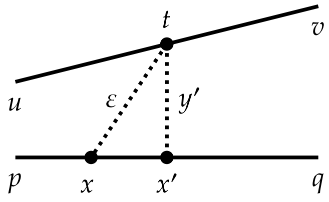


Figure 5.3. Illustration for the computation in Lemma 5.15.

- $s_i = s_{i-1}$; then we immediately get the same result, as also $p'_i \preccurlyeq p'_{i+1}$.

Therefore, we can conclude that the algorithm finds an alignment if one exists, as all t_i such that $p_1p_n(t_i) \equiv p'_i$ fall inside S_i, so all S_i are non-empty, and the algorithm returns True. □

Polygonal closed convex sets.

Lemma 5.15. *Given $n, k \in \mathbb{N}$ and $\varepsilon \in \mathbb{R}^+$, for any imprecise curve modelled with PCCSs $\mathcal{U} = \langle U_1, \dots, U_n \rangle$ with $U_i \subset \mathbb{R}^2$ and $V(U_i) = \{p_i^1, \dots, p_i^k\}$ for all $i \in [n]$, and given some $p_1 \in U_1$ and $p_n \in U_n$, we have, using Algorithm 5.4,*

$$\max_{\pi \Subset \mathcal{U}, \pi(1) \equiv p_1, \pi(n) \equiv p_n} d_F(\pi, p_1p_n) \le \varepsilon$$

if and only if

$$\textsc{CheckFréchet}PCCS(\mathcal{U}, p_1, p_n, n, k, \varepsilon) = \textit{True}\,.$$

Proof. As we have shown in Lemma 5.10, it suffices to test the vertices of a PCCS to establish that the distance from every point to the line segment is below the threshold. It remains to show that the extreme alignment (in terms of ordering) for the Fréchet distance is also achieved at a vertex. This case then becomes identical to the indecisive points case, so we can use Lemma 5.11 to show correctness.

Consider an arbitrary point $t \in U_i$ and let s be the earliest point in the ε-disk around t that is on pq. Clearly, if t is in the interior of U_i, then we can take any t' on the line through t parallel to pq and get the corresponding s' with $s \prec s'$. So, assume t is on the boundary of U_i. Suppose that $t \in uv$ with $u, v \in V(U_i)$. Rotate and translate the coordinate plane so that pq lies on the x-axis. Derive the equation for the line containing uv, say, $y' = kx' + b$. First consider $k = 0$, so the line containing uv is parallel to the line containing pq. In this case, clearly,

Algorithm 5.4. Testing a shortcut on an imprecise curve modelled with PCCSs with the Fréchet distance.

```
▷ Input constraint: $\mathcal{U} = \langle U_1, \dots, U_n \rangle$, $n, k \in \mathbb{N}$, $\forall i \in [n], j \in [k] : p_i^j \in \mathbb{R}^2$, $\forall i \in [n] : U_i$ is a PCCS, $V(U_i) = \{p_i^1, \dots, p_i^k\}$, $\varepsilon \in \mathbb{R}^+$, $p_1 \in U_1$, $p_n \in U_n$
function CheckFréchetPCCS($\mathcal{U}, p_1, p_n, n, k, \varepsilon$)
    $s_1 := 1$
    for all $i \in \{2, \dots, n-1\}$ do
        $T_i := \emptyset$
        for all $j \in [k]$ do
            $S_i^j := \{t \in [s_{i-1}, 2] \mid \|p_i^j - p_1 p_n(t)\| \le \varepsilon\}$
            if $S_i^j = \emptyset$ then
                return False
            $T_i := T_i \cup \min S_i^j$
        $s_i := \max T_i$
    return True
```

moving along uv in the direction coinciding with the direction from p to q increases the x-coordinate of point of interest, so moving to a vertex is optimal. Now assume $k > 0$. If $k < 0$, reflect the coordinate plane about $y = 0$. Geometrically, it is easy to see (Figure 5.3) that the coordinate of interest can be expressed as

$$x = x' - \sqrt{\varepsilon^2 - y'^2} = \frac{y' - b}{k} - \sqrt{\varepsilon^2 - y'^2}\,.$$

We want to maximise x by picking the appropriate y'. We take the derivative:

$$\frac{\mathrm{d}x}{\mathrm{d}y'} = \frac{1}{k} + \frac{y'}{\sqrt{\varepsilon^2 - y'^2}}\,.$$

We can equate it to 0 to find the critical point of the function. Simplifying, we find

$$y'_0 = -\frac{\varepsilon}{\sqrt{k^2 + 1}}\,.$$

We can check that for $y' < y'_0$, the value of the derivative is negative, and for $y' > y'_0$ it is positive, so at $y' = y'_0$ we achieve a local minimum. There are no other critical points. Therefore, to maximise x, we want to move as far as possible in either direction, away from the local minimum.

Since we are limited to the line segment uv, the maximum is clearly achieved at one of the segment endpoints. □

5.3 Shortcut Testing: All Points

In the previous section, we have covered testing a shortcut, given that the first and the last points are fixed. Here we remove that restriction.

Problem 5.16. Given an uncertain curve $\mathcal{U} = \langle U_1, \dots, U_n \rangle$ on $n \in \mathbb{N}$, $n \geq 3$ uncertain points in $\mathbb{R}^2$, check if the largest Hausdorff or Fréchet distance between $\mathcal{U}$ and its one-segment simplification is below a threshold $\varepsilon \in \mathbb{R}^+$ for any realisation, i.e. for $\delta \coloneqq d_{\mathrm{H}}$ or $\delta \coloneqq d_{\mathrm{F}}$, verify $\max_{\pi \Subset \mathcal{U}} \delta(\pi, p_1 p_n) \leq \varepsilon$.

We first show how this can be done for indecisive points, both for $\delta \coloneqq d_{\mathrm{H}}$ and $\delta \coloneqq d_{\mathrm{F}}$. We can simply test the shortcut using the corresponding procedure from Lemma 5.6 or Lemma 5.11, and do so for each combination of the start and the end points. We can then test an indecisive shortcut of length n overall in time $O(k^2 \cdot nk) = O(nk^3)$.

Lemma 5.17. *Given $n, k \in \mathbb{N}$, $n \geq 3$, and $\delta \coloneqq d_{\mathrm{H}}$ or $\delta \coloneqq d_{\mathrm{F}}$, for any indecisive curve $\mathcal{U} = \langle U_1, \dots, U_n \rangle$ with $U_i = \{p_i^1, \dots, p_i^k\}$ for all $i \in [n]$ and $p_i^j \in \mathbb{R}^2$ for all $i \in [n]$, $j \in [k]$, we have*

$$\max_{\pi \Subset \mathcal{U}} \delta(\pi, \langle \pi(1), \pi(n) \rangle) = \max_{a \in [k]} \max_{b \in [k]} \max_{\sigma \Subset \mathcal{U}, \sigma(1) \equiv p_1^a, \sigma(n) \equiv p_n^b} \delta(\sigma, p_1^a p_n^b)\,.$$

Proof. We can derive

$$\begin{aligned}
&\max_{\pi \Subset \mathcal{U}} \delta(\pi, \langle \pi(1), \pi(n) \rangle) \\
&\{\text{Def. } \Subset\} \\
={}& \max_{p_1 \in U_1, \dots, p_n \in U_n} \delta(\langle p_1, \dots, p_n \rangle, p_1 p_n) \\
={}& \max_{p_1 \in U_1} \max_{p_n \in U_n} \max_{p_2 \in U_2, \dots, p_{n-1} \in U_{n-1}} \delta(\langle p_1, \dots, p_n \rangle, p_1 p_n) \\
&\{\text{Def. } \Subset\} \\
={}& \max_{p_1 \in U_1} \max_{p_n \in U_n} \max_{\sigma \Subset \mathcal{U}, \sigma(1) \equiv p_1, \sigma(n) \equiv p_n} \delta(\sigma, p_1 p_n) \\
={}& \max_{a \in [k]} \max_{b \in [k]} \max_{\sigma \Subset \mathcal{U}, \sigma(1) \equiv p_1^a, \sigma(n) \equiv p_n^b} \delta(\sigma, p_1^a p_n^b)\,,
\end{aligned}$$

as was to be shown. □

We now proceed to show the approach for disks and polygonal closed convex sets. The procedure is the same for the Hausdorff and the Fréchet distance, but differs between disks and PCCSs, since disks have some convenient special properties.

5.3.1 Disks

We start by stating some useful observations.

Observation 5.18. *Suppose we are given two non-degenerate disks* $D_1 := D(p_1, r_1)$ *and* $D_2 := D(p_2, r_2)$ *with* $D_1 \not\subseteq D_2$ *and* $D_2 \not\subseteq D_1$. *We make the following observations. (See Figure 5.4.)*

- *There are exactly two* outer tangents *to the disks, and the convex hull of* $D_1 \cup D_2$ *consists of an arc from* D_1*, an arc from* D_2*, and the outer tangents.*
- *Assume the lines of the outer tangents intersect. When viewed from the intersection point, the order in which the tangents touch the disks is the same, i.e. either both first touch* D_1 *and then* D_2*, or the other way around. If the lines are parallel, the same statement holds when viewed from points on the tangent lines at infinity.*

To see that the second observation is true, note that the distance from the intersection point to the tangent points of a disk is the same for both tangent lines. These observations mean that we can restrict our attention to the area bounded by the outer tangents and define an ordering in the resulting strip.

Definition 5.19. Given two distinct non-degenerate disks $D_1 := D(p_1, r_1)$ and $D_2 := D(p_2, r_2)$, consider a strip defined by the lines that form the outer tangents to the disks. Assume we have two circular arcs O_1, O_2 that intersect both tangents and lie inside the strip. Define s_1 and v_1 to be the points where one of the tangents touches D_1 and D_2, respectively, and let t_1 and u_1 be the points where O_1 and O_2 intersect that tangent, respectively. Define the order on the tangents from D_1 to D_2, so $s_1 \prec v_1$. Define points s_2, t_2, u_2, v_2 similarly for the other tangent. We say that O_2 is *to the right* of O_1 if either $t_i = u_i$ for $i \in \{1, 2\}$ and the radius of O_1 is larger than that of O_2; or if otherwise $t_i \preccurlyeq u_i$ for $i \in \{1, 2\}$ and O_1 and O_2

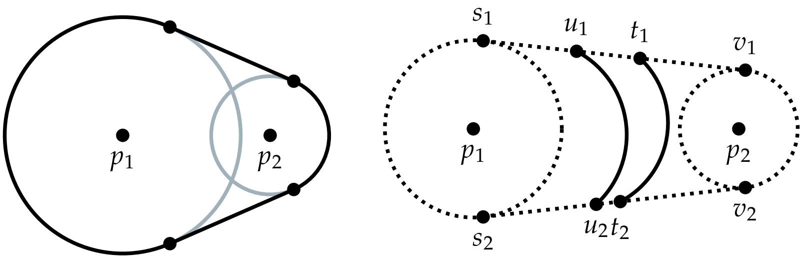


Figure 5.4. Left: Illustration for Observation 5.18. The convex hull of the disks is highlighted in black. The order in which the outer tangents touch the disks is the same. Right: Illustration for Definition 5.19. Here O_1 (t_1 to t_2) is to the right of O_2 (u_1 to u_2).

do not properly intersect. We say that O_2 is *to the left* of O_1 if either $t_i = u_i$ for $i \in \{1, 2\}$ and the radius of O_1 is smaller than that of O_2; or if otherwise $u_i \preccurlyeq t_i$ for $i \in \{1, 2\}$ and O_1 and O_2 do not properly intersect. (See Figure 5.4 for a visual interpretation.)

We are now ready to state the main result for the Hausdorff distance.

Lemma 5.20. *Given $n \in \mathbb{N}$, $n \geq 3$, for any imprecise curve modelled with disks $\mathcal{U} = \langle U_1, \ldots, U_n \rangle$ with $U_i = D(c_i, r_i)$ for all $i \in [n]$ and $c_i \in \mathbb{R}^2$, $r_i \in \mathbb{R}^+$ for all $i \in [n]$, and assuming $U_1 \neq U_n$, we have*

$$\max_{\pi \Subset \mathcal{U}} d_{\mathrm{H}}(\pi, \langle \pi(1), \pi(n) \rangle) \leq \varepsilon$$

if and only if both of the following are true:

- $\max\left\{ \max_{\pi \Subset \mathcal{U}, \pi(1) \equiv s, \pi(n) \equiv t} d_{\mathrm{H}}(\pi, st), \max_{\pi \Subset \mathcal{U}, \pi(1) \equiv u, \pi(n) \equiv v} d_{\mathrm{H}}(\pi, uv) \right\} \leq \varepsilon\,,$

  where $s, u \in U_1$, $t, v \in U_n$, and st and uv are the outer tangents to $U_1 \cup U_n$; and
- *for each $i \in \{2, \ldots, n-1\}$, the right arc of the disk $D(c_i, \varepsilon - r_i)$ bounded by the intersection points with the tangent lines is to the right of the right arc of U_1 and the left arc of the disk $D(c_i, \varepsilon - r_i)$ is to the left of the left arc of U_n.*

Proof. Assume the right side of the lemma statement holds. First of all, as we have $\max_{\pi \Subset \mathcal{U}, \pi(1) \equiv s, \pi(n) \equiv t} d_{\mathrm{H}}(\pi, st) \leq \varepsilon$, Lemma 5.9 shows that for all $i \in \{2, \ldots, n-1\}$, we have $d(c_i, st) + r_i \leq \varepsilon$, or $d(c_i, st) \leq \varepsilon - r_i$, so st stabs each disk $D(c_i, \varepsilon - r_i)$. We can draw a similar conclusion for uv. Therefore, each disk $D(c_i, \varepsilon - r_i)$ crosses the entire strip bounded by the tangent lines, with the intersection points splitting it into the left and the right circular arcs. We can thus apply Definition 5.19 to these arcs, as stated in the lemma.

First suppose that the disks U_1 and U_n do not intersect. Then for any line segment from U_1 to U_n and any disk $D' \coloneqq D(c_i, \varepsilon - r_i)$, we exit D' after exiting U_1 and enter D' before entering U_n. Hence, for any line pq with $p \in U_1$ and $q \in U_n$ and any $i \in \{2, \ldots, n-1\}$, we can find a point $w \in pq \cap D'$; this means, as stated in Lemma 5.13, that indeed $\max_{w' \in U_i} d(w', pq) \leq \varepsilon$. As this holds for all disks and any choice of p and q, we conclude that $\max_{\pi \Subset \mathcal{U}} d_{\mathrm{H}}(\pi, \langle \pi(1), \pi(n) \rangle) \leq \varepsilon$.

Now assume that the disks U_1 and U_n intersect. If we consider the line segment pq with $p \in U_1$, $q \in U_n$, we end up in the previous case if either $p \notin U_1 \cap U_n$ or $q \notin U_1 \cap U_n$. So assume that the segment pq lies entirely in the intersection $U_1 \cap U_n$. However, it can be seen that for each disk $D' \coloneqq D(c_i, \varepsilon - r_i)$, the left boundary of the intersection is to the right of the left boundary of the disk, and the right boundary of the intersection is to the left of the right boundary of the disk; hence, $pq \subset U_1 \cap U_n \subseteq D'$. Therefore, we have $\max_{w' \in U_i} d(w', pq) \leq \varepsilon$, and so also in this case $\max_{\pi \Subset \mathcal{U}} d_{\mathrm{H}}(\pi, \langle \pi(1), \pi(n) \rangle) \leq \varepsilon$.

We now assume that the right side of the lemma statement is false and show that then $\max_{\pi \Subset \mathcal{U}} d_{\mathrm{H}}(\pi, \langle \pi(1), \pi(n) \rangle) > \varepsilon$. If it holds that $\max_{\pi \Subset \mathcal{U}, \pi(1) \equiv s, \pi(n) \equiv t} d_{\mathrm{H}}(\pi, st) > \varepsilon$, then we can immediately observe that $\max_{\pi \Subset \mathcal{U}} d_{\mathrm{H}}(\pi, \langle \pi(1), \pi(n) \rangle) > \varepsilon$. Same holds for uv. So, assume those statements hold; then it must be that for at least one intermediate disk the arcs do not lie to the left or to the right of the arcs of the respective disks. Assume this is disk i, so the disk $D' \coloneqq D(c_i, \varepsilon - r_i)$. W.l.o.g. assume that the right arc of the disk does not lie entirely to the right of the right arc of U_1. The argument for the left arc w.r.t. U_n is symmetric.

There must be at least one point p' on the right arc of U_1 that lies outside of D'. Assume for now that U_1 and U_n are disjoint. Then a line segment $p'q$ for any $q \in U_n$ does not stab D', so $\max_{w' \in U_i} d(w', pq) > \varepsilon$, and so $\max_{\pi \Subset \mathcal{U}} d_{\mathrm{H}}(\pi, \langle \pi(1), \pi(n) \rangle) > \varepsilon$. If U_1 and U_n intersect, then either p' is outside of the intersection and of D' and there is a point $q \in U_n$ such that $p'q$ does not stab D'; or we can pick the degenerate line segment $p'p'$, as $p' \in U_1 \cap U_n$, and so $p'p'$ also does not stab D'. In either case, we conclude that $\max_{\pi \Subset \mathcal{U}} d_{\mathrm{H}}(\pi, \langle \pi(1), \pi(n) \rangle) > \varepsilon$. □

It is also worth noting that the case of $U_1 = U_n$ is similar to how we treat the intersection $U_1 \cap U_n$ above; however, our definition for the ordering between two disks does not apply. So, if $U_1 = U_n$, then $\max_{\pi \Subset \mathcal{U}} d_{\mathrm{H}}(\pi, \langle \pi(1), \pi(n) \rangle) \le \varepsilon$ if and only if $U_1 \subseteq D(c_i, \varepsilon - r_i)$ for all $i \in \{2, \dots, n-1\}$.

Similarly, we state the following for the Fréchet distance.

Lemma 5.21. *Given $n \in \mathbb{N}$, $n \ge 3$, for any imprecise curve modelled with disks $\mathcal{U} = \langle U_1, \dots, U_n \rangle$ with $U_i = D(c_i, r_i)$ for all $i \in [n]$ and $c_i \in \mathbb{R}^2$, $r_i \in \mathbb{R}^+$ for all $i \in [n]$, and assuming $U_1 \ne U_n$, we have*

$$\max_{\pi \Subset \mathcal{U}} d_{\mathrm{F}}(\pi, \langle \pi(1), \pi(n) \rangle) \le \varepsilon$$

if and only if both of the following are true:

- $\max\left\{ \max_{\pi \Subset \mathcal{U}, \pi(1) \equiv s, \pi(n) \equiv t} d_{\mathrm{F}}(\pi, st), \max_{\pi \Subset \mathcal{U}, \pi(1) \equiv u, \pi(n) \equiv v} d_{\mathrm{F}}(\pi, uv) \right\} \le \varepsilon$,

  where $s, u \in U_1$, $t, v \in U_n$, and st and uv are the outer tangents to $U_1 \cup U_n$; and
- *for each $i \in \{2, \dots, n-1\}$, the right arc of the disk $D(c_i, \varepsilon - r_i)$ bounded by the intersection points with the tangent lines is to the right of the right arc of U_1 and the left arc of the disk $D(c_i, \varepsilon - r_i)$ is to the left of the left arc of U_n.*

Proof. First assume that $\max_{\pi \Subset \mathcal{U}} d_{\mathrm{F}}(\pi, \langle \pi(1), \pi(n) \rangle) \le \varepsilon$. This also means that $\max_{\pi \Subset \mathcal{U}} d_{\mathrm{H}}(\pi, \langle \pi(1), \pi(n) \rangle) \le \varepsilon$ as $d_{\mathrm{F}}(\pi, \sigma) \ge d_{\mathrm{H}}(\pi, \sigma)$ for any curves π, σ. Furthermore, we can immediately conclude that $\max_{\pi \Subset \mathcal{U}, \pi(1) \equiv s, \pi(n) \equiv t} d_{\mathrm{F}}(\pi, st) \le \varepsilon$, and the same for uv. Together with Lemma 5.20, this yields the right side of the lemma.

Now assume that the right side holds. As in Lemma 5.20, we know that the disks cross the entire strip and that Definition 5.19 applies. It remains to show that for any line segment pq with $p \in U_1$, $q \in U_n$, there is a valid alignment that maintains the correct ordering and bottleneck distance, assuming it exists for every realisation for st and uv. Consider a valid alignment established for st and uv, so the sequence of points a_i on st and b_i on uv that are mapped to U_i. As we showed in Lemma 5.12, we can always find such points for each individual U_i, and as we know that the Fréchet distance is below the threshold for st and uv, there is such a valid alignment, i.e. we know that $a_i \preccurlyeq a_{i+1}$ and $b_i \preccurlyeq b_{i+1}$ for all $i \in [n-1]$.

First suppose that the disks U_1 and U_n do not intersect. Consider the region R bounded by the outer tangents and the disk arcs that are not part of the convex hull of $U_1 \cup U_n$. We connect, for each $i \in \{2, \dots, n-1\}$, a_i to b_i with a geodesic shortest path in R. We claim that for any line segment pq defined above, the intersection points of the shortest paths with the segment give a valid alignment, yielding $\max_{\pi \Subset \mathcal{U}, \pi(1) \equiv p, \pi(n) \equiv q} d_{\mathrm{F}}(\pi, pq) \leq \varepsilon$. As the choice of pq was arbitrary, this will complete the proof.

To show that the alignment is valid, we need to show that the order is correct and that the distances fall below the threshold. First consider the case where the geodesic shortest path for point i does not touch the boundary formed by arcs of region R. In this case, it is simply a line segment $a_i b_i$. Note that by definition $a_i, b_i \in D(c_i, \varepsilon - r_i)$; as disks are convex, also $a_i b_i \subset D(c_i, \varepsilon - r_i)$; thus, the intersection point p'_i of pq with $a_i b_i$ is in $D(c_i, \varepsilon - r_i)$, so by Lemma 5.12, $\max_{w \in U_i} \|p'_i - w\| \leq \varepsilon$. Furthermore, note that $a_i \preccurlyeq a_{i+1}$ and $b_i \preccurlyeq b_{i+1}$; thus, the line segments $a_i b_i$ and $a_{i+1} b_{i+1}$ cannot cross, so also $p'_i \preccurlyeq p'_{i+1}$.

Now w.l.o.g. consider the case where the geodesic shortest path for point i touches the arc of U_1. The geodesic shortest paths do not cross: on the path from a_i (or b_i) to the arc they form a tangent to the arc, thus for $a_i \preccurlyeq a_{i+1}$, the tangent point for a_i comes before that of a_{i+1} when going along the arc from s to u. So, just as in the previous case, these line segments cannot cross. Having reached the arc, both shortest paths will follow it, as otherwise the path would not be a shortest path; thus, the arcs do not cross, either. Finally, a path from the previous case does not touch any path that touches the arc boundary of R by definition. Finally, note that the condition that we have established on the right arcs of disks being to the right of the right arc of U_1 (and symmetric for the left arcs and U_n) means that the geodesic shortest paths that touch the arc boundary of R stay within the respective disks $D(c_i, \varepsilon - r_i)$. Thus, we have established that for all i, we have $p'_i \preccurlyeq p'_{i+1}$ and $\max_{w \in U_i} \|p'_i - w\| \leq \varepsilon$, concluding the proof for disjoint U_1 and U_n.

Finally, consider the case where U_1 intersects U_n. Above we used geodesic paths within the region R. However, when U_1 intersects U_n, R consists of two disconnected regions. Observe that one region contains a_i

and the other contains b_i. To connect a_i with b_i we use the geodesic from a_i to the intersection point of the two inner boundaries of U_1 and U_n that is in the same region of R, the geodesic from b_i to the other intersection point of the inner boundaries, and join these two by a line segment between the intersection points. Any line segment from a point in U_1 to a point in U_n crosses these paths in order, just like in the previous case. If the line segment goes through the intersection, note that any point in the intersection is close enough to all the intermediate objects, as the intersection is the subset of each disk. So, any point in the intersection can be chosen to establish the trivially in-order alignment to all the intermediate objects. □

Again, if $U_1 = U_n$, note that $\max_{\pi \Subset \mathcal{U}} d_{\mathrm{F}}(\pi, \langle \pi(1), \pi(n) \rangle) \leq \varepsilon$ if and only if $U_1 \subseteq D(c_i, \varepsilon - r_i)$ for all $i \in \{2, \ldots, n-1\}$.

5.3.2 Non-intersecting PCCSs

Suppose the regions are modelled by convex polygons. Consider first the case where the interiors of U_1 and U_n do not intersect, so at most they share a boundary segment.

Observation 5.22. *Given an uncertain curve modelled by convex polygons* $\mathcal{U} = \langle U_1, \ldots, U_n \rangle$ *with the interiors of* U_1 *and* U_n *not intersecting, note:*

- *There are two* outer tangents *to the polygons* U_1 *and* U_n*, and the convex hull of* $U_1 \cup U_2$ *consists of a convex chain from* U_1*, a convex chain from* U_n*, and the outer tangents.*
- *Let* C_i *be the convex chain from* U_i *that is not a part of the convex hull for* $i \in \{1, n\}$*. Then for* $\delta := d_{\mathrm{H}}$ *or* $\delta := d_{\mathrm{F}}$*,*

  $$\max_{\pi \Subset \mathcal{U}} \delta(\pi, \langle \pi(1), \pi(n) \rangle) \leq \varepsilon$$

  if and only if

  $$\max_{\pi \Subset \mathcal{U}, \pi(1) \in C_1, \pi(n) \in C_n} \delta(\pi, \langle \pi(1), \pi(n) \rangle) \leq \varepsilon\,.$$

To see that the second observation is true, note that one direction is trivial. In the other direction, note that any line segment pq with $p \in U_1$, $q \in U_n$ crosses both C_1 and C_n, say, at $p' \in C_1$ and $q' \in C_n$. We know that there is a valid alignment for $p'q'$, both for the Hausdorff and the Fréchet distance; we can then use this alignment for pq. See Figure 5.5.

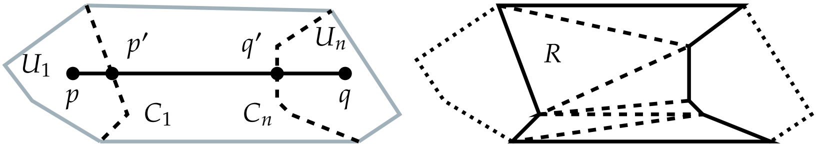


Figure 5.5. Left: Illustration for Observation 5.22. The convex hull of the disks is shown in grey. The dotted chains are C_1 and C_n. Any line segment pq with $p \in U_1$ and $q \in U_n$ crosses C_1 and C_n. Right: Illustration for the procedure. The region R is triangulated.

We claim that we can use the following procedure to check whether $\max_{\pi \Subset \mathcal{U}} d_{\mathrm{H}}(\pi, \langle \pi(1), \pi(n) \rangle) \leq \varepsilon$.

1. Triangulate the region R bounded by two convex chains C_1 and C_n and the outer tangents.
2. For each line segment st of the triangulation with $s \in C_1$, $t \in C_n$, check that $\max_{\pi \Subset \mathcal{U}, \pi(1) \equiv s, \pi(n) \equiv t} \delta(\pi, st) \leq \varepsilon$ for either $\delta := d_{\mathrm{H}}$ or $\delta := d_{\mathrm{F}}$.

First of all, observe that we can compute a triangulation, and that every triangle has two points from one convex chain and one point from the other chain (see Figure 5.5). If all three points were from the same chain, then the triangle would lie outside of R. Now consider some line segment pq with $p \in C_1$, $q \in C_n$. To complete the argument, it remains to show that the checks in step 2 mean that also $\max_{\pi \Subset \mathcal{U}, \pi(1) \equiv p, \pi(n) \equiv q} \delta(\pi, pq) \leq \varepsilon$. Observe that the triangles span across the region R, so when going from one tangent to the other within R we cross all the triangles. Therefore, we can order them, in the order of occurrence on such a path, from 1 to k. Denote the alignment established on line $j \in [k]$ with the sequence of a_i^j, for $i \in [n]$; this alignment can be established both for $\delta := d_{\mathrm{H}}$ and $\delta := d_{\mathrm{F}}$. We can then establish polygonal curves $A_i := \langle a_i^1, \dots, a_i^k \rangle$; clearly, they all stay within R. We claim that for any line segment pq defined above, it is possible to establish a valid alignment from intersection points of pq and A_i. We do this separately for the Fréchet and the Hausdorff distance.

Lemma 5.23. *Given a set of curves $A := \{A_2, \dots, A_{n-1}\}$ in R described above for $\delta := d_{\mathrm{H}}$ and a line segment pq with $p \in C_1$, $q \in C_n$, we have $\max_{\pi \Subset \mathcal{U}, \pi(1) \equiv p, \pi(n) \equiv q} d_{\mathrm{H}}(\pi, pq) \leq \varepsilon$.*

Proof. Note that pq crosses each A_i at least once. We can take any one crossing for each i and establish the alignment. Consider such a crossing point p'_i. It falls in some triangle bounded by a segment from either C_1 or C_n and two line segments that contain points a_i^j and a_i^{j+1} for some $j \in [k]$. We know, using Lemma 5.10, that $\max_{w \in U_i} \|a_i^j - w\| \le \varepsilon$ and $\max_{w \in U_i} \|a_i^{j+1} - w\| \le \varepsilon$. Consider any point $w' \in U_i$. Then, using Lemma 5.4 with $c := d := w'$, we find that $\|w' - p'_i\| \le \varepsilon$. Therefore, also $\max_{w \in U_i} \|p'_i - w\| \le \varepsilon$; using Lemma 5.10, we conclude that indeed $\max_{\pi \Subset \mathcal{U}, \pi(1) \equiv p, \pi(n) \equiv q} d_{\mathrm{H}}(\pi, pq) \le \varepsilon$. □

For the Fréchet distance, we can use the same argument to show closeness; however, we need more care to establish the correct order for the alignment to be valid.

Lemma 5.24. *Given a set of curves $A := \{A_2, \dots, A_{n-1}\}$ in R described above for $\delta := d_{\mathrm{F}}$ and a line segment pq with $p \in C_1$, $q \in C_n$, we have $\max_{\pi \Subset \mathcal{U}, \pi(1) \equiv p, \pi(n) \equiv q} d_{\mathrm{F}}(\pi, pq) \le \varepsilon$.*

Proof. Compared to Lemma 5.23, instead of taking any intersection point of pq with each A_i, we take the *last* intersection point.

We need to show, first of all, that curves A_i and A_{i+1} do not cross for any $i \in [n-1]$. Note that each curve A_i crosses each triangle once, so it suffices to show that a segment $a_i^j a_i^{j+1}$ does not cross $a_{i+1}^j a_{i+1}^{j+1}$. Indeed, as $a_i^j \preccurlyeq a_{i+1}^j$ and $a_i^{j+1} \preccurlyeq a_{i+1}^{j+1}$, these line segments cannot cross.

Now consider, for each $i \in \{2, \dots, n-1\}$, the polygon P_i bounded by C_1, A_i, and the corresponding segments of the outer tangents. With the previous statement, it is easy to see that $P_2 \subseteq P_3 \subseteq \cdots \subseteq P_{n-1}$. Assume this is not the case, so some $P_i \nsubseteq P_{i+1}$. Then there is a point $z \in P_i$, but $z \notin P_{i+1}$. The point z falls into some triangle with lines j and $j+1$. In this triangle, it means that z is between C_1 and $a_i^j a_i^{j+1}$, but not between C_1 and $a_{i+1}^j a_{i+1}^{j+1}$. However, as these segments do not cross, this would imply that $a_{i+1}^j \prec a_i^j$, but then the check in step 2 would not pass for line j.

Consider the points at which the line segment pq leaves the polygons P_i for the last time. From the definition it is obvious that $p \in P_i$ for all $i \in \{2, \dots, n-1\}$, so this is well-defined. Clearly, due to the subset relationship, the order of such points p'_i is correct, i.e. $p'_i \preccurlyeq p'_{i+1}$. Furthermore, each such $p'_i \in A_i$, so using the arguments of Lemma 5.23, we can show that also the distances are below ε. Thus, we conclude that indeed $\max_{\pi \Subset \mathcal{U}, \pi(1) \equiv p, \pi(n) \equiv q} d_{\mathrm{F}}(\pi, pq) \leq \varepsilon$. □

The proofs of Lemmas 5.23 and 5.24 show us how to solve the problem for two convex polygons with non-intersecting interiors. We can also use them directly for the case of line segments that do not intersect except at endpoints.

Corollary 5.25. *Given $n \in \mathbb{N}$, $n \geq 3$, for any imprecise curve modelled with line segments $\mathcal{U} = \langle U_1, \dots, U_n \rangle$ with $U_i = p_i^1 p_i^2 \subset \mathbb{R}^2$ for all $i \in [n]$, given a threshold $\varepsilon \in \mathbb{R}^+$, and given that $U_1 \cap U_n \subset \{p_1^1, p_1^2\}$, and assuming that the triangles $p_1^1 p_n^1 p_1^2$ and $p_1^2 p_n^1 p_n^2$ form a triangulation of the convex hull of $U_1 \cup U_n$, we have*

$$\max_{\pi \Subset \mathcal{U}} \delta(\pi, \langle \pi(1), \pi(n) \rangle) \leq \varepsilon$$

if and only if

$$\begin{aligned}\max\Big\{ &\max_{\pi \Subset \mathcal{U}, \pi(1) \equiv p_1^1, \pi(n) \equiv p_n^1} \delta(\pi, p_1^1 p_n^1)\,, \\ &\max_{\pi \Subset \mathcal{U}, \pi(1) \equiv p_1^2, \pi(n) \equiv p_n^1} \delta(\pi, p_1^2 p_n^1)\,, \\ &\max_{\pi \Subset \mathcal{U}, \pi(1) \equiv p_1^2, \pi(n) \equiv p_n^2} \delta(\pi, p_1^2 p_n^2) \Big\} \leq \varepsilon\,.\end{aligned}$$

We should note that in this particular case it is not necessary to use a triangulation, so we can get rid of the second term; also in the previous proofs a convex partition could work instead, but a triangulation is easier to define.

5.3.3 Intersecting PCCSs

We proceed to discuss the situation where the interiors of U_1 and U_n intersect, or where the line segments U_1 and U_n cross. The argument is the same for both $\delta := d_{\mathrm{H}}$ and $\delta := d_{\mathrm{F}}$, but it is easier to treat line segments and convex polygons separately.

Line segments. Assume line segments $U_1 \overset{\text{def}}{=} p_1^1 p_1^2$ and $U_n \overset{\text{def}}{=} p_n^1 p_n^2$ cross; call their intersection point s. Then we can use Corollary 5.25 separately on pairs of $\{p_1^1 s, s p_1^2\} \times \{p_n^1 s, s p_n^2\}$. Clearly, together this will cover the entire set of realisations of pq with $p \in U_1$, $q \in U_n$, thus completing the checks.

Lemma 5.26. *Given $n \in \mathbb{N}$, $n \geq 3$, for any imprecise curve modelled with line segments $\mathcal{U} = \langle U_1, \ldots, U_n \rangle$ with $U_i = p_i^1 p_i^2 \subset \mathbb{R}^2$ for all $i \in [n]$, given a threshold $\varepsilon \in \mathbb{R}^+$, we can check for both $\delta := d_\mathrm{H}$ and $\delta := d_\mathrm{F}$, using procedures above, that*

$$\max_{\pi \Subset \mathcal{U}} \delta(\pi, \langle \pi(1), \pi(n) \rangle) \leq \varepsilon\,.$$

Convex polygons. Convex polygons whose interiors intersect can be partitioned along the intersection lines, so into a convex polygon $R := U_1 \cap U_n$ and two sets of polygons $\mathcal{P}_1 := \{P_1^1, \ldots, P_1^k\}$ and $\mathcal{P}_n := \{P_n^1, \ldots, P_n^\ell\}$ for some $k, \ell \in \mathbb{N}$. Just as for line segments, we can look at pairs from $\mathcal{P}_1 \times \mathcal{P}_n$ separately. The pairs where R is involved are treated later. Consider some $(P, Q) \in \mathcal{P}_1 \times \mathcal{P}_n$. Note that P and Q are convex polygons with a convex cut-out, so the boundary forms a convex chain, followed by a concave chain. We need to compute some convex polygons P' and Q' with non-intersecting interiors that are equivalent to P and Q, so that we can apply the approaches from Section 5.3.2.

We claim that we can simply take the convex hull of P and Q to obtain P' and Q'. Clearly, the resulting polygons will be convex. Also, the concave chains of P are bounded by points s and t and are replaced with the line segment st; same happens for Q with points u and v. The points s, t, u, v are points of intersection of original polygons U_1 and U_n, so they lie on the boundary of R, and their order along that boundary can only be s, t, u, v or s, t, v, u. Thus, it cannot happen that st crosses uv, and it cannot be that uv is in the interior of the convex hull of P, as otherwise R would not be convex. Hence, the interiors of P' and Q' cannot intersect, so they satisfy the necessary conditions.

Finally, we need to show that the solution for (P', Q') is equivalent to that for (P, Q). One direction is trivial, as $P \subseteq P'$ and $Q \subseteq Q'$; for the other direction, consider any line segment that leaves P through the concave chain. In our approach, we test the lines starting in s and t; the established alignments are connected into paths. The paths A_i do not cross st. So, any alignment in the region of $\mathrm{CH}(P \cup Q) \setminus (P \cup Q)$ can also be made in the region $\mathrm{CH}(P' \cup Q') \setminus (P' \cup Q')$. So, this approach yields valid solutions for all pairs not involving R.

Now consider the pair (R, R). A curve may now consist of a single point, so the approach for the Fréchet and the Hausdorff distance is the same: all the points of U_i need to be close enough to all the points of R. To check that, observe that the pair of points $p \in U_i$ and $q \in R$ that has maximal distance has the property that p is an extreme point of U_i in direction qp and q is an extreme point of R in direction pq. So, it suffices, starting at the rightmost point of U_i and the leftmost point of R in some coordinate system, to then rotate clockwise around both regions keeping track of the distance between tangent points. Note that only vertices need to be considered, as the extremal point cannot lie on an edge. Finally, any other pair that involves R is covered by the stronger case of (R, R): for any line, we can align every intermediate object to any point in R.

Lemma 5.27. *Given $n \in \mathbb{N}$, $n \geq 3$, for any imprecise curve modelled with convex polygons $\mathcal{U} = \langle U_1, \dots, U_n \rangle$ with $U_i \subset \mathbb{R}^2$ for all $i \in [n]$ and $V(U_i) = \{p_i^1, \dots, p_i^k\}$ for all $i \in [n]$, $k \in \mathbb{N}$, given a threshold $\varepsilon \in \mathbb{R}^+$, we can check for both $\delta := d_\mathrm{H}$ and $\delta := d_\mathrm{F}$, using procedures above, that*

$$\max_{\pi \Subset \mathcal{U}} \delta(\pi, \langle \pi(1), \pi(n) \rangle) \leq \varepsilon\,.$$

5.4 Combining Steps

In Sections 5.2 and 5.3, we have established correctness of the routines that can be used to check if a shortcut is valid under either the Hausdorff distance or the Fréchet distance. In this section, we summarise the approach, discuss the shortcut graph, and analyse the running times.

Lemma 5.28. *Given $n \in \mathbb{N}$, for any uncertain curve modelled with indecisive points, disks, or PCCSs $\mathcal{U} = \langle U_1, \dots, U_n \rangle$, and given a threshold $\varepsilon \in \mathbb{R}^+$, and fixing either $\delta := d_\mathrm{H}$ or $\delta := d_\mathrm{F}$, if we can check in time T for any pair $i, j \in [n]$, $i < j$ that*

$$\max_{\pi \Subset \mathcal{U}[i:j]} \delta(\pi, \langle \pi(1), \pi(j - i + 1) \rangle) \leq \varepsilon\,,$$

then in time $O(Tn^2)$ we can find the shortest index subsequence $I \subseteq [n]$ with $|I| = \ell$ such that for all $j \in [\ell]$,

$$\max_{\pi \Subset \mathcal{U}[I(j):I(j+1)]} \delta(\pi, \langle \pi(1), \pi(I(j+1) - I(j) + 1) \rangle) \leq \varepsilon\,.$$

Proof. The approach is simple: construct a graph $G := (V, E)$ with $V := \{v_1, \dots, v_n\}$ and $(v_i, v_j) \in E$ if and only if $\max_{\pi \Subset \mathcal{U}[i:j]} \delta(\pi, \langle \pi(1), \pi(j -$

$i+1)\rangle) \leq \varepsilon$. Clearly, this takes $O(Tn^2)$ time. Any path in the graph from v_1 to v_n gives a subsequence for which the condition in the statement of the lemma holds; there are no simplifications that would not correspond to such a path; thus, finding the shortest path in G using e.g. BFS in time $O(n^2)$ indeed yields the answer. □

It is easy to see that the result of the lemma is exactly the problem we were trying to solve: obtaining a single simplification such that no matter which realisation of the curve is chosen, the resulting realisation of the simplification is valid.

We now proceed to recap the methods for checking the shortcuts. For indecisive points, one can test all combinations for the first and the last point of the shortcut, as in Lemma 5.17, and for each such combination do the testing either for the Hausdorff or the Fréchet distance, as in Lemmas 5.6 and 5.11.

For imprecise points modelled with disks, it suffices to test the outer tangents and check some extra conditions on the intermediate disks, as in Lemmas 5.20 and 5.21. For the outer tangents, the testing can be done using the approaches of Lemmas 5.9 and 5.14.

For imprecise points modelled with line segments, one can split the first and the last one into regions if they cross, as in Lemma 5.26, and apply Corollary 5.25 to each pair. The testing of the outer tangents can be done using Lemmas 5.10 and 5.15 for the Hausdorff and the Fréchet distance, respectively.

Finally, for imprecise points modelled with convex polygons, we again split the first and the last one into regions if their interiors intersect, as in Lemma 5.27, and apply Lemmas 5.23 and 5.24. To test each shortcut with the fixed endpoints, we can again use Lemmas 5.10 and 5.15.

Having constructed the graph, we can find the shortest path through it from vertex corresponding to U_1 to that corresponding to U_n, as discussed in Lemma 5.28.

Theorem 5.29. *We can solve the problem of finding the shortest vertex-constrained simplification of an uncertain curve, such that for any realisation, the simplification is valid, both for the Hausdorff and the Fréchet distance, and for uncertainty modelled using indecisive points, disks, line segments, or convex polygons in time shown in Table 5.1.*

Proof. Correctness of the approaches has been shown before. For the running time, observe that we need $O(n^2T)$ time in any setting, due to the shortcut graph construction.

For indecisive points, when testing a shortcut, we do $O(nk)$-time testing for $O(k^2)$ combinations of starting and ending points, where k is the number of options per point. For disks, we do a linear number of constant-time checks and two linear-time checks, getting $T \in O(n)$. For line segments, we also do two (three) linear-time checks per part; two line segments can be split into at most two parts each, so we repeat the process four times. Either way, we get $T \in O(n)$.

Finally, for convex polygons, assume the complexity of each polygon is at most k. Assume the partitioning resulting from two intersecting polygons yields ℓ_1 and ℓ_2 parts for the first and the second polygon, respectively. Denote the two polygons P and Q and the resulting parts with $P_1, \dots, P_{\ell_1}$ and $Q_1, \dots, Q_{\ell_2}$, respectively. Suppose part P_i has complexity k_i and part Q_j has complexity k'_j, so $|V(P_i)| = k_i$ and $|V(Q_j)| = k'_j$ for some $i \in [\ell_1]$, $j \in [\ell_2]$. We know that every vertex of the original polygons occurs in a constant number of parts, so $\sum_{i=1}^{\ell_1} k_i \in O(k)$ and $\sum_{j=1}^{\ell_2} k'_j \in O(k)$; we also know $\ell_1 + \ell_2 \in O(k)$. We consider all pairs from P and Q, and for each pair, we triangulate and do the checks on the triangulation. The triangulation can be done in time $O((k_i + k'_j) \cdot \log(k_i + k'_j))$, yielding $O(k_i + k'_j)$ lines, each of which is tested in time $O(nk)$. The testing dominates, so we need $O((k_i + k'_j) \cdot nk)$ time. We are interested in

$$\sum_{i=1}^{\ell_1} \sum_{j=1}^{\ell_2} O((k_i + k'_j) \cdot nk) = O(nk) \cdot \sum_{i=1}^{\ell_1} \sum_{j=1}^{\ell_2} O(k_i + k'_j) = O(nk^3)\,.$$

So, $T \in O(nk^3)$ both for the Fréchet and the Hausdorff distance. □

5.5 Conclusions

In this chapter, we have provided a comprehensive set of approaches to the following problem: given an uncertain curve, simplify it so that

- the output is a subsequence of the input, and
- for any possible realisation of the input curve, the corresponding realisation of the output curve is a valid simplification.

This approach is safe, in the sense that we only throw out points that were sufficiently precise and close to any possible path. This way, we know that we are capturing any area that the subject passed through. The points that are kept introduce the most uncertainty, so they may require special attention. However, one may imagine a different goal of addressing uncertainty—removing any unlikely areas, basing the trajectory on the more precise points. In that setting, we would likely want to get rid of the larger uncertainty regions, so it would require a different definition of a valid simplification. Studying such variants would form an interesting future research direction.

CHAPTER 6

Map-Matching Queries under Fréchet Distance on Low-Density Spanners

In this chapter, we turn our attention to map matching, introduced in Section 1.2. Map matching is inherently about uncertainty: since vehicles commonly move on road networks, we can reduce the imprecision introduced by measurement errors by matching their trajectories to a road network (map) in a way that most closely resembles the original. This chapter discusses map matching from a data structure perspective, that is, we preprocess a map so that given a query trajectory, we can efficiently report a path on the map that is most similar to the query. To measure similarity, we use the Fréchet distance, like in the previous chapters—it is a natural metric for trajectories. In order to make fast query times possible, we need to make realistic assumptions about the map; we discuss all of these aspects in detail below. We assume that the trajectory is a polygonal curve and the map is an undirected geometric graph in the plane. Such a graph has its vertices embedded in $\mathbb{R}^2$ with straight-line edges connecting them.

The map-matching problem has received considerable attention, as we discuss in Section 1.2. In particular, there is a significant amount of work on map matching under the Fréchet distance, including a seminal paper by Alt et al. [22]. Their algorithm requires $O(mn \log mn \log n)$ time and $O(mn)$ space to match a polygonal curve of length m to a planar graph $G = (V, E)$ with complexity $|V| + |E| = n$. As shown via a conditional lower bound by Gudmundsson et al. [127], this query time is close to optimal for planar graphs: there is no algorithm that runs in $O((mn)^{1-\delta})$ time for any $\delta > 0$ that solves this problem after polynomial-time preprocessing of the graph. However, real-world road networks are rarely planar due to the presence of bridges and tunnels, so would like to find a different assumption.

Chen et al. [79] study the map-matching problem under *realistic input* assumptions, which aim to exclude particular types of degenerate instances to provide stronger results. In their work in particular, the graph has low density and the trajectory is c-packed. A polygonal curve is called *c-packed* if in any ball of radius r, the total length of the curve inside the ball is at most cr. We can use a similar definition for geometric graphs, measuring the total length of the edges inside the ball instead. The assumption that a geometric graph is c-packed is very strong; in search of a different approach, let us define low density.

Definition 6.1. A geometric graph $P = (V, E)$ is *λ-low density* [199, 204] if for every disk of radius $r > 0$ in the plane, there are at most λ edges of length at least $2r$ that intersect the disk.

Our work has no assumptions on the trajectories at the expense of stricter assumptions on the maps (geometric graphs). Most often one will have a large number of trajectories being mapped to a relatively complex network, so to avoid the steep dependency on network complexity when matching every trajectory, we consider the query version of the problem. We preprocess the map so that we can quickly answer many map-matching queries, where each query is a trajectory. To our knowledge, this problem has only been studied on c-packed graphs [127, 128]. Gudmundsson and Smid [128] show an approach for c-packed trees with long edges and query trajectories with long edges. Gudmundsson et al. [127] assume that the graph is c-packed, but do not impose any restrictions on the query trajectories. However, c-packedness is not a realistic assumption for graphs representing road networks. Consider the example map of Figure 6.1: it is not c-packed for any constant c, as that would require the total length of roads be at most cr in all disks

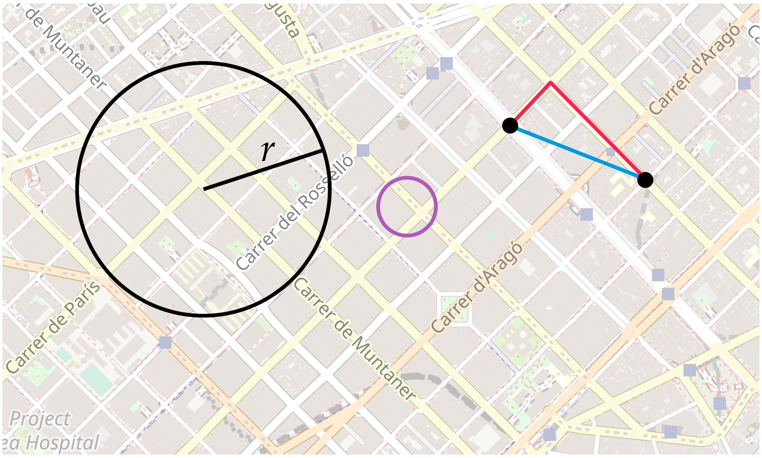


Figure 6.1. An example road network in the centre of Barcelona. The total road length in a disk of some radius r is closer to cr^2 than cr, so this road network is not c-packed. However, the number of long edges intersecting the purple disk is small, and the red path is not much longer than the blue path, so the network is λ-low density and a t-spanner for small λ and t. Map data from OpenStreetMap [188].

of radius r, and it is instead often much closer to cr^2. On some scale, this problem arises with many road networks, including city streets or motorways. Therefore, we would like to devise an approach with more realistic assumptions on the graph.

We instead assume that our graph has low density, defined above. As observed by Chen et al. [79], the value of density does not grow with the considered area on the map, and road networks typically have low density; it is also a strictly weaker assumption than packedness. We further assume that the graph is a *t-spanner.* A geometric graph is a t-spanner if for any two vertices, the length of the shortest path between them in the graph is at most t times larger than the Euclidean distance between them. Road networks, in particular in urban areas, are typically good spanners [26, 184]. Recalling the example of Figure 6.1, it is clear that this road network is a t-spanner and λ-low density for some low t and λ. Compared to previous work, our assumptions make the approach significantly more applicable on real-life road networks.

In this chapter, we solve the *map-matching query problem* under realistic assumptions:

Problem 6.2. Given a geometric graph P, construct a data structure that can answer the following queries: for a polygonal curve Q in $\mathbb{R}^2$,

1. compute $\min_\pi d_F(\pi, Q)$ and
2. report $\arg\min_\pi d_F(\pi, Q)$,

where π ranges over all paths between two vertices in P and d_F denotes the Fréchet distance.

We present a $(1+\varepsilon)$-approximation under the assumptions listed above. Our approach differentiates from previous work by Gudmundsson et al. [127] in two key aspects:

- we require the graph P to be λ-low density and a t-spanner, rather than c-packed, which is a more realistic assumption for a road network [26, 79, 184], while still allowing the query curve to remain unrestricted;
- we solve the problem of reporting the path that minimises the Fréchet distance, which was stated as an open problem in their paper.

In order to achieve these results, we have to use different techniques, albeit at the cost of a $\sqrt{n}$ factor replacing a polylogarithmic factor in the running time. Where the paper by Gudmundsson et al. [127] uses a semi-separated pair decomposition, we construct a hierarchy of small balanced separators and store appropriate associated data to guide the search for the optimal Fréchet distance. A *balanced separator* of a graph $P = (V, E)$ is a set of vertices $S \subseteq V$ that splits P into connected components of size at most $c \cdot |V|$ for constant c. Combining the changes in analysis and the capability to report a path, we get the following result. Here $n = |V| + |E|$ for a geometric graph $P = (V, E)$, and m is the number of vertices on the query polygonal curve.

Theorem 6.3. *Suppose we are given a λ-low-density t-spanner of complexity n and a fixed $0 < \varepsilon < 1$. Let $\chi = {}^1/_{\varepsilon^2} \log {}^1/_\varepsilon$ and let $\varphi = ({}^\lambda/_{\varepsilon^3} + {}^{t^2}/_{\varepsilon^2})^2$. In time $O(\lambda\chi^2 n^{5/2} \log n)$ and using $O(\lambda\chi^2 n^{3/2})$ space, we can construct a data structure for Problem 6.2 achieving a $(1+\varepsilon)$-approximation that performs distance queries in time $O(m\sqrt{n} \log mn \cdot \varphi \cdot {}^\lambda/_\varepsilon \cdot (\log^2 n + \log n \cdot \varphi + \varphi \cdot {}^\lambda/_\varepsilon))$, and answers the reporting queries for a path of length ℓ in $O({}^\ell/_\varepsilon)$ extra time.*

In a typical setting, when λ and t are small constants, our data structure uses $O(\varepsilon^{-4} \log \varepsilon^{-1} n^{3/2})$ space, resolving to $O(n^{3/2})$ for fixed ε, and supports distance queries in time $O(m\sqrt{n}\log mn \cdot \varepsilon^{-7} \cdot (\log^2 n + \varepsilon^{-6}\log n + \varepsilon^{-7}))$, resolving to $O(m\sqrt{n}\log mn \log^2 n)$ for fixed ε.

Preliminaries. In this chapter, we work with geometric graphs, that is, graphs embedded in the plane with straight-line edges. For a graph $P = (V, E)$, we denote its complexity with $n = |V| + |E|$.

We denote the fact that a path π goes from $u \in V$ to $v \in V$ by $\pi : u \rightsquigarrow v$. Recall that for points $x, y \in \mathbb{R}^2$, we denote their Euclidean distance with $\|x - y\|$. We can consider the edges weighted by defining the weight of an edge $e \in E$ as the Euclidean distance between its endpoints: $|e| = \|u - v\|$ for e connecting $u, v \in V$. We define the *graph distance* d_P as the shortest path distance along the graph between any two vertices $u, v \in V$: $d_P(u, v) = \sum_{e\in\pi}|e|$, where $\pi : u \rightsquigarrow v$ is the shortest path in P between u and v.

We assume our input graph has low density in this chapter, as defined previously. We need to define two more graph properties that we use in our construction.

Definition 6.4. A graph $P = (V, E)$ is *τ-lanky* [167] if for every disk of radius $r > 0$ centred at any vertex $v \in V$, there are at most τ edges of length at least r that are cut by the disk.

Here an edge is *cut* by a disk if exactly one of its endpoints is inside the disk. It is easy to see that a τ-lanky graph has bounded degree of at most τ; and that any λ-low-density graph is also λ-lanky. Let us also formally define a t-spanner.

Definition 6.5. A graph $P = (V, E)$ is called a *t-spanner* if for any two vertices $u, v \in V$, we have $d_P(u, v) \leq t \cdot \|u - v\|$.

A query is a polygonal curve in the plane, that is, a sequence of points in $\mathbb{R}^2$ connected with line segments. For a query curve Q, let m be the number of points in the sequence.

Organisation. The rest of the chapter is organised as follows. We first tackle the simpler problem of finding a path in the graph that most closely follows a line segment between two vertices of the graph in terms of the Fréchet distance in Section 6.1. In that setting, we find a 3-approximation. The data structure we develop there is used later for obtaining a search window for when the end points of the path are not given. In Section 6.2, we generalise this to an arbitrary query line segment that does not have to start or end at a graph vertex, and show how to achieve a $(1+\varepsilon)$-approximation. We also describe how to report a path that corresponds to a $(1+\varepsilon)$-approximation. Finally, in Section 6.3, we show how to combine the segment queries in order to handle a complete polygonal curve.

6.1 Straight Path Queries

In this section, we present a 3-approximation to the following problem, so that for the value r that we return, we have $\min_\pi d_\mathrm{F}(\pi, uv) \leq r \leq 3 \cdot \min_\pi d_\mathrm{F}(\pi, uv)$.

Problem 6.6. Given a geometric graph $P = (V, E)$, construct a data structure that can answer the following queries: for a pair of vertices $u, v \in V$, compute $\min_\pi d_\mathrm{F}(\pi, uv)$, where $\pi : u \rightsquigarrow v$ is a path in P.

In Section 6.2, we show how to generalise the query to arbitrary segment endpoints and how to improve the result to a $(1+\varepsilon)$-approximation for any fixed $\varepsilon > 0$.

Let $n = |V| + |E|$. In order to solve the problem efficiently, we impose an additional constraint on P—we require that P has a graph property satisfying two criteria:

1. the property is decreasing monotone, so it holds on all induced subgraphs;
2. and any graph with the property admits a small separator.

An example of such a property is planarity: any subgraph of a planar graph is planar, and the existence of small separators in planar graphs is a classical result [170]. However, not all road networks are planar, as most road networks include bridges and tunnels. Instead, we require that P is τ-lanky [167]. It is trivial to show that any subgraph of a τ-lanky graph is also τ-lanky; and Le and Than [167] show that a τ-lanky graph of complexity n admits a balanced separator of size $O(\tau\sqrt{n})$ that can be found in $O(\tau n)$ expected time.

Intuition. When constructing the data structure, we can use the algorithm by Alt et al. [22] in order to compute the Fréchet distance between a line segment and a path in the graph. At query time, running that algorithm would be prohibitively slow. We also want to achieve subquadratic storage, so we cannot precompute the distances for all pairs of vertices.

Broadly, the idea is to find sufficient structure in the graph to be able to find a small set of vertices so that any path in the graph passes through at least one of these vertices; we call them *transit* vertices. Then we can precompute the distances between the optimal path and the line segment when going from any vertex of the graph to one of the transit vertices. At query time, we then only need to find an optimal transit vertex. Since we are composing two paths, the computed distance is only a 3-approximation.

More specifically, a balanced separator in a graph forms a set of transit vertices. We can compute them hierarchically and store the separators and the precomputed distances in a binary tree. With some organisation, at query time, we can efficiently find all the relevant transit vertices—the ones that may separate the two query vertices on a path.

Data structure. We construct a hierarchy of separators on the graph and store it with some extra information in a binary tree. Each node in the tree represents both an induced subgraph of P, and a separator of that induced subgraph. Consider the node i corresponding to some induced subgraph $P_i = (V_i, E_i)$ of P. Conceptually, the node represents the balanced separator S_i, so the subset of vertices of P_i, splitting it into two subgraphs A_i and B_i.

The root stores S_1 and the extra information for all pairs from $V \times S_1$, so the top-level balanced separator for the entire graph. The two children of each node correspond to the subgraphs A_i and B_i. The recursion ends when the subgraphs in the leaves have constant size. In a leaf i, assign $S_i = V_i$, so compute the distances for all pairs of vertices.

For every pair of vertices $(u, s) \in V_i \times S_i$, we store $\min_\pi d_F(\pi, us)$, where $\pi : u \rightsquigarrow s$ in P. (Note that a path π may leave P_i.) We call all vertices in S_i *transit* vertices; and all pairs (u, s) for which we store the distances are called *transit pairs.*

In addition, for each vertex $u \in V$, we store a pointer to the tree node i so that $u \in S_i$. There is exactly one such node in the tree for every vertex: if a vertex is part of a separator, it will not be in an induced subgraph further down in the recursion, and if it is never chosen to be in a separator, then it belongs to a leaf, which is treated as S_i in its entirety.

Construction. We construct the hierarchy top–down, computing the separators on the induced subgraphs at every level using the result of Le and Than [167]. For each transit pair (p, s) in a node, we compute the appropriate Fréchet distance in the entire graph using the algorithm by Alt et al. [22], extended by Gudmundsson et al. [127, Lemma 13]. As we construct the separators, we also store in a table the pointer for each vertex to the correct node.

Distance query. Suppose the query is to find the minimal Fréchet distance between the segment uv and any path between u and v. Initialise $\text{opt} = \infty$. First, we use the table to find the pointers to the two nodes in the tree i and j so that $u \in S_i$ and $v \in S_j$. Then we find their lowest common ancestor, call it node a. For every node a' on the path from a to the root of the tree, perform the following procedure.

Denote $D_{xy} = \min_\pi d_\text{F}(\pi, xy)$ over all $\pi : x \rightsquigarrow y$. For the query uv, denote $D'_x = \min_{r \in uv} \|r - x\|$, so the shortest distance between x and any point on uv. For all $s \in S_{a'}$, fetch the stored D_{us} and D_{sv} and compute D'_s. Then compute $D = \max(D_{us}, D_{sv}) + D'_s$ and finally assign $\text{opt} = \min(\text{opt}, D)$. At the end, return opt.

Running time analysis. For the distance queries, we take $O(1)$ time to follow the pointers; $O(\log n)$ time to find the lowest common ancestor; and then $O(1)$ time to check the distance per transit vertex. As we check all transit vertices on the path from the lowest common ancestor to the root, we can write down the worst-case recurrence as

$$T(k) = T(2k/3) + O(\tau\sqrt{k})$$

for a graph on k vertices, since the balanced separator we use subdivides the graph into two subgraphs on at most $2k/3$ vertices. For the entire graph, this resolves to $O(\tau\sqrt{n})$. This term dominates the query time.

For the construction, we need $O(\tau k)$ time to find a separator in a graph of size k. In each node, for each of $O(\tau k\sqrt{k})$ transit pairs, we compute the distance in $O(n \log n)$ time. Assuming we build the tree until the leaves are of constant size, we get the recurrence

$$\begin{aligned} T(k) &= T(2k/3) + T(k/3) + O(\tau k + \tau k\sqrt{k} \cdot n \log n) \\ &= T(2k/3) + T(k/3) + O(\tau k\sqrt{k} \cdot n \log n)\,, \end{aligned}$$

which resolves to $O(\tau n^{5/2} \log n)$ overall.

Space. We store a table of pointers of size $O(n)$ and the main data structure. For a graph on k vertices, we store constant-size data for each transit pair; and there are $O(\tau k\sqrt{k})$ transit pairs. Overall, the space used is represented by the recurrence

$$T(k) = T(2k/3) + T(k/3) + O(\tau k\sqrt{k})\,,$$

which resolves to $O(\tau n\sqrt{n})$.

Correctness. It remains to show that the described query procedure gives us an appropriate distance. First, assume that we do consider an optimal transit vertex; we show that we indeed compute a 3-approximation. The following proof is essentially given by Gudmundsson et al. [127, Theorem 4.1], relying on a further statement [95, Lemma 5.5], and using a semi-separated pair decomposition rather than separators. We include the proof for the sake of completeness and ease of reading.

Lemma 6.7. *Suppose that* $\mathrm{opt} = \min_{\pi'} d_F(\pi', uv)$ *for query* uv *and* $\pi' : u \rightsquigarrow v$*, and that* $\pi = \arg\min_{\pi'} d_F(\pi', uv)$ *passes through a transit vertex* s*, so* $\pi : u \rightsquigarrow s \rightsquigarrow v$*. Let* s' *be the transit vertex that minimises* $D = \max(D_{us'}, D_{s'v}) + D'_{s'}$*. If* s *is considered when finding* D*, then* $\mathrm{opt} \leq D \leq 3 \cdot \mathrm{opt}$.

Proof. Let $\pi_{us'} = \arg\min_{\pi'} d_F(\pi', us')$ over $\pi' : u \rightsquigarrow s'$, and define $\pi_{s'v}$ similarly. Note that the composition of these paths $\pi_{us'} \circ \pi_{s'v}$ does not have to be the same as π. Let t be the point on uv closest to s'. Then

$$\begin{aligned} \mathrm{opt} = d_F(\pi, uv) \leq d_F(\pi_{us'} \circ \pi_{s'v}, uv) &\leq \max\left(d_F(\pi_{us'}, ut), d_F(\pi_{s'v}, tv)\right) \\ &\leq \max\left(d_F(\pi_{us'}, us') + \|s' - t\|, d_F(\pi_{s'v}, s'v) + \|s' - t\|\right) \\ &= \|s' - t\| + \max\left(d_F(\pi_{us'}, us'), d_F(\pi_{s'v}, s'v)\right) \\ &= D'_{s'} + \max(D_{us'}, D_{s'v}) = D\,. \end{aligned}$$

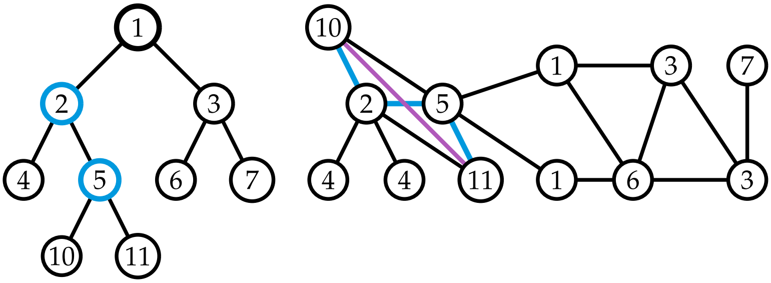


Figure 6.2. A representation of the hierarchy (left) for the graph (right). A query segment is shown in purple, a possible path in blue. We check nodes 5, 2, and 1. If we pick the transit vertex in 5, then the path may be 10 → 2 → 5, so we may need to go up the tree to find the next transit pair.

On the other hand, note that $D \le \max(D_{us}, D_{sv}) + D'_s$, as s' minimises that expression. We also have $D'_s \le d_{\mathrm{F}}(\pi, uv)$. Let $\pi(u,s)$ be the subpath of π from u to s. Let r be the point in uv that is aligned to s in the Fréchet alignment between π and uv. Then

$$\begin{aligned} D_{us} = d_{\mathrm{F}}(\pi_{us}, us) &\le d_{\mathrm{F}}(\pi(u,s), us) \\ &\le d_{\mathrm{F}}(\pi(u,s), ur) + d_{\mathrm{F}}(ur, us) \\ &= d_{\mathrm{F}}(\pi(u,s), ur) + \|r - s\| \\ &\le d_{\mathrm{F}}(\pi, uv) + d_{\mathrm{F}}(\pi, uv) = 2d_{\mathrm{F}}(\pi, uv)\,. \end{aligned}$$

Using the same argument for D_{sv}, we conclude

$$D \le D'_s + \max(D_{us}, D_{sv}) \le d_{\mathrm{F}}(\pi, uv) + 2d_{\mathrm{F}}(\pi, uv) = 3d_{\mathrm{F}}(\pi, uv)\,,$$

and so $\mathrm{opt} \le D \le 3 \cdot \mathrm{opt}$ and the value is a 3-approximation. ☐

Now we show that we consider all the relevant transit vertices. See Figure 6.2.

Lemma 6.8. *For the query uv, the procedure considers a transit vertex s such that s lies on the optimal path $\pi = \arg\min_{\pi'} d_{\mathrm{F}}(\pi', uv)$, where $\pi' : u \rightsquigarrow v$.*

Proof. We consider two cases based on where the lowest common ancestor is found. First, suppose that the lowest common ancestor a contains u, v, or both u and v in the separator. In other words, $u \in S_a$ or $v \in S_a$, and so $s = u$ or $s = v$. Then π passes through s, and we consider s as a transit vertex.

Now assume that the lowest common ancestor a does not contain u or v in the separator; then u and v are separated by S_a. Without loss of generality, let $u \in A_a$ and $v \in B_a$. If the path π stays within the subgraph P_a, then it has to go through some $s \in S_a$, which we consider. If not, then it goes through some separator that separates P_a from the rest of the graph; and we consider exactly all the vertices in these separators, as they fall on the path from a to the root. $\square$

Bringing the above considerations together, we get the main result of this section.

Theorem 6.9. *Given a τ-lanky graph of complexity n, we can construct a data structure for Problem 6.6 in time $O(\tau n^{5/2} \log n)$, using $O(\tau n \sqrt{n})$ space, that supports distance queries in time $O(\tau \sqrt{n})$.*

6.2 Map-Matching Segment Queries

In this section, we generalise the construction we just presented to compute a $(1+\varepsilon)$-approximation and to handle reporting, as well as to support arbitrary query line segments.

Problem 6.10. Given a geometric graph $P = (V, E)$, construct a data structure that can answer the following queries: for a line segment pq in the plane,

1. compute $\min_\pi d_F(\pi, pq)$ and
2. report $\arg\min_\pi d_F(\pi, pq)$,

where π ranges over all paths between two vertices in P.

For distance queries, we closely follow the work of Gudmundsson et al. [127], which in turn follows the approach of Driemel and Har-Peled [95]. The latter appears at first to be devoted to a rather different problem, but turns out to be very helpful in the map-matching setting.

Data structure for distance queries with fixed path endpoints. We can immediately use the approach of Section 6.1 on arbitrary segments. Suppose the query is a pair of vertices $u, v \in V$ and a segment $pq \subset \mathbb{R}^2$. Then we can find a 3-approximation to $\min_\pi d_F(\pi, pq)$, where $\pi : u \rightsquigarrow v$. To achieve that, we just need to define $D'_{s'} = d_F(us' \circ s'v, pq)$ and let t be the point on pq aligned with s' under this matching.

We can directly use the following statement [127, Lemma 15], which closely mimics the approach of Driemel and Har-Peled [95, Lemma 5.8]:

Lemma 6.11. *Let* $u, v \in V$ *be a fixed pair of vertices. Let* $\varepsilon > 0$ *and* $\chi = {}^{1}\!/\!_{\varepsilon^2} \log {}^{1}\!/\!_{\varepsilon}$. *In* $O(\chi^2 n \log n)$ *time and using* $O(\chi^2)$ *space, one can construct a data structure that, given a query segment* pq *in the plane, returns in* $O(1)$ *time a* $(1+\varepsilon)$*-approximation to* $\min_\pi d_{\mathrm{F}}(\pi, pq)$, *where* $\pi : u \rightsquigarrow v$.

The idea behind this lemma is to construct an exponential grid around both fixed vertices, so that the grid is denser close to the vertices. There is an upper bound and a lower bound on how far the grid goes, which is based on ε and $\min_\pi d_{\mathrm{F}}(\pi, uv)$. If the segment pq is closer to uv than the smallest grid cell, then taking $\min_\pi d_{\mathrm{F}}(\pi, uv)$ gives us a good approximation; if the segment is very far, then $d_{\mathrm{F}}(pq, uv)$ dominates. Otherwise, we are guaranteed that there are grid points p' and q' that match p and q closely with respect to u and v. We can simply precompute $\min_\pi d_{\mathrm{F}}(\pi, p'q')$ for all pairs of points p' and q' and return an appropriate value in constant time when given a query. See Figure 6.3.

In order to improve the approximation ratio to $1+\varepsilon$, we use Lemma 16 by Gudmundsson et al. [127], substituting our data structure of Theorem 6.9 for their data structure of Lemma 14. The argument is the same: we can construct a grid around each graph vertex and precompute the distances for all pairs of grid vertices for each transit pair; and we can store that in the data structure of Theorem 6.9. At query time, when testing each transit vertex s, we subsample the relevant part of segment pq with $O({}^{1}\!/\!_{\varepsilon})$ points to find an optimal spot that should align with s. We use the data structure of Theorem 6.9 to make sure we do not need to sample too many points. Plugging in our time and space bounds, we get the following lemma.

Lemma 6.12. *Let* $\varepsilon > 0$ *and* $\chi = {}^{1}\!/\!_{\varepsilon^2} \log {}^{1}\!/\!_{\varepsilon}$. *In time* $O(\tau \chi^2 n^{5/2} \log n)$ *and using* $O(\tau \chi^2 n \sqrt{n})$ *space, we can construct a data structure that, given a query segment* pq *in the plane and a pair of vertices* $u, v \in V$, *returns in* $O({}^{\tau}\!/\!_{\varepsilon} \sqrt{n})$ *time a* $(1+\varepsilon)$*-approximation to* $\min_\pi d_{\mathrm{F}}(\pi, pq)$, *where* $\pi : u \rightsquigarrow v$.

Reporting a path. Next we discuss the modifications needed to report a curve that realises the $(1+\varepsilon)$-approximate distance. We can perform an approximate distance query first. Once we have the distance, we can find the transit pairs that realise it; with these pairs, we can store the next vertex on the optimal path. We can then repeat these queries with the resulting new pairs. We need to show how to do this sequence of queries; and we need to show that this preserves consistency, i.e. that an approximate route for a subpath is also approximate for the complete path.

Recall that when computing the $(1+\varepsilon)$-approximation, we consider a ball of a certain radius around a transit vertex, and we take $\mathcal{O}(1/\varepsilon)$ sample points on the query segment pq inside the ball, to test the Fréchet alignment with the transit vertex. The first modification is that we impose fixed coordinates for the sample points: they have to be located at points that are $\mathcal{O}(c/\varepsilon)$ away from a fixed point on the line containing pq for some natural c.

Next, we describe the necessary modifications to the data structure of Lemma 6.12. With each transit pair and for each pair of grid points, in addition to the Fréchet distance, we also store the first vertex on the optimal path, so $u' \in V$ such that for $\pi' = \arg\min_{\pi} d_{\mathrm{F}}(\pi, pr)$, we have $\pi' : u \longrightarrow u' \rightsquigarrow s$. (Here r is the point on pq that maps to s.)

The query proceeds as follows. First, we perform the distance query for pq and record the optimal transit vertex s. Find the point r among the $\mathcal{O}(1/\varepsilon)$ samples on pq that aligns with s. Query the pairs (u, s) and (s, v) with pr and rq, respectively, and retrieve the stored adjacent vertices u' and v'. Again, find the optimal alignment points on pr and rq; find pairs (u'', s) and (s, v''); repeat until the complete path is reported. In the special case when $s = v$ or $s = u$, only one sequence of queries has to be performed. If u and v are both in a leaf, we can proceed as if $s = v$. See Figure 6.3.

Suppose the transit vertex s is stored in some S_i. If the optimal path leaves P_i, then it is possible that u' is not in P_i, and so the pair (u', s) is not stored in node i. However, then u' must be in some separator separating P_i from a different subgraph P_j. Furthermore, note that the separator in question must be on the path from i to the root. Thus, we can go up until we find $u' \in S_j$ for some $j < i$. We can continue the procedure, now for the transit pair (s, u'), finding some s' so that the path is of the shape $s \longrightarrow s' \rightsquigarrow u'$. See Figures 6.2 and 6.3.

We briefly analyse the time and space bounds. The relevant vertex can be obtained from the free-space diagram when computing the Fréchet

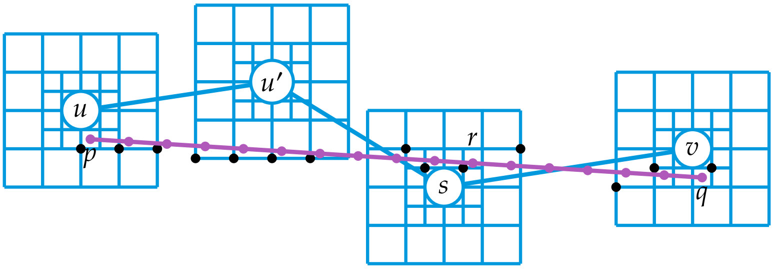


Figure 6.3. A query trajectory pq is shown in purple, and the reported path in the graph is shown in blue. We sample points on pq at regular distance and snap them to the exponential grid around the graph vertices. Once we find r on pq that aligns with the transit vertex s, we can query the pair (u, s) with the (snapped) segment pr to find the next vertex u'.

distance. As we store constant extra information, the preprocessing and space bounds are unchanged. For the query time, in addition to the distance query, we report a path of length ℓ. To find each next vertex, we find the correct transit pair in constant time, then test $O(^1/_\varepsilon)$ alignment options. We may have to go up the tree; however, as we never go down the tree, that traversal happens only once per query. Therefore, the extra time needed to report the path with ℓ vertices is $O(\log n + {}^\ell/_\varepsilon)$. It remains to show that the reported path indeed corresponds to a $(1 + \varepsilon)$-approximation.

Lemma 6.13. *For query pq, if $\pi = \arg\min_{\pi^*} d_F(\pi^*, pq)$ has the shape $\pi : u \longrightarrow u' \rightsquigarrow s$, and u' is aligned to some $p' \in pq$ under the Fréchet alignment, then $\pi' = \arg\min_{\pi^*} d_F(\pi^*, p'q)$ of the shape $\pi' : u' \rightsquigarrow s$ is a subpath of π.*

Proof. Without loss of generality, we can assume that s is a transit vertex. If the distances were computed exactly, the statement would clearly hold. We need to show that the sampling and the grid do not introduce inconsistencies.

Recall that the sample points are placed on the line segment independently from context. Therefore, the location of sample points is the same on pq and $p'q$. Furthermore, we always snap these original sample points to the grid, and the grid does not depend on the path. Therefore, we can view pq as a sequence of grid points that all possible sample points would snap to; and $p'q$ then snaps to a subsequence of those grid points. For the pairs of grid points, the distances are computed directly. Therefore, we do not introduce any additional error, compared to a distance query, and so the reported path corresponds to a $(1+\varepsilon)$-approximation. □

Data structure for finding the path endpoints. So far, we only required that P is τ-lanky. For the next data structure, we also need P to be a t-spanner. Recall that $d_P(u,v)$ denotes the shortest path distance in the graph between the vertices u and v. If $P=(V,E)$ is a t-spanner, then for any $u,v \in V$, we have $d_P(u,v) \leq t \cdot \|u-v\|$. To solve Problem 6.10, we need one more data structure. Lemma 6.12 still requires us to pick vertices u and v to check the paths $\pi : u \rightsquigarrow v$. We need to be able to select a subset of candidate vertices, so that we can obtain a $(1+\varepsilon)$-approximation, but the subset is still small enough. To that aim, we perform the same procedure as Gudmundsson et al. [127], but get different bounds.

In particular, we run Gonzalez's k-centre clustering algorithm [118] for $k=n$ on the vertices of the graph $P=(V,E)$ using the distance d_P. In short, the algorithm selects cluster centres from V iteratively, starting with a random one; and each following one is the furthest away from any other centre. Let c_1 be the first (random) centre. Define the *radius* of a clustering to be the maximum distance from any vertex to its closest centre. Denote $C_i = \{c_1,\dots,c_i\}$. Then for all $2 \leq i \leq n$, we compute

$$c_i = \arg\max_{v \in V} \min_{c \in C_{i-1}} d_P(v,c)\,, \qquad r_i = \max_{v \in V} \min_{c \in C_i} d_P(v,c)\,.$$

We obtain a sequence $\langle (C_1,r_1),\dots,(C_n,r_n)\rangle$, where $r_n = 0$, since all vertices are centres. Using this sequence, we can show the following lemma.

Lemma 6.14. *Let $P=(V,E)$ be a t-spanner and let S be a square in the plane with side length $2r$. Then there exists a set of vertices $T \subseteq V$ satisfying two properties:*

1. *$|T| = O((t/\varepsilon)^2)$, and*
2. *for all $v \in V \cap S$, there is $z \in T$ such that $d_P(v,z) \leq \varepsilon r$.*

Proof. Let i be the index so that $r_i \geq \varepsilon r$ and $r_{i+1} < \varepsilon r$. This is always possible, since $r_n = 0$. Take S' to be the square concentric with S, but with the side length of $4r$, and let $T = C_{i+1} \cap S'$. The second property is immediately satisfied, since any vertex, including those in S, is closer than εr to some centre; and choosing S' this way ensures that we cannot exclude any relevant centres, since $\varepsilon < 1$.

To see that the first property is true, consider C_i. Due to the sequence of picking centres, we know that any two vertices c_j and c_ℓ with $j < \ell$ in T are far apart, i.e. $d_P(c_j, c_\ell) \geq \varepsilon r$. If this were not true, then c_ℓ would not have been chosen as a centre, since there still are vertices in V that are further than εr away from any centre. But then we know

$$\varepsilon r \leq d_P(c_j, c_\ell) \leq t \cdot \|c_j - c_\ell\|,$$

so any two points are at least ${}^{\varepsilon}/{}_{t} \cdot r$ apart in the plane. In a square with side length $4r$, we can only pack $O(({}^{t}/{}_{\varepsilon})^2)$ of these vertices. C_{i+1} has only one more vertex, so the first property holds for T, as well. □

Gudmundsson et al. [127] show how to construct a data structure based on their version of Lemma 6.14. The proof is exactly the same; only the bounds change.

Lemma 6.15. *Let $P = (V, E)$ be a t-spanner, and let $0 \leq \varepsilon \leq 1$. In $O(n^2 \log n)$ time and using $O(n \log n)$ space, we can construct a data structure that, given a query square S in the plane with side length $2r$, returns a set of vertices T satisfying Lemma 6.14 in time $O(\log n + ({}^{t}/{}_{\varepsilon})^2)$.*

The main theorem of this section follows using our data structures.

Theorem 6.16. *Given a τ-lanky t-spanner of complexity n, we can construct the data structure for Problem 6.10 in time $O(\tau \varepsilon^{-4} \log^2({}^{1}/{}_{\varepsilon}) n^{5/2} \log n)$ and using $O(\tau \varepsilon^{-4} \log^2({}^{1}/{}_{\varepsilon}) n \sqrt{n})$ space, so the distance queries can be answered in time $O(\tau t^8 \varepsilon^{-9} \sqrt{n} \log n (\log n + {}^{\tau}/{}_{\varepsilon}))$; and the reporting queries for a path of length ℓ can be answered in $O({}^{\ell}/{}_{\varepsilon})$ additional time.*

Proof. The changes we made for reporting do not affect Lemma 6.15, so the proof of Gudmundsson et al. [127] applies. We only discuss the time bounds. Preprocessing simply consists of building the data structures for Lemmas 6.12 and 6.15. For the query time, consider first the decision version of the algorithm. We query the data structure of Lemma 6.15 twice; and then for every pair of possible matching vertices, we query the data structure of Lemma 6.12. The second step dominates, taking $O(\tau t^4 \varepsilon^{-5} \sqrt{n})$ time.

For the optimisation version, we use parametric search with $N_P = \sqrt{n}$ parallel processors. The sequential version runs in the same time as the decision version, so $T_S = \mathcal{O}(\tau t^4 \varepsilon^{-5}\sqrt{n})$. In the parallel version, querying the distance data structure can be done with $\sqrt{n}$ processors, each performing $\mathcal{O}(\tau/\varepsilon)$ amount of work, then combining the values to find the minimum in $\mathcal{O}(\log n)$ time. Thus, $T_P = \mathcal{O}((t/\varepsilon)^4 \cdot (\tau/\varepsilon + \log n))$. The time for the optimisation version is now $\mathcal{O}(N_P T_P + T_P T_S \log N_P)$. This amounts to $\mathcal{O}(\tau t^8 \varepsilon^{-9}\sqrt{n}\log n(\log n + \tau/\varepsilon))$.

For the reporting query, perform the distance query and record the optimal path endpoints; then, as we discussed, it costs extra $\mathcal{O}(\ell/\varepsilon)$ time to report a path of length ℓ. $\square$

6.3 General Map-Matching Queries

In this section, we generalise the problem again to handle a polygonal curve rather than a line segment as a query. The procedure is very similar; however, we want to make sure that the Fréchet alignment between a query curve and a path can align vertices of the query to points on graph edges, and not just to graph vertices. To that effect, we need to extend Lemma 6.14 so we can sample a small number of points on graph edges.

Gudmundsson et al. [127] use c-packedness again; however, in our setting, the t-spanner property is not sufficient, as it does not give us guarantees about the graph distance between points on the edges. Here we require the graph to also be λ-low density.

Lemma 6.17. *Let $P = (V, E)$ be a λ-low-density t-spanner, let $F = \{f \in \mathbb{R}^2 \mid f \in e, e \in E\}$, and let S be a square in the plane with side length $2r$. Then there exists a set of points $T \subseteq F$ satisfying two properties:*

1. *$|T| = \mathcal{O}(t^2/\varepsilon^2 + \lambda/\varepsilon^3)$, and*
2. *for all $p \in F \cap S$, there is $z \in T$ such that $d_P(p, z) \leq \varepsilon r$.*

Proof. We can use Lemma 6.14 with $\varepsilon' = \varepsilon/2$ to obtain the set T_1 of size $\mathcal{O}((t/\varepsilon')^2)$ so that for all $v \in V \cap S$, $d_P(p, z) \leq \varepsilon' r$ for some $z \in T_1$. Define $E_r \subseteq E$ to contain the edges of length at least εr. Let S' be a square concentric with S but with the side length $4r$. For each $e \in E_r$, choose $\mathcal{O}(1/\varepsilon)$ evenly spaced points on $e \cap S'$ with the distance between them of at most εr. Let T_2 be the set of all such points, and assign $T = T_1 \cup T_2$.

We first show that the first property holds. For T_2, we need to bound the size of $E_r \cap S'$. As P is λ-low density, we know that there are at most λ edges of length at least εr intersecting any disk of diameter εr, and every edge has $O(1/\varepsilon)$ sample points. We can cover S' with $O((1/\varepsilon)^2)$ such disks, so $|T_2| = O(\lambda(1/\varepsilon)^3)$. Therefore, $|T| = O(t^2/\varepsilon^2 + \lambda/\varepsilon^3)$.

Now consider the second property. Note that $V \subset F$. For any $v \in V \cap S$, we immediately conclude that the property holds by Lemma 6.14. For any $p \in e \cap S, e \in E$ with $|e| \leq \varepsilon r$, note that both endpoints of e lie in S'. So there is a vertex $v \in V \cap S'$ so that $d_P(p, v) \leq \varepsilon' r$, and by Lemma 6.14, $d_P(v, z) \leq \varepsilon' r$ for some $z \in T_1$. Therefore, $d_P(p, z) \leq 2\varepsilon' r = \varepsilon r$. Finally, for any $p \in e \cap S$, $e \in E_r$, it is clear that there is a point not further than εr in T_2. □

We can build a data structure just as Gudmundsson et al. [127]. We need to show that it is possible to do so in our setting. We start by stating a definition of λ-low density in R^3.

Definition 6.18. A set of objects in $\mathbb{R}^3$ is k-low density if, for every axis-parallel cube H_r with side length r, there are at most k objects of size at least r that intersect H_r. The size of an object is the side length of its smallest axis-parallel enclosing cube.

Definition 6.19. Given a segment $e \subset \mathbb{R}^2$ and $0 < \varepsilon < 1$, define

$$\text{trough}(e, \varepsilon) = \{(x, y, z) \in \mathbb{R}^3 \mid d((x, y), e) \leq 4z \leq 8|e|/\varepsilon\},$$

where $d((x, y), e)$ is the distance from (x, y) to the closest point on e.

A trough is a three-dimensional object consisting of two half-cones and a triangular prism. Any z-slice can be seen as a z-neighbourhood of e. We show the following lemma.

Lemma 6.20. *Let $P = (V, E)$ be λ-low density, and let $0 < \varepsilon < 1$. The set $\{\text{trough}(e, \varepsilon) \mid e \in E\}$ is k-low density for $k = O(\lambda/\varepsilon^2)$.*

Proof. We first bound the size of trough(e, ε). Let $(x, y, z) \in \text{trough}(e, \varepsilon)$. Note that $0 \leq z \leq 2|e|/\varepsilon$. Furthermore, $d((x, y), e) \leq 8|e|/\varepsilon$, so (x, y) must lie inside a disk centred at the midpoint of e with radius $9|e|/\varepsilon$. Thus (x, y, z) lies inside a cube with side length $18|e|/\varepsilon$, which bounds the size of trough(e, ε).

Let H_r be an axis-parallel cube with side length r, and let its smallest z-coordinate be $z_{\min} \geq 0$. Suppose trough(e, ε) of size at least r intersects H_r, and let (x, y, z) be a point in the intersection. Let h be the projection of the centre of H_r onto the plane $z = 0$. Then

$$d(h, e) \leq d(h, (x, y)) + d((x, y), e) \leq r + 4z \leq 5r + 4z_{\min} ,$$

where the first step follows by the triangle inequality, the second by (x, y, z) lying in the intersection, and the third by $z \leq z_{\min} + r$. Furthermore, the size of trough(e, ε) is at least r and at most ${}^{18|e|}/_{\varepsilon}$, so $r \leq {}^{18|e|}/_{\varepsilon}$; and $z_{\min} \leq {}^{2|e|}/_{\varepsilon}$. Therefore,

$$5r + 4z_{\min} \leq {}^{98|e|}/_{\varepsilon} .$$

So we know $|e| \geq (5r + 4z_{\min}) \cdot {}^{\varepsilon}/_{98}$. By λ-low-density property, any ball with diameter $(5r + 4z_{\min}) \cdot {}^{\varepsilon}/_{98}$ is intersected by at most λ such edges. Consider a disk in $z = 0$ with diameter $2 \cdot (5r + 4z_{\min})$ centred at h. It can be covered by ${}^{c}/_{\varepsilon^2}$ smaller disks for a constant c, so there may be at most $k = {}^{c\lambda}/_{\varepsilon^2}$ edges close enough to h for H_r to intersect their troughs; and so the set of troughs is k-low density. □

Using the range searching data structure for low-density sets by Schwarzkopf and Vleugels [199], we obtain the following result.

Lemma 6.21. *Let $P = (V, E)$ be a λ-low-density t-spanner, let $0 \leq \varepsilon \leq 1$, and let $F = \{f \in \mathbb{R}^2 \mid f \in e, e \in E\}$. In $O(n^2 \log n + {}^{\lambda}/_{\varepsilon^2} \cdot n \log n)$ time and using $O(n \log^2 n + n \cdot {}^{\lambda}/_{\varepsilon^2})$ space, we can construct a data structure that, given a query square S in the plane with side length $2r$, returns a set of vertices T satisfying Lemma 6.17 in time $O(\log^2 n + {}^{t^2}/_{\varepsilon^2} + {}^{\lambda}/_{\varepsilon^3})$.*

Finally, we obtain the main result of the chapter. The proof of Gudmundsson et al. [127, Theorem 3] applies here directly, instead using the data structures of Lemmas 6.12 and 6.21. In short, one can design a decision procedure and then use parametric search. For the decision procedure, we can find a small set of points that each vertex of the query curve may match to, and then construct a directed graph with these as vertices where the weights correspond to the Fréchet distance obtained from our other data structure. For querying that data structure, when considering a point on an edge, we check both its endpoints.

Theorem 6.3. *Suppose we are given a λ-low-density t-spanner of complexity n and a fixed $0 < \varepsilon < 1$. Let $\chi = {}^1/_{\varepsilon^2} \log {}^1/_\varepsilon$ and let $\varphi = ({}^\lambda/_{\varepsilon^3} + {}^{t^2}/_{\varepsilon^2})^2$. In time $O(\lambda\chi^2 n^{5/2} \log n)$ and using $O(\lambda\chi^2 n^{3/2})$ space, we can construct a data structure for Problem 6.2 achieving a $(1 + \varepsilon)$-approximation that performs distance queries in time $O(m\sqrt{n} \log mn \cdot \varphi \cdot {}^\lambda/_\varepsilon \cdot (\log^2 n + \log n \cdot \varphi + \varphi \cdot {}^\lambda/_\varepsilon))$, and answers the reporting queries for a path of length ℓ in $O({}^\ell/_\varepsilon)$ extra time.*

Proof. See the proof by Gudmundsson et al. [127, Theorem 3], but using Lemmas 6.12 and 6.21. We analyse the space and time requirements. For preprocessing and space, we construct the two data structures. For distance queries, we first analyse the decision version.

We query the data structure of Lemma 6.21 m times to obtain the candidate points. Then we construct a directed graph with $O(m \cdot ({}^\lambda/_{\varepsilon^3} + {}^{t^2}/_{\varepsilon^2})^2)$ edges. For each edge, we do a constant number of queries to the data structure of Lemma 6.12, each taking $O(\sqrt{n} \cdot {}^\lambda/_\varepsilon)$ time. Finally, we decide if there is a suitable directed path in the graph. Overall, the decision version takes $O(m\sqrt{n} \cdot {}^\lambda/_\varepsilon \cdot ({}^\lambda/_{\varepsilon^3} + {}^{t^2}/_{\varepsilon^2})^2)$ time.

For the optimisation version, we apply parametric search using $N_P = m\sqrt{n}$ parallel processors. The sequential version runs in the same time as the decision version, so $T_S = O(m\sqrt{n} \cdot {}^\lambda/_\varepsilon \cdot ({}^\lambda/_{\varepsilon^3} + {}^{t^2}/_{\varepsilon^2})^2)$. In the parallel version, the steps for each of m points can be executed in parallel; and finding the weight of an edge by querying the distance data structure can be done with $\sqrt{n}$ processors, each performing $O({}^\lambda/_\varepsilon)$ amount of work, then combining the values to find the minimum in $O(\log n)$ time. Thus, $T_P = O(\log^2 n + ({}^\lambda/_\varepsilon + \log n) \cdot ({}^\lambda/_{\varepsilon^3} + {}^{t^2}/_{\varepsilon^2})^2)$. The time for the optimisation version is now $O(N_P T_P + T_P T_S \log N_P)$. Let $\varphi = ({}^\lambda/_{\varepsilon^3} + {}^{t^2}/_{\varepsilon^2})^2$; then the query time is

$$O\big(m\sqrt{n} \log mn \cdot \varphi \cdot {}^\lambda/_\varepsilon \cdot (\log^2 n + \log n \cdot \varphi + \varphi \cdot {}^\lambda/_\varepsilon)\big)\,.$$

Treating t and λ as constant, we get $O(m\sqrt{n} \log mn \cdot \varepsilon^{-7} \cdot (\log^2 n + \varepsilon^{-6} \log n + \varepsilon^{-7}))$; and treating also ε as a constant, the query time becomes $O(m\sqrt{n} \log mn \log^2 n)$.

Finally, for the reporting query, we first run the distance query and record the path in the graph, as well as the edges aligned with the query curve vertices, and record the optimal order of endpoints of these edges. Now we can simply perform the individual segment reporting queries, as before, costing us extra $O({}^\ell/_\varepsilon)$ time for a path of length ℓ. □

6.4 Conclusions

In this chapter, we have considered the problem of matching an unrestricted trajectory to a realistic map under the Fréchet distance, having preprocessed the map to achieve faster queries. We have shown how to compute the approximate distance to the closest path in the map and how to report the corresponding path. We believe this approach has a very natural set of graph assumptions and reasonable running times. The running times depend on t and λ, but we do not need to compute these parameters to execute the algorithm. However, we need to set ε before building the data structure; while not a significant issue, it is a minor obstacle in face of practicality, so it would be interesting to resolve it—in particular, the solution then cannot use exponential grids as we do. In future work, it would be interesting to see if one could change the machinery for matching the endpoints to lower the assumptions on the graph even further. Of course, it is also possible that there is an equally reasonable set of assumptions that would make the procedure faster.

CHAPTER 7

Segment Visibility Counting Queries in Polygons

In this final chapter, we turn our attention to the following problem. Let P be a simple polygon with n vertices, and let A be a set of m points or line segments inside P. We develop efficient data structures for *visibility counting queries* in which we wish to report the number of objects from A visible to some (constant-complexity) query object Q. An object X in A is *visible* from Q if there is a line segment connecting X and Q contained in P; other objects in A do not block visibility. We focus on the case when Q is a point or a line segment. We aim to obtain fast, $O(\text{polylog}\, nm)$, query times, using as little space as possible.

Our work is motivated by problems in movement analysis where we have sets of moving entities, for example, an animal species and their predators, and we wish to determine if there is mutual visibility between the entities of different sets. We also want to quantify 'how much' the sets can see each other. Given measurements at certain points in time, solving the mutual visibility problem between two such times reduces to counting visibility between line segments (for moving entities) and points (for static objects). This visibility counting problem is also of general interest.

Where is uncertainty? At first glance, this problem does not have anything to do with uncertainty; however, there is a natural connection. Suppose you are tracking the movement of two subjects, and you have their trajectories with measurement uncertainty, with some probability distribution associated with each measured location. Rather than moving on an endless plane, they move around obstacles, modelled as a simple polygon. The two subjects can only interact when there is a direct line of sight between them. You wish to know the likelihood of them having interacted with each other.

To suitably approximate the solution to this problem, instead of working directly with the probability distributions, one may instead sample them, getting indecisive points with a uniform distribution. If we sample k times per measurement, then we can connect the two consecutive measurements with k^2 line segments. If we use a probabilistic model between the measurements, we can also sample ℓ points in the middle of the paths, giving us a sequence of $2k\ell$ line segments. We can do the same for both subjects. Then the original question essentially becomes a counting problem: on how many pairs of line segments can the two subjects see each other? We touch upon this particular variation in Section 7.5, as it seems more complicated still than what we discuss here. However, when one of the subjects is static, our approach can be used directly.

Related work. Computing visibility in polygons is a classical problem in computational geometry [116, 186]. Algorithms for efficiently testing visibility between a pair of points, for computing visibility polygons [101, 148, 168], and for constructing visibility graphs [189] have been a topic of study for over thirty years. There is even a host of work on computing visibility on terrains and in other three-dimensional environments [187, Section 33.8]. For many of these problems, the data structure version of the problem has also been considered. In these versions, the polygon is given up front, and the task is to store it so that we can efficiently query whether a pair of points p, q is mutually visible [77, 131, 141], or report the visibility polygon $V(q)$ of q [27]. In particular, when P is a simple polygon with n vertices, the former type of queries can be answered optimally—in $O(\log n)$ time using linear space [141]. The latter type of queries can be answered in $O(\log^2 n + |V(q)|)$ time using $O(n^2)$ space [27]. The visibility polygon itself has complexity $O(n)$ [102].

Visibility polygons inherit structure from the boundaries of P, so the approaches that use it do not transfer to our setting.

Computing the visibility polygon of a line segment has been considered as well. When the polygon modelling the environment is simple, the visibility polygon, called a *weak visibility polygon*, denoted $V(pq)$ for a line segment pq, still has linear complexity, and can be computed in $O(n)$ time [131]. Chen and Wang [81] consider the data structure version of the problem: they describe a linear-space data structure that can be queried in $O(|V(pq)| \log n)$ time, and an $O(n^3)$-space data structure that can be queried in $O(\log n + |V(pq)|)$ time.

Computing the visibility polygon of a line segment pq allows us to answer whether an entity moving along pq can see a particular fixed point r, i.e. there is a time at which the moving entity can see r if and only if r lies inside $V(pq)$. If the point r may also move, it is not necessarily true that the entity can see r if the trajectory of r intersects $V(pq)$. Eades et al. [99] present data structures that can answer such queries efficiently. In particular, they present data structures of size $O(n \log^5 n)$ that can answer such a query in time $O(n^{3/4} \log^3 n)$. They present results even in case the polygon has holes. Aronov et al. [27] show that we can also efficiently maintain the visibility polygon of an entity as it is moving.

Visibility counting queries have been studied before, as well. Bose et al. [34] studied the case where, for a simple polygon and a query point, the number of visible polygon edges is reported. The same problem has been considered for weak visibility from a query segment [67]. For the case of a set of disjoint line segments and a query point, approximation algorithms exist [20, 126, 206]. In contrast to these settings, we wish to count visible line segments with visibility obstructed by a simple polygon (other than the line segments). Closer to our setting is the problem of reporting all pairs of visible points in a simple polygon [30].

Results and organisation. Our goal is to efficiently count the objects, in particular, line segments or points, in a set A that are visible to a query object Q. We denote this count by $C(Q, A)$. Given P, A, and Q, we can easily compute $C(Q, A)$ in optimal $O(n + m \log n)$ time (see Lemma 7.1). We are mostly interested in the data structure version of the problem, in which we are given the polygon P and the set A in advance, and we wish to compute $C(Q, A)$ efficiently once we are given the query object Q. We show that we can indeed answer such queries efficiently, that is, in

Table 7.1. Results in this chapter. • and / denote points and line segments, respectively.

A	Q	data structure		
		space	preprocessing	query
•	•	$O(nm^2)$	$O(nm \log n + nm^2)$	$O(\log nm)$
•	•	$O(n + m^{2+\varepsilon} \log n)$	$O(n + m \log^2 n + m^{2+\varepsilon} \log n)$	$O(\log n \log nm)$
/	•	$O(nm^2)$	$O(nm \log n + nm^2)$	$O(\log nm)$
•	/	$O(n^2 + nm^{2+\varepsilon})$	$O(n^2 \log m + nm^{2+\varepsilon})$	$O(\log n \log nm)$
/	/	$O(n^2 + nm^{2+\varepsilon})$	$O(n^2 \log m + nm^{2+\varepsilon})$	$O(\log n \log nm)$

polylogarithmic time in n and m. The exact query times and the space usage and preprocessing times depend on the type of the query object and the type of objects in A. See Table 7.1 for an overview. Here and in the rest of the chapter, $\varepsilon > 0$ denotes an arbitrarily small constant.

In Section 7.2, we consider the case where the query object is a point. We show how to answer queries efficiently using the arrangement of all (weak) visibility polygons. As Bose et al. [34, Section 6.2] argued, such an arrangement has complexity $\Theta(nm^2)$ in the worst case. We then show that if the objects in A are points, we can do significantly better. We argue that we do not need to construct the visibility polygons of all points in A, avoiding an $O(nm)$ term in the space and preprocessing time. We use a hierarchical decomposition of the polygon and the fact that the visibility of a point $a \in A$ in a subpolygon into another subpolygon is described by a single constant-complexity cone. Aronov et al. [27] also use hierarchical decomposition, but the rest of their approach cannot be efficiently used in our setting, since it uses the cyclic ordering of the vertices of a visibility polygon. We discuss this in detail in Section 7.5.

In Section 7.3, we turn our attention to the case where the query object is a line segment pq and the objects in A are points. One possible solution in this scenario would be to store the visibility polygons for the points in A so that we can count such polygons stabbed by the query segment. However, since these visibility polygons have total complexity $O(nm)$ and the query may have an arbitrary orientation, a solution achieving polylogarithmic query time will likely use at least $\Omega(n^2m^2)$ space [13, 15, 135]. So, we again use an approach that hierarchically decomposes the polygon to limit the space usage. Unfortunately, testing visibility between the points in A and the query segment is more complicated in this case. Moreover, the segment can intersect multiple regions of the decomposition, so we have to avoid double counting. All of this makes the problem significantly harder. We manage to overcome these difficulties using careful geometric arguments and an inclusion–exclusion-style counting scheme. Our result in Table 7.1 saves at least a linear factor compared to an approach based on stabbing visibility polygons. We then show that we can extend these arguments even further and solve the scenario where the objects in A are also line segments. Surprisingly, this does not impact the space or time complexity of the data structure.

Finally, in Section 7.5, we discuss some extensions of our results. In particular, we show that just testing whether the count $C(Q, A)$ is non-zero (i.e. whether Q is visible from any of the objects) is easier, and that we can compute the pairwise visibility of two sets of objects—that is, solve one of the problems that motivated this work—in time subquadratic in the number of objects.

7.1 Preliminaries

In this section, we review some basic tools we use to build our data structures.

Visibility in a simple polygon. Denote the (weak) visibility polygon in a simple polygon P of a point p (resp. line segment pq) by $V(p)$ (resp. $V(pq)$). A *cone* is a subset of the plane that is enclosed by two rays starting at some point p, called the *apex* of the cone; the angle between any two rays in the cone is acute. We refer to the two bounding rays as the *left* and the *right* ray, so that moving clockwise from the left to the right ray traverses the cone. A *subcone* of some cone C is a cone with the same apex as C that is a subset of C. For a segment $rs \subset P$, define the *visibility cone* of a point $p \in P$ *through* rs, denoted $V(p, rs)$, as a region consisting of the rays from p that intersect rs before properly crossing the boundary of P.[1] Define a visibility cone $C(p)$ *into* a subpolygon U for

---

[1] In case $p \in rs$ holds, this definition yields $\mathbb{R}^2$. We handle such cases separately.

$p \in P \setminus U$ as the visibility cone through the diagonal of P that separates U from the subpolygon containing p.

Lemma 7.1. *Let P be a simple polygon with n vertices, and let A be a set of m points or line segments in P. For a point or a line segment Q, we can find $C(Q, A)$ in time $O(n + m \log n)$.*

Proof. If A is a set of points, it suffices to compute the visibility polygon of Q and preprocess it for $O(\log n)$-time point location queries. Both preprocessing steps take linear time [131, 153], and querying takes $O(m \log n)$ time in total. In case A consists of line segments, we can similarly test if one of the endpoints of each segment of A is visible, thus making the segment visible. We also need to count the number of visible segments whose endpoints lie outside $V(Q)$. This can be done in $O(n + m \log n)$ time by computing a sufficiently large bounding box B of $V(Q)$ and constructing an $O(\log n)$-time ray shooting data structure on $B \setminus V(Q)$. This allows us to test if a segment intersects $V(Q)$ in $O(\log n)$ time. The polygon $B \setminus V(Q)$ has only a single hole, so we can connect the boundary of $V(Q)$ to the boundary of B with a line segment rs and cut $B \setminus V(Q)$ along rs to obtain a simple polygon. We can then build a ray shooting structure [141] on this simple polygon, and answer a query by $O(1)$ ray shooting queries. In particular, for any segment in A that does not cross rs, we get the result directly; and for any segment that crosses rs, we detect that the ray hits rs and do a second query on the other side of the cut. Either way, we use $O(1)$ ray shooting queries. ☐

Lemma 7.2. *Given a visibility polygon $V(p) \subseteq P$ for some point $p \in P$ and a line segment $rs \subset P$, either rs and $V(p)$ do not intersect, or their intersection is a line segment.*

Proof. Assume for contradiction that the intersection between $V(p)$ and rs consists of multiple, possibly degenerate, line segments, $S_1, \ldots, S_k$ for some $k \in \mathbb{N}$, $k > 1$. Take some $i \in \{1, \ldots, k\}$ and pick some points q_i and q_{i+1} on consecutive segments S_i and S_{i+1}. Consider the line segments pq_i and pq_{i+1}. By definition of the visibility polygon, these segments are inside P. Since rs is inside P, the segment $q_i q_{i+1}$ is also inside P. Since P is simple, it must then hold that the interior of the triangle T with vertices p, q_i, and q_{i+1} is also inside P. More precisely, T

cannot contain any of the boundary of P. Now consider a line segment pq_o for a point q_o between segments S_i and S_{i+1} on rs. Since its endpoint q_o is outside $V(p)$, the line segment must cross the boundary of P inside T. This contradicts the previous claim that T is empty; thus, it must be that the intersection between $V(p)$ and rs is a line segment if they intersect. □

Corollary 7.3. *The intersection between the line segment* $rs \subset P$ *and the visibility cone* $V(p, rs)$ *for some* $p \in P$ *is either empty or a line segment.*

Proof. The visibility cone intersected with P is by definition a subset of the visibility polygon $V(p)$. Since the intersection between rs and $V(p)$ is either empty or a line segment by Lemma 7.2, the same must hold for $V(p, rs)$. □

Cutting trees. A *cutting tree* [74, 78, 85] is a data structure commonly used for efficient half-plane range queries. Nesting multiple cutting trees in levels allows one to efficiently perform simplex range searching and solve other related queries; we make extensive use of this. See Figure 7.1 for an illustration. We now discuss this data structure in more detail.

Suppose we want to preprocess a set $\mathcal{L}$ of m lines in the plane so that given a query point q, we can count the number of lines below the query point. Let $r \in [1, m]$ be a parameter; then a $(^1/_r)$-*cutting* of $\mathcal{L}$ is a subdivision of the plane with the property that each cell is intersected by at most $^m/_r$ lines [74]. If q lies in a certain cell of the cutting, we know, for all lines that do not cross the cell, whether they are above or below q, and so we can store the count with the cell, or report the lines in a precomputed *canonical subset;* for the lines that cross the cell, we can recurse. The data structure that performs such a query is called a *cutting tree;* it can be constructed in $O(m^{2+\varepsilon})$ time, uses $O(m^{2+\varepsilon})$ space, and supports answering the queries in time $O(\log m)$ for any constant $\varepsilon > 0$. Intuitively, the parameter r here determines the trade-off between the height of the recursion tree and the number of nodes for which a certain line in $\mathcal{L}$ is relevant. If we pick $r = m$, the $(^1/_r)$-cutting of $\mathcal{L}$ is just the arrangement of $\mathcal{L}$. The bounds above are based on picking $r \in O(1)$, so the height of the recursion tree is $O(\log m)$. This approach follows the work of Clarkson [85], with Chazelle [74] obtaining the bounds above by improving the cutting construction.

An obvious benefit of this approach over just constructing the arrangement on $\mathcal{L}$ and doing point location in that arrangement is that using cuttings, we can obtain $O(\log m)$ canonical subsets and perform nested queries on them without an explosion in storage required; the resulting data structure is called a *multilevel cutting tree.* Specifically, we can query with k points and a direction associated with each point (above or below) and return the lines of $\mathcal{L}$ that pass on the correct side (above or below) of all k query points. If we pick $r \in O(1)$ and nest k levels in a k-level cutting tree, we get the same construction time and storage bounds as for a regular cutting tree; but the query time is now $O(\log^k m)$. Chazelle et al. [78] show that if we set $r = n^{\varepsilon/2}$, each level of a multilevel cutting tree is a constant-height tree, so the answer to the query can be represented using only $O(1)$ canonical subsets and the query time is reduced to $O(\log m)$. The space used and the preprocessing time remains $O(m^{2+\varepsilon})$.

Lemma 7.4 ([78])**.** *Let $\mathcal{L}$ be a set of m lines and let k be a constant. Suppose we want to answer the following query: given k points and associated directions (above or below), find the lines in $\mathcal{L}$ that lie on the correct side of all k points. In time $O(m^{2+\varepsilon})$, we can construct a data structure using $O(m^{2+\varepsilon})$ storage that supports such queries. The lines are returned as $O(1)$ canonical subsets, and the query time is $O(\log m)$.*

Dualising the problem in the usual way, we can alternatively report or count points from the set A that lie in a query half-plane; or in the intersection of several half-planes, using a multilevel cutting tree.

Lemma 7.5 ([78])**.** *Let A be a set of m points and let k be a constant. In time $O(m^{2+\varepsilon})$, we can construct a data structure using $O(m^{2+\varepsilon})$ storage that returns $O(1)$ canonical subsets with the points in A that lie in the intersection of the k query half-planes in time $O(\log m)$.*

Lemma 7.6. *Let A be a set of m arbitrary points in P, with each $a \in A$ an apex of some cone C_a. At query time, we get the point q, an apex of a cone C_q. In time $O(m^{2+\varepsilon})$, we can construct a data structure using $O(m^{2+\varepsilon})$ space that returns a representation of the points in $A' \subseteq A$, so that for any $p \in A'$, we have $q \in C_p$ and $p \in C_q$. The points are returned as $O(1)$ canonical subsets, and the query time is $O(\log m)$; they can be counted in the same time.*

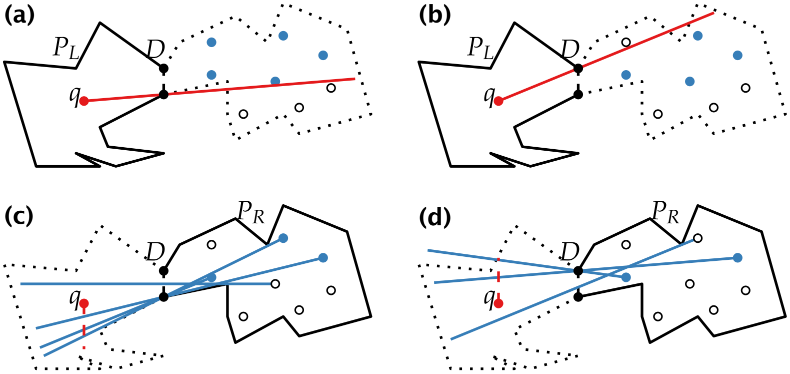


Figure 7.1. A query in a multilevel cutting tree, top left to bottom right. The query point is red; the selected points of A are blue. Black outline shows the relevant part of the polygon. **(a, b)** We select points in A above (resp. below) the right (resp. left) cone boundary of q. **(c, d)** We refine by taking points whose left (resp. right) cone boundary is below (resp. above) q.

Proof. We can construct a four-level cutting tree; the first two levels can select the nodes that represent points from A lying in C_q. Note that to select the points that lie in C_q, we need to perform two consecutive half-plane queries, as C_q is an intersection of two half-planes that meet at point q.[2] We can use Lemma 7.5 to handle these; note that every time we get a constant number of canonical subsets, so any new point location queries can be done in $O(\log m)$ time on each level. After two levels, we get $O(1)$ canonical subsets. The next two levels handle the other condition: select the points whose cones contain q. This can be done by checking that q lies below the upper boundaries of the cones and that q lies above the lower boundaries of the cones. Again, we need to do point location queries on each level and for each canonical subset; we can use Lemma 7.4 to see that we still have a constant number of those. Overall, we do a constant number of point location queries and go down a four-level data structure, where every level is a constant-depth tree. Therefore, the query overall takes $O(\log m)$ time. As stated previously, adding the levels does not increase the storage or the preprocessing time requirements. □

---

[2] The cone C_q can deviate from our definition of a cone, so the two rays can lie on the same supporting line. We can still query twice with the same half-plane.

Lemma 7.7. *Let L be a vertical line and let A be a set of m cones starting left of L and whose left and right rays intersect L. In time $O(m^{2+\varepsilon})$, we can construct two two-level cutting trees for A of total size $O(m^{2+\varepsilon})$, so that for a query segment pq that is fully to the right of L, we can count the cones that contain or intersect pq in $O(\log m)$ time.*

Proof. A cone $C \in A$ partitions the space to the right of L in three regions: the regions above and below C and the region inside C. Segment pq does not intersect C if it is contained in either the top or the bottom region. This is exactly when either both points of pq are above the supporting line of the upper boundary of C, formed by its left ray, or when both are below the supporting line of the lower boundary of C, formed by its right ray. Hence, if we store the supporting lines of the left and right rays of A in two two-level cutting trees, similarly to Lemma 7.6, we can count the cones that are not visible for pq. By storing the total number of cones, we can now determine the number of visible cones. □

Lemma 7.8. *Let $\mathcal{L}$ be a set of m lines and pq a query line segment. We can store $\mathcal{L}$ in a multilevel cutting tree, using $O(m^{2+\varepsilon})$ space and preprocessing time, so that we can count the lines in $\mathcal{L}$ intersected by pq in time $O(\log m)$.*

Proof. Consider the dual version of this problem. The set $\mathcal{L}$ is dualised to the set Λ of m points, and pq becomes a double wedge bounded by two intersecting lines. The key property is that pq intersects a line in the original problem if and only if the corresponding point in the dual problem lies inside the double wedge. Observe that the double wedge is just the union of two cones, and a cone is an intersection of two half-planes; so we can construct a two-level cutting tree using Lemma 7.5 and perform two queries in it, getting $O(1)$ counts from each query. The bounds follow from Lemma 7.5. □

Polygon decomposition. For a simple polygon P on n vertices, we can construct a balanced hierarchical decomposition of P by recursively splitting the polygon into two subpolygons of roughly equal size, using only diagonals (segments between two vertices of the polygon), as shown by Chazelle [73]. The recursion stops when reaching triangles. The decomposition can be computed in $O(n)$ time and stored using $O(n)$ space in a balanced binary tree [73, 75, 130].

Hourglasses and the shortest path data structure. An *hourglass* for two diagonals pq and rs in a simple polygon P is the union of geodesic shortest paths in P from points on pq to points on rs [130]. Such an hourglass is bounded by two diagonals and two inward convex chains. Call one of the diagonals the *left* diagonal and the other the *right* diagonal. By following the hourglass boundary in a clockwise manner, starting from the left diagonal, we visit the *upper convex chain*, the right diagonal, and the *lower convex chain*. If the upper chain and lower chain of an hourglass share vertices, it is *closed*, otherwise it is *open*. We only explicitly use open hourglasses in our constructions.[3] A *visibility glass* is a subset of the hourglass, restricted to the line segments between points on pq and points on rs [99].

Guibas and Hershberger [130, 140] describe a data structure to compute shortest paths in a simple polygon P. They use the polygon decomposition by Chazelle [73] and also store hourglasses between the splitting diagonals of the decomposition. The data structure uses $O(n)$ storage and preprocessing time and can answer the following queries in $O(\log n)$ time:

Segment location query. Given a segment pq, return the two leaf triangles containing p and q in the decomposition and the $O(\log n)$ pairwise disjoint open hourglasses so that the triangles and hourglasses fully cover pq. We call this structure the *polygon cover* of pq.

Shortest path query. Given points $p, q \in P$, return the geodesic shortest path between p and q in P as a set of $O(\log n)$ nodes of the decomposition. The shortest path between p and q is a concatenation of subcurves of the polygonal chains of the appropriate boundaries of the (open or closed) hourglasses in these $O(\log n)$ nodes together with at most $O(\log n)$ segments connecting two consecutive subcurves.

Cone query. Given a point s and a line segment pq in P, return $V(s, pq)$. This can be done by computing the shortest paths from s to p and to q and taking the first segments of each path starting at s to extend them into the bounding rays of a cone.

---

[3] If an hourglass is closed, the two chains are only inward convex from the endpoints of a diagonal until they meet.

7.2 Point Queries

In this section, given a set A of m points in a simple polygon P on n vertices, we count the points of A that are in the visibility polygon of a query point $q \in P$. We present two solutions: (i) an arrangement-based approach that also applies in the case where A contains line segments, which achieves low query time at the cost of large storage and preprocessing time; and (ii) a cutting-tree-based approach with query times slower by a factor of $O(\log n)$, but with much better storage requirements and preprocessing time.

7.2.1 Point Location in an Arrangement

The approach relies on the fact that the number of objects in A visible to a query point q is equal to the number of (weak) visibility polygons of the objects in A stabbed by q. We construct all (weak) visibility polygons of the objects in A and compute the arrangement $\mathcal{A}$ of the edges of these polygons. For each cell C in the arrangement, we store the number of visibility polygons that contain C. Then a point location query for q yields the number of visible objects in A. Computing the visibility polygons takes $O(nm)$ time, and constructing the arrangement using an output-sensitive line segment intersection algorithm takes $O(nm \log nm + |\mathcal{A}|)$ time [76], where $|\mathcal{A}|$ is the number of vertices of $\mathcal{A}$. Building a point location structure on $\mathcal{A}$ for $O(\log|\mathcal{A}|)$-time point location queries takes $O(|\mathcal{A}|)$ time [153]; the space used is $O(|\mathcal{A}|)$. As Bose et al. [34] show, the worst-case complexity of $\mathcal{A}$ is $\Theta(nm^2)$.

Theorem 7.9. *Let P be a simple polygon with n vertices, and let A be a set of m points or line segments in P. In $O(nm^2 + nm \log nm)$ time, we can build a data structure of size $O(nm^2)$ that can report the number of points or segments in A visible from a query point q in $O(\log nm)$ time.*

7.2.2 Hierarchical Decomposition

To design a data structure that uses less storage than that of Section 7.2.1, we observe that if we subdivide the polygon, we can count the visible objects by summing up the number of visible objects in the cells of the subdivision. To efficiently compute these counts, we use the polygon decomposition approach (see Section 7.1). With each split in our

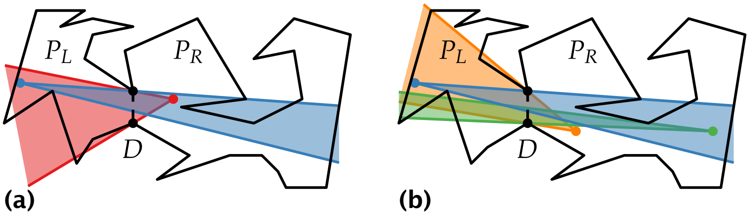


Figure 7.2. Visibility cones (coloured regions) of (coloured) points w.r.t. some diagonal D. **(a)** Blue and red are mutually visible. **(b)** Green and blue cannot see each other, nor can orange and blue.

decomposition, we store data structures that can efficiently count the visible objects in the associated subpolygon.

Cone containment. Let us solve the following problem first. We are given a simple polygon P and a (w.l.o.g.) vertical diagonal D that splits it into two simple polygons P_L and P_R. Furthermore, we are given a set A of m points in P_L. Given a query point q in P_R, we want to count the points in A that see q. We base our approach on the following observation.

Lemma 7.10. *Given a simple polygon P, split into two simple polygons P_L and P_R by a diagonal D between two vertices; and given two points $p \in P_L$ and $q \in P_R$, consider the visibility cones $V(p, D)$ and $V(q, D)$, i.e. the cones from p and q through D into the other subpolygons. Point p sees q in P if and only if $q \in V(p, D)$ and $p \in V(q, D)$.*

Proof. First suppose that $p \in V(q, D)$ and $q \in V(p, D)$. We need to show that p and q see each other, that is, that the line segment pq lies in P. Observe that both p and q lie in $V(p, D)$, and $V(p, D)$ is convex, so pq lies in $V(p, D)$; symmetrically, pq lies in $V(q, D)$. Furthermore, note that since both cones are cones through D, the segment pq must cross D at some point r. Then by construction of $V(p, D)$, the segment pr lies entirely in P_L; similarly, rq lies entirely in P_R. As D also lies in P, we conclude that pq lies in P.

Now suppose that p and q see each other in P. They are on the opposite sides of the diagonal D and the polygon P is simple, so pq must cross D at some point r. As pq lies inside P, clearly, pr lies inside P_L and rq lies inside P_R. Then the visibility cone $V(p, D)$ must include

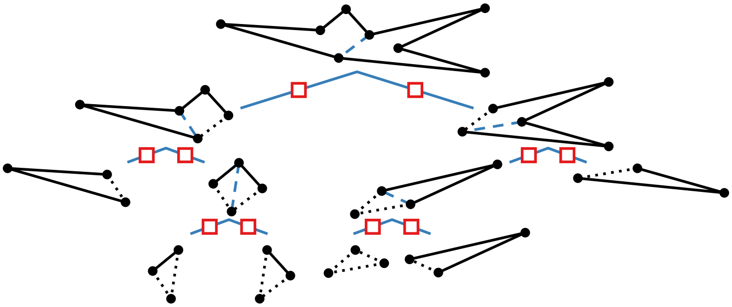


Figure 7.3. Augmented polygon decomposition following the approach by Chazelle [73]. Each node corresponds to the splitting diagonal (blue dashed line). Along the tree edges (blue lines), we store the multilevel cutting tree (red box) for the polygon in the child using the diagonal of the parent.

the ray from p through r, and so q is in $V(p, D)$; symmetrically, p is in $V(q, D)$. □

Lemma 7.10 shows that to count the points in A that see q, it suffices to construct the cones from all points in A through D and the cone from q through D and count the points in A satisfying the condition of Lemma 7.10 (see Figure 7.2). The cones $V(p, D)$ from all $p \in A$ can be precomputed (we shall handle this later), so only the cone $V(q, D)$ needs to be computed at query time. The query of this type can be realised using a multilevel cutting tree (Lemma 7.6).

Decomposition. Let us return to the original problem. To solve it, we can use the balanced polygon decomposition [73] (see Section 7.1). Following Guibas and Hershberger [130, 140], we represent it as a binary tree (see Figure 7.3). Observe that as long as there is some diagonal D separating our query point from a subset of points of A, we can use the approach above.

Every node of the tree is associated with a diagonal, and the two children correspond to the left and the right subpolygons. With each node, we store two data structures described above: one for the query point to the left of the diagonal and one for the query to the right.

The query then proceeds as follows. Suppose the polygon P is triangulated, and the triangles correspond to the leaves in the decomposition. Given a query point q, find the triangle it belongs to; then traverse the tree bottom up. In the leaf, q can see all the points of A that are in the same triangle, so we start with that count. As we proceed up the tree, we query the correct associated data structure—if q is to the right of the diagonal, we want to count the points to the left of the diagonal in the current subpolygon that see q. It is easy to see that this way we end up with the total number of points in A that see q, since the subsets of A that we count are disjoint as we move up the tree and cover the entire set A.

Theorem 7.11. *Let P be a simple polygon with n vertices, and let A be a set of m points inside P. In $O(n + m^{2+\varepsilon} \log n + m \log^2 n)$ time, we can build a data structure of size $O(n + m^{2+\varepsilon} \log n)$ that can report the number of points from A visible from a query point q in $O(\log n \log m + \log^2 n)$ time.*

Proof. The correctness follows from the considerations above; it remains to analyse the time and storage requirements. For the query time, we do point location of the query point q in the triangulation of P and make a single pass up the decomposition tree, making queries in the associated multilevel cutting trees. Clearly, the height of the tree is $O(\log n)$. At every level of the decomposition tree, we need to construct the visibility cone from the query point to the current diagonal; this can be done in $O(\log n)$ time with a cone query (see Section 7.1). Then we need to query the associated data structure, except at the leaf, where we simply fetch the count. The query then takes time $O(\log n \log m + \log^2 n)$. For the storage requirements, we need to store the associated data structures on in total m points at every level of the tree, as well as a single copy of the shortest path data structure, yielding overall $O(n + m^{2+\varepsilon} \log n)$ storage. Finally, we analyse the preprocessing time. Triangulating a simple polygon takes $O(n)$ time [75]. Constructing the decomposition can be done in additional $O(n)$ time [130]. Constructing the associated data structures takes time $O(m^{2+\varepsilon})$ per level, so $O(m^{2+\varepsilon} \log n)$ overall, after determining the visibility cones for the points of A to all the relevant diagonals, which can be done in time $O(m \log^2 n)$, as each point of A occurs a constant number of times per level of the decomposition, and constructing the cone takes $O(\log n)$ time. Overall we need $O(n + m^{2+\varepsilon} \log n + m \log^2 n)$ time. □

Observation 7.12. *While this approach uses many of the ideas needed to tackle the setting with segment queries, Lemma 7.10 does not apply—see Figure 7.4.*

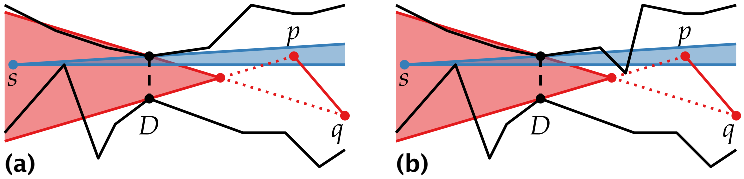


Figure 7.4. **(a)** For the cone that describes visibility of pq through D, Lemma 7.10 does not hold—there can be visibility without visibility between the apices of the cones. **(b)** The segment pq intersects the cone of s, and s is in the cone of pq, but they cannot see each other, so testing intersection between the objects and the cones also does not work directly.

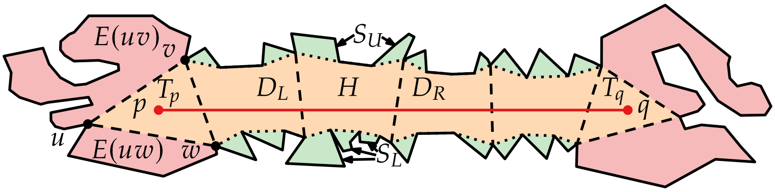


Figure 7.5. Partitioning of the polygon based on the polygon cover of pq. The hourglasses and triangles are shown in orange; the side polygons in green; and the end polygons in red.

7.3 Segment Queries

In this section, we are given a simple polygon P and a set A of stationary entities (points) in P. We construct a data structure to count the points in A that see a query segment pq. We cannot reuse the approach using arrangements from Section 7.2.1, as the query pq may intersect $\Omega(n)$ arrangement cells. In addition, the hierarchical decomposition approach in Section 7.2 does not carry over directly to the setting with segment query objects. Thus, we construct a new data structure using the insights of the hierarchical decomposition. We first show a high-level overview of our argument where, given pq, we decompose the polygon P into three types of regions. Then we show how to count the points visible to pq per type.

High-level overview. We use the data structure by Guibas and Hershberger [130] (abbreviated GHDS) on P as the basis and augment the elements of the GHDS with data structures that allow us to perform the queries. Recall from Section 7.1 that we can obtain the polygon cover for a query segment pq using the GHDS, consisting of $O(\log n)$ hourglasses and two triangles T_p, T_q containing p and q. For a given query pq, the polygon cover partitions P into regions of three types (see Figure 7.5):

1. Hourglasses and triangles T_p, T_q intersecting pq and containing p or q, respectively.
2. Side polygons, each incident to the upper or lower chain of an hourglass.
3. End polygons that are adjacent to T_p and T_q.

For each type, we now count the points in A in a region of that type that see pq. First, we review how to construct the relevant visibility cones.

Lemma 7.13. *At preprocessing, for every side polygon or end polygon S bounded by an hourglass H or a triangle T, we can compute*

- *the number of points in $S \cap A$ and*
- *the cones $C(a)$ for $a \in S \cap A$ into H or T*

in $O(n \log^2 n + nm \log^2 n)$ total time.

Proof. We do this using the approach of Eades et al. [99]. For completeness, we sketch it here. We first construct the GHDS data structure on P. For any point a and any vertex $v \in P$, this data structure can return the shortest path $\pi(p, v)$ in $O(\log n)$ time, represented as a balanced decomposition. We augment this decomposition so that any vertex x on $\pi(p, v)$ stores whether the two edges incident to x have an acute or an obtuse angle. Then we iterate over all explicit $O(n \log n)$ hourglasses H or triangles T. We describe the procedure for an hourglass H and a side polygon S, the other combinations are symmetric.

We count the number of points in $S \cap A$ as follows. For each $a \in S \cap A$, we compute the shortest paths from a to the diagonals of H in $O(\log n)$ time. We know $a \in S$ if and only if these shortest paths intersect the upper chain π_U. Thus, we identify all the points in $S \cap A$ in $O(m \log n)$ time per (H, S).

For every $a \in S \cap A$, we compute the cone $C(a)$ into H. Denote the subpolygon of S that contains a by S'. This is a subpolygon of P bounded by a single edge (x, y) which is a part of the boundary of H (or of the triangle T). We obtain the shortest paths $\pi(a, x)$ and $\pi(a, y)$

in P as a balanced binary tree in $O(\log n)$ time. These two paths are semi-convex chains, with a 'peak' vertex x' and y', respectively. We can identify these peaks in $O(\log n)$ time as these are the first vertices of $\pi(a, x)$ and $\pi(a, y)$ where the angles change from being obtuse to acute (or vice versa). The cone $C(a)$ into H is defined by the rays from a through x' and y'. Thus, we compute all cones $C(a)$ into H in $O(m \log n)$ time per (H, S). □

Since the construction time of Lemma 7.13 is dominated by the construction time of Theorem 7.26, we henceforth assume that for any hourglass H or triangle T and any side or end polygon S, we have access to the visibility cones from points in $A \cap S$ into H or T.

7.3.1 Counting Points in Triangles and Hourglasses

We count the points in A contained in a triangle or an hourglass that see pq. Triangles are convex, so any point in a triangle can see pq. Similarly, each hourglass is completely traversed by pq, so every point inside an hourglass can see pq. At preprocessing, for every region in the GHDS (a triangle T or an hourglass H), we count the points from A in the region. Specifically, we construct a half-plane range query data structure in $O(m^{2+\varepsilon})$ time and, for a triangle T, count the points in $A \cap T$ with three consecutive half-plane range queries. The total complexity of the hourglasses in the GHDS is $O(n \log^3 n)$ [99, Lemma 7]. We triangulate each hourglass H to compute $|A \cap H|$ in $O(n \log^3 n \log m)$ total time. Given pq, we compute the sum over all $O(\log n)$ hourglasses and $O(1)$ triangles in $O(\log n)$ time.

Lemma 7.14. *We can construct an $O(n)$-size data structure in $O(m^{2+\varepsilon} + n \log^3 n \log m)$ time, so that we can count all points in hourglasses and triangles that see pq in $O(\log n)$ time.*

7.3.2 Counting Points in Side Polygons

The points in side polygon do not always see pq. Let H be an hourglass with two diagonals D_L, D_R and two chains π_U, π_L. Let S_U be the side polygon bounded by the upper chain π_U; the analysis for S_L bounded by the lower chain π_L is symmetrical. For a point $a \in A \cap S_U$ to see pq, its visibility cone $C(a)$ into $P \setminus S_U$ cannot be empty. Furthermore, we discern three mutually exclusive types of cones (Figure 7.6):

(a) $C(a)$ intersects the lower chain π_L of H;

(b) $C(a)$ intersects *only* the left diagonal D_L of H; and

(c) $C(a)$ intersects *only* the right diagonal D_R of H.

Cones of Type (a) see pq, since pq separates the upper and the lower chain, and the interior of H does not contain any polygon edges that may block visibility. At preprocessing, we can identify the number of cones of Type (a) for each of the $O(n)$ hourglasses in $O(nm \log^3 n)$ total time by checking for an intersection with the cone boundary for each boundary edge of H. This time is dominated by the construction time in Lemma 7.17. Given a query pq, we need $O(1)$ per each of $O(\log n)$ hourglasses to retrieve the number of cones of Type (a).

Types (b) and (c) are symmetrical; we discuss Type (b) for a fixed side polygon S_U and, without loss of generality, we assume that the directed segment pq crosses D_L before D_R. At preprocessing, we store $A_L \subseteq A \cap S_U$ where for all $a \in A_L$, visibility cone $C(a)$ intersects only D_L; and we store $|A_L|$. To count the visible cones of Type (b) at query time, we subtract from $|A_L|$ the number of elements in A_L that do *not* see pq.

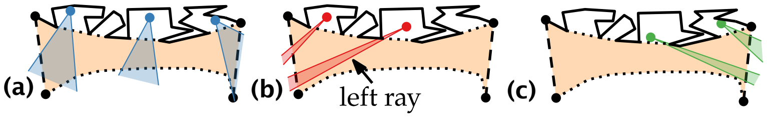


Figure 7.6. The three cases **(a)**, **(b)**, and **(c)** for cones originating from a side polygon S_U.

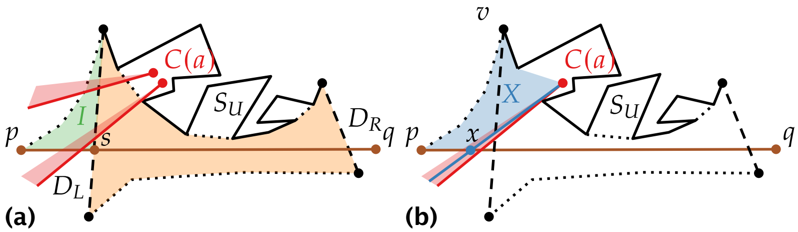


Figure 7.7. The construction used in the proof of Lemma 7.15. The left rays of each cone are highlighted in red. **(a)** The first direction of the bi-implication considers some area I. **(b)** The second direction of the bi-implication considers some ray r in blue and some area X.

Lemma 7.15. *A point $a \in A_L$ is not visible to pq if and only if the left ray of $C(a)$ intersects the shortest path from p to the top of D_L.*

Proof. Let H be bounded by diagonals D_L and D_R. Let A_L be the set of points in $S_U \cap A$ whose visibility cone $C(a)$ intersects D_L. The proof is illustrated by Figure 7.7.

First assume that $a \in A_L$ does not see pq. Let v be the top vertex of D_L, and let $\pi(p, v)$ be the shortest path from p to v in P. Suppose D_L and pq intersect in point s. Since a does not see pq, this implies that $C(a)$ does not intersect sq. Thus, any ray in $C(a)$ intersects D_L in the segment sv.

Let I be the region bounded by ps, sv, and $\pi(p, v)$. The region I cannot contain any polygon vertices; it lies to the left of the supporting line of D_L; and any ray of $C(a)$ must enter this region via sv. Therefore, any ray of $C(a)$ must leave this region via $\pi(p, v)$ or via ps. If a does not see pq, then no ray in $C(a)$ can intersect ps, and so it must intersect $\pi(p, v)$. It follows that the left ray of $C(a)$ intersects $\pi(p, v)$.

Now assume that $a \in A_L$ sees pq. Whenever a sees pq, there must be a ray $r \in C(a)$ which hits pq in some point x and passes through the interior of P. Consider the region X bounded by $\pi(p, v)$, $\pi(v, a)$, ax, and px. The region X and the left ray of $C(a)$ are separated by the supporting line of r. Thus, the left ray of $C(a)$ cannot intersect $\pi(p, v)$. □

On a high level, we count the points $a \in A_L$ for which the left ray of $C(a)$ intersects the shortest path from p to the top vertex v of D_L as follows. For each hourglass in the GHDS, we store the shortest path map SPM(v) rooted at v [131], and for each edge e in SPM(v), we store the number of points $a \in A_L$ for which the left ray of $C(a)$ intersects e. We construct the SPMs and the cutting trees for the $O(n)$ hourglasses in the GHDS using $O(nm^{2+\varepsilon} + n^2)$ space and $O(nm^{2+\varepsilon} + n^2 \log m)$ time. At query time, we obtain the shortest path $\pi(p, v)$ from p to v as a single segment pu, followed by a path $\pi(u, v)$ in SPM(v) for some vertex u. We count the points $a \in A_L$ where the left ray of $C(a)$ intersects pu in $O(\log m)$ time, and those where the left ray of $C(a)$ intersects $\pi(u, v)$ in $O(\log n)$ time, thus avoiding double counting. Thus, we count the elements in A_L that do *not* see pq in $O(\log nm)$ time per side polygon. Given pq, we apply this strategy for all $O(\log n)$ hourglasses.

Lemma 7.16. *We can construct a data structure using $O(m^{2+\varepsilon} + n)$ space in $O(m^{2+\varepsilon} + n \log m)$ time for each hourglass H so that we can count all points of Type (b) (or Type (c)) that see the query pq in $O(\log mn)$ time.*

Proof. By Lemma 7.15, whenever we want to count the visible points of Type (b), we only need to count the points in A_L and subtract the points whose cones' left ray intersects the shortest path from p to the top vertex of D_L, denoted by v.

For an hourglass H and a side polygon S_U, we count the elements in A_L in $O(m)$ time by checking the left ray of each $C(a)$ in constant time. We store the left rays of cones $C(a)$ for $a \in A_L$ in a multilevel cutting tree in $O(m^{2+\varepsilon})$ time. In addition, we compute a shortest path map SPM_H on the polygon P_L to the left of D_L with v as its root—it can be computed in $O(n)$ time and uses $O(n)$ space [131]. Finally, we augment the shortest path map SPM_H: for every edge e stored in SPM_H, we compute the number of left rays that intersect e in $O(\log m)$ time using our multilevel cutting tree. For every vertex w in SPM_H, we consider the shortest path π between w and v and store the total number of intersections between all edges e in π and the rays that intersect e. This data structure needs $O(m^{2+\varepsilon} + n)$ space and can be constructed in $O(m^{2+\varepsilon} + n \log m)$ time per hourglass.

Given a query segment pq, we now want to count the points $a \in A_L$ for which the left ray of $C(a)$ intersects the shortest path $\pi(p, v)$. Observe that the area X bounded by $\pi(p, v)$ and pv is convex. A ray intersects $\pi(p, v)$ if and only if it intersects X. Moreover, due to convexity, any ray that intersects X must intersect two edges of X (assuming no ray goes through a vertex of X). Then to answer our query, we compute the sum over all the edges e bounding X of the number of rays that intersect e, and halve this sum.

Specifically, $\pi(p, v)$ consists of a convex chain of segments of the shortest path between v and some vertex u, followed by a segment pu. We retrieve the shortest path from u to v in $O(\log n)$ time. Observing this, we do the following.

- Using the multilevel cutting tree, we count the rays that intersect pv in $O(\log m)$ time.
- Using the multilevel cutting tree, we count the rays that intersect pu in $O(\log m)$ time.

- We retrieve the rest of the sum from the augmented SPM$_H$ in $O(1)$ time.

Summing these values in constant time, we answer a query in $O(\log nm)$ time. □

Finally, we construct a data structure that can count the points in $A \cap S_U$ that see the query pq, for any hourglass H in the GHDS and for any side polygon S_U of H. We summarise the steps per type.

(a) Cones of Type (a) intersect the lower chain of an hourglass; they see pq. We count them at preprocessing in $O(nm \log^3 n)$ total time using $O(n)$ total space. Given a query pq, we can report the count in $O(1)$ time per hourglass returned by the GHDS, for the total time of $O(\log n)$.

(b) Cones of Type (b) intersect only the left diagonal of an hourglass. For each of the $O(n)$ hourglasses in the GHDS, we construct a $O(m^{2+\varepsilon} + n)$-space data structure in $O(m^{2+\varepsilon} + n \log m)$ time. Given a query pq, we need $O(\log nm)$ time per hourglass, so $O(\log n \log nm)$ total time, to count the cones of this type.

(c) Cones of Type (c) intersect only the right diagonal of an hourglass. They are symmetric to Type (b).

Lemma 7.17. *We can construct an $O(nm^{2+\varepsilon} + n^2)$-size data structure in time $O(nm^{2+\varepsilon} + n^2 \log m)$, so we can count all points in side polygons that see pq in $O(\log n \log nm)$ time.*

7.3.3 Counting Points in End Polygons

End polygons are bounded by triangles in the decomposition T_p or T_q, containing p or q, respectively. Let $T_p = uvw$; it is incident to at most three end polygons. We describe the data structure for the end polygon $E(uv)$ incident to T_p through the edge uv (see Figure 7.8). All other data structures are symmetrical. We count the points in $A \cap E(uv)$ that see pq. Denote the set of all visibility cones $V(a, uv)$ by C_{uv} over all $a \in A \cap E(uv)$.

Consider the special case where pq is fully contained in a triangle ($T_p = T_q$). A triangle is convex, so we can immediately apply Lemma 7.8. Now assume $T_p \neq T_q$. The query segment pq intersects some boundary of T_p. We construct a data structure under the assumption that pq intersects

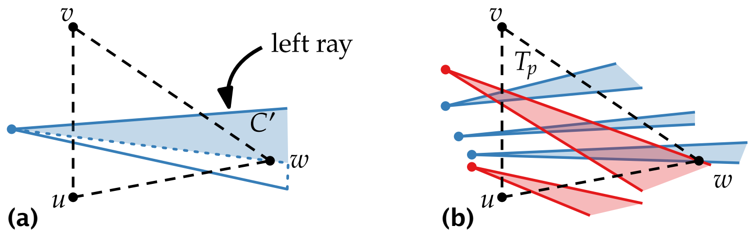


Figure 7.8. **(a)** We replace any cone $C(a) \in \mathcal{C}_{uv}$ that contains w by the cone C' of rays that pass above w. **(b)** We then partition $\mathcal{C}_{uv}$ into $C{\uparrow}(uv)$ (blue) and $C{\downarrow}(uv)$ (red) based on whether the cone passes over or under w.

edge vw of T_p; for pq intersecting uw, we construct a symmetrical structure. So, given a triangle T_p, where pq intersects vw, we want to count the cones $C(a) \in \mathcal{C}_{uv}$ whose apex a sees pq. For brevity, we say that such a cone sees pq, meaning that pq is visible from the cone's apex. Our argument is a multilevel case distinction on $\mathcal{C}_{uv}$. We first partition $\mathcal{C}_{uv}$ during preprocessing (Figure 7.8) into two sets:

1. $C{\downarrow}(uv)$ are the cones in $\mathcal{C}_{uv}$ that pass entirely below the vertex w; and
2. $C{\uparrow}(uv)$ are the cones in $\mathcal{C}_{uv}$ that pass entirely above the vertex w.

We crop the cones that contain w; we now show why this is correct.

Lemma 7.18. *Let $C(a) \in \mathcal{C}_{uv}$ be a cone which contains the vertex w of T. Denote by C' its subcone that passes above w. The apex a sees pq if and only if a sees pq through a ray in C'.*

Proof. Let C'' be the region $C(a) \setminus C'$. Suppose for the sake of contradiction that a see pq but there is no ray $r \in C'$ that hits pq. Then pq must contain some point $s \in C''$. Denote by s^* the intersection between pq and the edge vw, which must exist by construction. Then the segment from s to s^* must intersect C'; the first point of intersection is hit by a ray $r' \in C'$ which realises visibility to pq, which is a contradiction. □

As a consequence of this lemma, we replace at preprocessing any cone $C \in \mathcal{C}_{uv}$ that contains the vertex w with the subcone C' that passes above w.

Counting the cones in $C{\downarrow}(uv)$ that see pq. For this class of cones, we can prove the following statement.

Lemma 7.19. *A cone $C(a) \in C{\downarrow}(uv)$ sees pq if and only if p lies below the supporting line of the left ray of $C(a)$.*

Proof. We assumed earlier that the pq intersects the segment vw of the triangle T_p. Since T_p is convex, a sees pq if and only if pq intersects the left ray of $C(a)$. By construction, the vertex q lies above the left ray of $C(a)$. Hence, the latter occurs if and only if p lies below the left ray of $C(a)$. □

This means we can count the rays in $C{\downarrow}(uv)$ where p lies below the corresponding supporting line of the left ray by storing these rays in a half-space range data structure, which has $O(\log m)$ query time.

Lemma 7.20. *In $O(nm^{2+\varepsilon})$ time, we can construct a data structure of size $O(nm^{2+\varepsilon})$ so that given a query segment pq, a triangle $T = uvw$, and set of cones $C{\downarrow}(uv)$, we can count the cones $C(a) \in C{\downarrow}(uv)$ that see pq in $O(\log m)$ time.*

Counting the cones in $C{\uparrow}(uv)$ that see pq. We partition $C{\uparrow}(uv)$ with an elaborate double case distinction. Observe that this conceptual case distinction can only be made at query time when we have access to $s = pq \cap vw$ and q (Figures 7.8 and 7.9).

(b) $U \subseteq C{\uparrow}(uv)$ are the cones where both boundary rays intersect vs (but not s):
  - $U^* \subseteq U$ are the cones whose apices lie above the supporting line of sq;
  - $U^\square \subseteq U$ is the set $U \setminus U^*$.

(c) $S \subseteq C{\uparrow}(uv)$ are the cones which contain s:
  - $S^{(i)} \subseteq S$ are the cones whose right ray intersects sq;
  - $S^{(ii)} \subseteq S$ is the set $S \setminus S^{(i)}$.

(d) $D \subseteq C{\uparrow}(uv)$ are the cones where both boundary rays intersect sw (but not s):
  - $D^{(i)} \subseteq D$ are the cones whose right ray intersects sq;
  - $D^{(ii)} \subseteq D$ is the set $D \setminus D^{(i)}$.

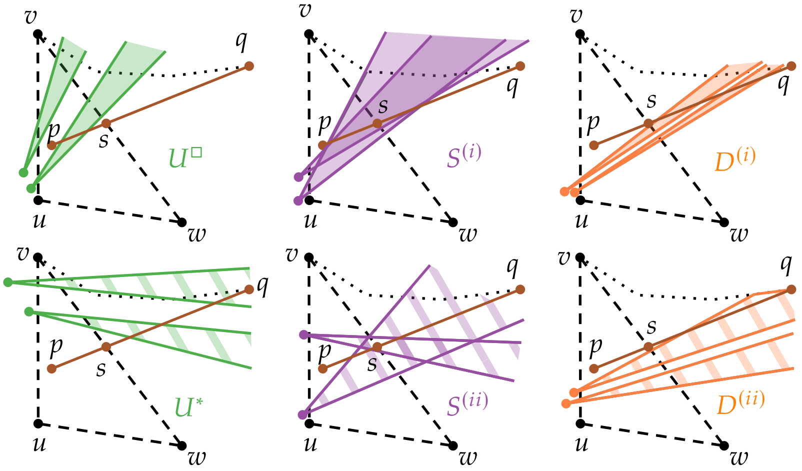


Figure 7.9. We partition $C{\uparrow}(uv)$ into six sets: U^*, $U^\square$, $S^{(i)}$, $S^{(ii)}$, $D^{(i)}$, and $D^{(ii)}$.

Counting the cones in U that see pq. Both U^* and $U^\square$ may contain cones whose apex sees pq. We show how to count these for both sets using a data structure *only* on $C{\uparrow}(uv)$. We show that cones in $U^\square$ are visible if and only if they intersect the segment ps; we can test this in $O(\log m)$ time. We count the cones in U^* whose apices see pq via an inclusion–exclusion argument. Specifically, we observe that all cones in $C{\uparrow}(uv) \setminus U^*$ have one of two mutually exclusive properties:

(i) For all cones in $S^{(i)}$, $D^{(i)}$, and $U^\square$, their right ray intersects $\pi(v,q)$ and lies above s.

(ii) For all cones in $S^{(ii)}$ and $D^{(ii)}$, their right ray lies below s and does not intersect $\pi(v,q)$.

Cones in U^* never have Property (ii). Moreover, they are not visible if and only if their right ray intersects $\pi(v,q)$, i.e. invisible cones have Property (i). Thus, the number of cones in U^* whose apices see pq is equal to $|C{\uparrow}(uv)|$ minus all cones with Property (i) or (ii). We count cones with Property (i) using a shortest path map in $O(\log n)$ time, identically to Section 7.3.2. We count cones with Property (ii) using half-plane range queries in $O(\log m)$ time. We now elaborate on these points further.

Lemma 7.21. *For all $C(a) \in U^\square$, the right ray of $C(a)$ passes over s. Moreover, a sees pq if and only if the right ray of $C(a)$ intersects the segment ps.*

Proof. By definition, the right ray of $C(a)$ passes over s. Assume w.l.o.g. that vw is vertical. Since a lies below the supporting line of pq, the right ray of $C(a)$ must have greater slope than pq. Thus, no rays in $C(a)$ can hit sq. This implies the lemma. □

Using Lemma 7.21, we can count the cones $C(a) \in U^{\Box}$ that see pq as follows. At preprocessing, for every triangle T and for every set $C\!\uparrow\!(uv)$, we construct a multilevel cutting tree on the right rays in $C\!\uparrow\!(uv)$. Given a query pq and the point of intersection s, we identify in $O(\log m)$ time the unique cones of $C\!\uparrow\!(uv)$

1. whose right ray passes over s, or
2. whose right ray intersects the segment ps.

Lemma 7.22. *In $O(nm^{2+\varepsilon})$ time and using $O(nm^{2+\varepsilon})$ space, we can construct a data structure so that given a query segment pq, a triangle T, and a set $C\!\uparrow\!(uv)$, we can count the cones $C(a) \in U^{\Box}$ that see pq in $O(\log m)$ time.*

It remains to identify the cones $C(a) \in U^*$ that see pq. We observe that all cones in $C\!\uparrow\!(uv) \setminus U^*$ have one of two mutually exclusive properties:

(i) For all cones in $S^{(i)}$, $D^{(i)}$, and $U^{\Box}$, their right ray intersects $\pi(v, q)$ (and lies above s).

(ii) For all cones in $S^{(ii)}$ and $D^{(ii)}$, their right ray lies below s (and does not intersect $\pi(v, q)$).

Lemma 7.23. *All cones $C(a) \in U^*$ do not have Property (ii). They have Property (i) if and only if they do not see pq.*

Proof. These cones do not have Property (ii) by definition of the set U. The proof for the second claim is identical to the proof of Lemma 7.15. □

Therefore, the number of cones in U^* visible to pq is $|C\!\uparrow\!(uv)|$ without all cones with Property (i) or (ii).

Lemma 7.24. *In $O(nm^{2+\varepsilon} + n^2 \log m)$ time and using $O(nm^{2+\varepsilon} + nm)$ space, we can construct a data structure so that given a query segment pq, a triangle T, and a set $C\!\uparrow\!(uv)$, we can count the cones $C(a) \in U^*$ that see pq in $O(\log nm)$ time.*

Proof. Our data structure consists of two parts for a given triangle T and a set $C\!\uparrow\!(uv)$: a multilevel cutting tree on the right rays of the cones and an augmented shortest path map of Section 7.3.2. Constructing these for every triangle and set $C\uparrow(uv)$ requires $O(nm^{2+\varepsilon} + nm)$ space and $O(nm^{2+\varepsilon} + n^2 \log m)$ total time.

Given a query pq, we count the cones in $C\uparrow(uv)$ with Property (i) in $O(\log m)$ time using the cutting tree. We also count the cones in $C\uparrow(uv)$ with Property (ii) in $O(\log n)$ time using the augmented shortest path map. Overall, we count the cones in $C(a) \in U^*$ that see pq in $O(\log nm)$ time. □

Counting the cones in S and D that see pq. The apices of all cones in S see pq. We identify these in $O(\log m)$ time using a stabbing query on $C\uparrow(uv)$. For cones in D, we can make a symmetrical case distinction, creating the sets D^* and $D^\square$, and we can handle them through an identical data structure.

Completing the argument. Using Lemmas 7.20, 7.22 and 7.24, we construct a data structure of $O(m^{2+\varepsilon} + n)$ size in $O(m^{2+\varepsilon} + n \log nm)$ time for all $O(n)$ triangles and end polygons. At query time, we retrieve $O(1)$ triangles, with $O(1)$ end polygons each. For every such combination, we query the corresponding data structure in $O(\log nm)$ time.

Lemma 7.25. *We can construct an $O(nm^{2+\varepsilon} + n^2)$-size data structure in time $O(nm^{2+\varepsilon} + n^2 \log m)$, so we can count all points in end polygons that see pq in $O(\log n \log nm)$ time.*

We store all points and visibility cones in the data structures of Lemmas 7.14, 7.17 and 7.25 to show the main result of this section.

Theorem 7.26. *Let P be a simple polygon with n vertices, and let A be a set of m points inside P. In time $O(nm^{2+\varepsilon} + n^2 \log m)$, we can build a data structure of size $O(nm^{2+\varepsilon} + n^2)$ to count the points from A visible from a query segment pq in $O(\log n \log nm)$ time.*

7.4 Segment Query for a Set of Segments

As a natural extension to our GHDS, we consider the case where A is a set of segments and we want to count the segments that see query pq. As we show next, we can reuse the approach of the previous section with some minor additions and answer this query in polylogarithmic time.

A difficulty that arises in this setting is that the segments in A are no longer partitioned by the polygon cover of pq, that is, segments in A may start or end in the polygon cover or pass through the cover. To be able to correctly count the visible cones, we propose instead to count the cones that we *cannot* see and subtract this from the total count. Observe that any line segment that is *not* entirely contained in a single end polygon or side polygon intersects a triangle or an hourglass and thus is visible to pq. Hence, it suffices to count the *invisible* line segments inside the individual side and end polygons and subtract that count from the total.

7.4.1 Using Visibility Glasses

To determine visibility of the segments in A, we use the *visibility glasses*, i.e. the collections of all visibility lines between two line segments (see Figure 7.10 and Section 7.1). Let ac be a segment to the left of a (vertical) diagonal $D = uv$ in P (see Figure 7.10). We now check what ac sees in the subpolygon to the right of the diagonal. To do this, we construct the visibility glass $L(ac, D)$ between ac and the diagonal. Eades et al. [99] show that $L(ac, D)$ is an hourglass defined by some subsegment $or \subseteq ac$ and a subsegment $wx \subseteq D$ (potentially, $o = r$ or $w = x$). We can now compute the lines connecting the opposite endpoints of the visibility glass that still provide visibility, that is, the lines through ox and rw. Note that these lines define the most extreme slopes under which there can still be visibility. These two lines intersect in a single point i. We now consider this point and the lines through it as a new cone that describes the visible region to the right of diagonal D. We call this cone the *visibility glass cone*. Note that the left and the right rays of the cone are actual visibility rays to points o and r on the line segment for points to the right of D.

Lemma 7.27. *Consider a polygon P, split into subpolygons P_L and P_R by a diagonal $D = uv$, and let ac be a line segment in P_L. Let C be the visibility glass cone of ac into P_R through D. If a point $p \in P_R$ sees ac, then $p \in C$.*

Proof. Assume w.l.o.g. that v is above u, and let wx, with x above w, be the part of D inside C. See Figure 7.10. Suppose for a contradiction that p sees ac but is not in C. Let q be a visible point on ac and let L be the line segment connecting p and q. Since D separates P_L from P_R, and L must be inside P to be a visibility line, L must cross D. Suppose w.l.o.g.

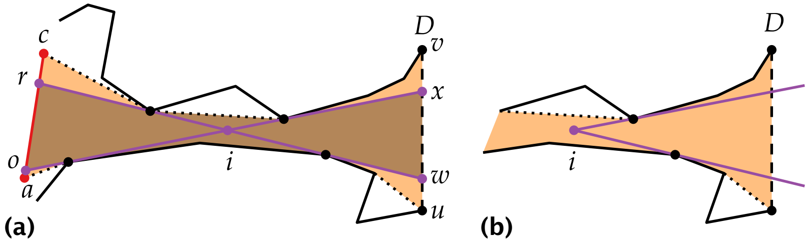


Figure 7.10. (a) The visibility glass (dark region) inside the hourglass (orange region) from segment ac to diagonal $D = uv$. **(b)** The intersection point i and the two rays from i through w and x form a new visibility region in the subpolygon to the right of D.

that L crosses D above x, that is, above the left ray R_L of C. The left ray must intersect a reflex vertex of the upper chain of its associated hourglass H, so there is a region above R_L bounded by the upper chain, R_L, and xv. Since L enters this region, it must also exit the region. There can only be visibility if L is inside P, hence, it must exit the region via the edge bounded by R_L. Therefore, its slope is higher than the slope of R_L. However, R_L is the visibility ray with the highest slope, so by definition of the visibility glass, L then cannot see ac, leading to a contradiction. Thus, p is in C. □

Corollary 7.28. *If p is visible, the ray from p through the apex i of C is a visibility line to ac.*

Lemma 7.27 shows that the visibility glass cones are functionally the same as the visibility cones of points, thus we can reuse our data structures of Section 7.3 for side and end polygons.

Observation 7.29. *If a segment $ac \in A$ cannot see pq, it must be fully contained in a side or an end polygon.*

This follows easily from the fact that if a segment ac is not contained in either an end or a side polygon, then it is in the polygon cover or intersects the boundary of the polygon cover. This then means that ac sees pq. It now suffices to count the segments in the side and the end polygons that are not visible to determine the total number of entities invisible to pq, and thus determine the number of entities in A visible to pq.

Since our data structure of Section 7.3 can already correctly count visible segments from the end and the side polygons, we can simply determine the number of invisible segments by subtracting the number of visible segments from the total number of segments in the end or side polygon.

Theorem 7.30. *Let P be a simple polygon with n vertices, and let A be a set of m segments in P. In time $O(nm^{2+\varepsilon} + n^2 \log m)$, we can build a data structure of size $O(nm^{2+\varepsilon} + n^2)$ that can report the number of segments in A visible from a query segment pq in $O(\log n \log nm)$ time.*

Proof. We can reuse the data structures of Section 7.3 for the side and the end polygon queries, provided we use the visibility glass cones of the segments in A that fall strictly inside the side and the end polygons. Using the data structure by Eades et al. [99], we can compute the visibility glasses to all diagonals in $O(nm \log^4 n + n \log^3 n)$ time and extract the visibility glass cones in constant time per visibility glass. We can then use these cones to build the data structures of Section 7.3. Construction time of the data structure dominates the time required to construct the visibility glass cones, and storage requirements remain the same. In addition, we store the total number of segments for each possible side and end polygon, requiring a total of $O(n)$ extra space.

Given the query, we determine the total number of invisible segments in the side and the end polygons by subtracting the number of reported visible segments from the total number of segments in the side and the end polygons in $O(\log n \log nm)$ time. We then subtract this count from m to arrive at the result. □

7.5 Extensions, Discussion, and Future Work

In this section, we present some natural extensions to our work and discuss possible variations, as well as the obstacles in the way of obtaining results in those settings.

In particular, we extend the approach of Section 7.4 to the case where A contains constant-complexity simple polygons, with the same bounds. We also discuss the approach using visibility polygons by Aronov et al. [27] and point out why it would be inefficient in our setting. Furthermore, we consider a version of the problem where we only want

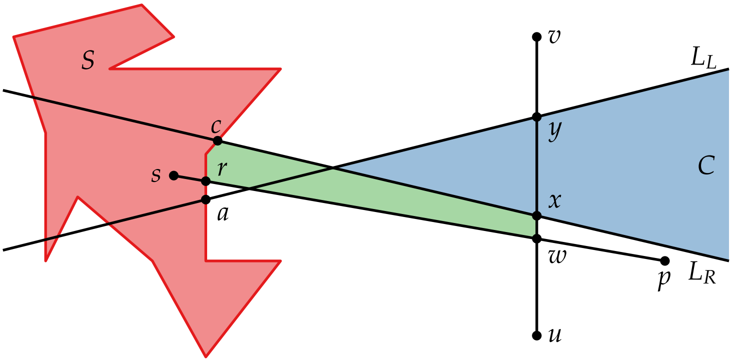


Figure 7.11. The visibility glass cone C for a region S and a diagonal $D = uv$.

to test if at least one object in A sees the query. Finally, we tackle the question of subquadratic counting, where given two sets A and B with m points each, we wish to count the pairs from $A \times B$ that see each other in the simple polygon in time subquadratic in m.

7.5.1 Preprocessing Polygons

Instead of line segments in the set A, we can extend our approach to polygons in A. So consider now the setting where we have a query segment pq and a set of polygons A.

For this extension, we mainly have to show that an equivalent to Lemma 7.27 holds—the rest then easily follows as for line segments in Section 7.4. We first have to define the visibility glass and the visibility glass cone. The visibility glass between a diagonal $D = uv$ and a polygon $S \in A$ is defined as the set of segments aw where $a \in S$ and $w \in D$. Without loss of generality, assume that D is vertical and that it splits P into subpolygons P_L and P_R, left and right of D, respectively, and assume $S \subseteq P_L$.

Consider the segment ay of the visibility glass with the highest slope. In case of ties, take the shortest segment, so the intersection of ay and S consists of only a. Similarly, define cx as the segment with the lowest slope. Let L_L and L_R denote the supporting lines of ay and cx, respectively. The visibility glass cone of S through D is then the cone defined by L_L and L_R that passes through D. We are now ready to prove an equivalent to Lemma 7.27.

Lemma 7.31. *Consider a polygon P, split into subpolygons P_L and P_R by a diagonal $D = uv$ between two vertices u and v, and let S be a simple polygon in P_L. Let C be the visibility glass cone of S into P_R through D. If some point $p \in P_R$ sees S, it must be in C.*

Proof. For a contradiction, assume that there is a point $p \in P_R$ that sees S but lies outside of C. Without loss of generality, suppose that p is below C. Let $s \in S$ be a point that is visible to p, and let $r \in S$ be the point closest to p on the line segment sp—see also Figure 7.11. Let w be the intersection point of sp and D. We first note that s cannot lie above (or on) L_R. If that were the case, then the line segment ws would have a lower slope than L_R and would be in the visibility glass, contradicting the definition of the visibility glass cone. So we can assume that s is below L_R. However, if s is below L_R, then so is r. Now consider the region defined by cx, xw, wr, and the path from c to r along the boundary of S in clockwise direction (green region in Figure 7.11). None of these segments or the path can be intersected by the polygon boundary, so the region is empty. However, in that case, also the line segment cw must be in the visibility glass and has a lower slope, again contradicting the definition of the visibility glass cone. From this contradiction we can conclude that any $p \in P_R$ that sees S must be inside the visibility glass cone C. □

Using this lemma, we can apply the same methods as in Section 7.4 for a set of segments.

7.5.2 Details of Visibility Polygon Computation

We recap the approach of Aronov et al. [27] and indicate why it is not efficient in our case.

They preprocess a simple polygon P with n vertices so that given a query point $q \in P$, they can report the visibility polygon $V(q)$ in time $O(\log^2 n + |V(q)|)$. The data structure uses $O(n^2)$ space and can be constructed in $O(n^2 \log n)$ time. Similar to our approach in Section 7.2.2, they use a hierarchical decomposition of the polygon.

For all polygons P split into two subpolygons P_L and P_R by a diagonal D, assuming $q \in P_R$, they compute the partial visibility polygon $V(q) \cap P_L$. Then they recurse on P_R. They precompute partial visibility polygons of P_L that ignore the obstacles in P_R. Then they create an arrangement on P_R so that all points in one cell combinatorially get the same partial visibility polygon in P_L. At query time, one wants to receive the actual visibility polygon of q and not just a partial one. To obtain this, they compute the visibility cone through the diagonal D (denoted by $V(q, D)$) in the same way as we do. To compute the result at each level of the decomposition, they do point location for q in the arrangement, retrieve the corresponding partial visibility polygon, and intersect it with $V(q, D)$. They store the partial polygons in a persistent red–black tree, making it efficient to compute the visibility polygons for each cell of the arrangement, as well as to compute the intersection with the cone $V(q, D)$.

In our work, we want to return at each level the set $A \cap P_L$ rather than the region $V(q) \cap P_L$. The partial visibility polygons by Aronov et al. [27] can be stored as a sequence of vertices and edges in cyclic order along the boundary of P. This property is what saves one from storing an explicit polygon for each cell in the arrangement, which would significantly increase the space used. We do not have a clear order for arbitrary points in A. Moreover, the cyclic order of the vertices of the visibility polygons in P_L is the same for all points in P_R; but the cyclic order of points in $A \cap P_L$ depends on q.

Due to the these considerations, a naïve adaptation of their approach would be slow. Using the multilevel cutting trees within the general framework of hierarchical decomposition avoids the issue.

7.5.3 Testing for Visibility

Here we discuss the emptiness version of the problem: given a query Q, can any of the objects from A see Q? The solution is the same for all four versions of the problem, whether Q is a point or a line segment and whether A contains points or line segments. We explicitly compute the union of all (weak) visibility polygons of all m objects in A and build a point location or ray shooting data structure on top of it. In all cases, this leads to a $O(m(m + n))$-space data structure that can answer queries in $O(\log(m + n))$-time. We start with some simple observations.

Observation 7.32. *The union of two visibility polygons is either disconnected or simply connected (has no holes).*

Proof. We prove this by contradiction. Let p and q be two objects, and assume that $V(p) \cup V(q)$ has a hole H. Since there is a cycle surrounding H that lies completely inside the union of $V(p) \cup V(q) \subseteq P$ and P is simple, it follows H is contained in P as well. However, since p cannot see points in H, there must be a part of the boundary of the polygon P on the boundary of H. Hence, polygon P has a hole as well. Contradiction. □

Observation 7.33. *A single visibility polygon intersects a polygon edge in a single contiguous line segment.*

This now allows us the prove the following key lemma.

Lemma 7.34. *Let P be a simple polygon with n vertices. Let A be a set of m points or line segments in P. The complexity of the union U of the (weak) visibility polygons of the elements of A inside P is $\Theta(m(m+n))$ in the worst case.*

Proof. A *window* is an edge of a visibility polygon that lies in the interior of P. The union U of the visibility polygons can then have three types of vertices:

1. vertices of P;
2. intersections between two windows; or
3. intersections between an edge of P and a window.

The number of vertices of Type 1 is clearly $O(n)$.

To bound the number of vertices of Type 2, we argue that each pair of viewpoints can only lead to a constant number of vertices, and hence the total number of vertices of Type 2 is $O(m^2)$. We argue as follows. Consider two different objects $p, q \in A$. By Observation 7.32, either the visibility polygons of p and q are disjoint, or their union is simply connected. In the first case, there is no vertex of U that is caused by p and q, so assume the second case. Since the union is simply connected (and the original regions are also simply connected), the regions are *pseudo-ellipses:* their boundaries intersect each other in at most four contiguous curves. These four curves can be degenerate points: if so, they are intersections between windows (one originating from p and one from q) and there are only four of them, p and q contribute a constant number of vertices of Type 2 to U as required. If they are *not* points, then either the shared boundaries are a part of the boundary of P, or p

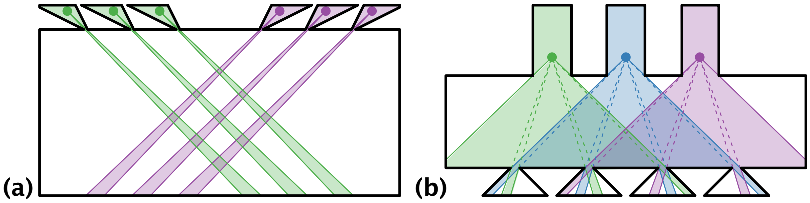


Figure 7.12. Lower bound constructions. **(a)** A construction with a union complexity of $\Omega(m^2)$. **(b)** A construction with a union complexity of $\Omega(mn)$.

and q produce partially coinciding windows; in both cases, they do not contribute any vertices of Type 2 to U.

The number of vertices of Type 3 is at most $O(nm)$ by Observation 7.33.

□

The bound in Lemma 7.34 is tight, as sketched in Figure 7.12: we present two separate constructions with a lower bound of $\Omega(m^2)$ and $\Omega(mn)$. Clearly, the two constructions can happen simultaneously in a single instance, leading to a bound of $\Omega(m^2 + nm)$ as required. Note that this construction is very similar to the one of Bose et al. [34].

Constructing the data structure. To construct U, we first compute all (weak) visibility polygons and merge them using a divide-and-conquer approach. Constructing the visibility polygons takes $O(mn)$ time. Merging two planar subdivisions of sizes $O(m_1(m_1 + n))$ and $O(m_2(m_2 + n))$ into one of size $O(m(m + n))$, where $m = m_1 + m_2$, can be done in $O(m(m+n)\log(m+n)) = O(m^2 \log(m+n) + mn\log(m+n))$ time [76]. The running time thus follows the recurrence $T(m) = 2T(^m/_2) + O(m^2 \log(m+n) + mn\log(m+n))$, which solves to $O(m^2 \log(m+n) + mn\log(m+n)\log m)$.

By storing U in a data structure for point location queries [198], we can directly test if a query point q is visible from any of the objects in A in $O(\log(n+m))$ time. In case the query is a segment pq, we additionally preprocess all components of $P \setminus U$ for ray shooting queries. Note that each such component is indeed a simple polygon (i.e. it is either a hole of U or a region bounded by the outer boundary of U and the polygon boundary), and thus we can answer such queries in logarithmic time using linear space [141]. The total space and preprocessing time for these structures is thus only $O(m(m+n))$. To answer a query, we use the

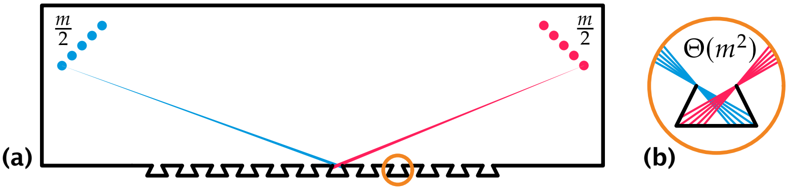


Figure 7.13. The arrangement of m visibility polygons in a simple polygon with n vertices can have complexity $\Omega(m^2 n)$.

point location structure to find the faces containing the two endpoints. If either endpoint lies inside U, we can immediately answer that pq is visible from an object in A. Otherwise, we use the ray shooting data structure associated with the face containing endpoint p, and shoot a ray towards q. If the ray enters U before reaching q, we also find that pq is visible from an an object in A, otherwise (i.e. we reach q before entering U or we hit the boundary of P) pq is not visible.

Theorem 7.35. *Let P be a simple polygon with n vertices and let A be a set of m points or line segments inside P. We can construct a data structure of $O(m(m+n))$ size, in $O(m^2 \log(m+n) + mn \log(m+n) \log m)$ time, that can test if a given query point q or query segment pq is visible from any of the objects in A in $O(\log(m+n))$ time.*

In contrast, the total *arrangement* of all m visibility polygons may have complexity $\Omega(m^2 n)$ (see Figure 7.13). This implies that the same solution for counting or reporting queries would be much slower. Instead, in the remainder of the chapter we explore a different approach.

7.5.4 Subquadratic Counting

Given our data structures, we can generalise the problem: given two sets of points or line segments A and B, each of size m, in a simple polygon P with n vertices, count the number of pairs in $A \times B$ that see each other. Using previous work by Guibas and Hershberger [130] and Eades et al. [99], we can solve this problem by checking the visibility for all pairs. If $n \gg m$, this approach is optimal. In particular, if both sets A and B consist of points, or if one of the sets contains only segments, this yields a solution with the running time $O(n + m^2 \log n)$. However, when $m \gg n$, we want to avoid the m^2 factor. Furthermore, the setting of Section 7.4 is novel, so we consider the full spectrum of trade-offs.

The trick is to use the following well-known technique. Suppose we have a data structure for visibility counting queries with query time $Q(m,n)$ and preprocessing time $P(m,n)$. Pick $k = m^s$ with $0 \leq s \leq 1$. We split the set A into sets $A_1, \ldots, A_k$, with m/k objects each; then we construct a data structure for each set. Finally, with each point in B, we query these k data structures and sum up the counts. It is easy to see that the count is correct; the time that this approach takes is $O\big(k \cdot P(m/k, n) + mk \cdot Q(m/k, n)\big)$. We need to pick s to minimise $O\big(m^s \cdot P(m^{1-s}, n) + m^{1+s} \cdot Q(m^{1-s}, n)\big)$.

Let us show the results for the various settings. Suppose that both sets A and B contain points. First, let us consider the approach of Section 7.2. We have $P(m,n) = O(n + m^{2+\varepsilon} \log n + m \log^2 n)$ and $Q(m,n) = O(\log^2 n + \log n \log m)$. The summands depending only on n come from preprocessing of the polygon that only needs to be done once; so we get

$$\begin{aligned}
&O\big(n + m^s \cdot (m^{(1-s)(2+\varepsilon)} \log n + m^{1-s} \log^2 n) \\
&\qquad\qquad + m^{1+s} \log^2 n + m^{1+s} \log n \log m^{1-s}\big) \\
&= O(n + m^{(1-s)(2+\varepsilon)+s} \log n + m^{1+s} \log^2 n + m^{1+s} \log n \log m)\,.
\end{aligned}$$

Unless $n \gg m$, we pick s such that $(1-s)(2+\varepsilon) + s = 1 + s$; we find $s = (1+\varepsilon)/(2+\varepsilon)$. Therefore, the running time is $O(n + m^{3/2+\varepsilon'} \log n \log nm)$ for this choice of s, where $\varepsilon' > 0$ is an arbitrarily small constant.

Alternatively, we could apply the arrangement-based method of Section 7.2.1. We have $P(m,n) \in O(nm^2 + nm \log n)$ and $Q(m,n) \in O(\log nm)$. Using the formula above, we get

$$\begin{aligned}
&O\big(m^s \cdot (nm^{2-2s} + nm^{1-s} \log n) + m^{1+s} \cdot \log(nm^{1-s})\big) \\
&= O(nm^{2-s} + nm \log n + m^{1+s} \log nm)\,.
\end{aligned}$$

If $m \gg n$, we can pick s to balance the powers of m in the terms; so we set $s = 1/2$ to get

$$O(m^{3/2} \cdot (n + \log n + \log m) + nm \log n) = O(nm^{3/2} + m^{3/2} \log m + nm \log n)\,.$$

If $n \gg m$, it is best to use the pairwise testing approach; however, if $m \gg n$, the arrangement-based approach performs best, and if $m \approx n$, we obtain best results with the decomposition-based approach of Section 7.2.

Now suppose that one of the sets contains points and the other set contains line segments. As it turns out, using the approach of Section 7.3 is always inefficient here; if $m \gg n$, we can use the approach of Section 7.2.1, making sure that we do point queries, and otherwise pairwise testing is fastest.

Finally, suppose both sets consist of line segments. We have $P(m, n) \in O(n^2 \log m + nm^{2+\varepsilon})$ and $Q(m, n) \in O(\log^2 n + \log n \log m)$. We get

$$\begin{aligned} &O\big(m^s \cdot (n^2 \log m^{1-s} + nm^{(1-s)(2+\varepsilon)}) + m^{1+s} \log n \log nm^{1-s}\big) \\ = &O(n^2 m^s \log m + nm^{(1-s)(2+\varepsilon)+s} + m^{1+s} \log^2 n + m^{1+s} \log n \log m)\,. \end{aligned}$$

For $n \gg m$, the time is dominated by $O(n^2 m^s \log m)$, so we pick $s = 0$ and get $O(n^2 \log m + nm^{2+\varepsilon})$ time. For $n \approx m$ or $m \gg n$, we balance the powers by picking $s = {}^{(1+\varepsilon)}\!/\!_{(2+\varepsilon)}$ to get $O(n^2 m^{1/2+\varepsilon'} \log m + nm^{3/2+\varepsilon'} + m^{3/2+\varepsilon'} \log n \log m)$ time for this choice of s, where $\varepsilon' > 0$ is an arbitrarily small constant.

7.5.5 Moving Points

In the context of moving objects, we may interpret a segment as a moving object that traverses the segment from start to end with a constant velocity. This applies both to the objects in a given set A and the query object. More formally, consider objects p, q that have trajectories $p(t) : \mathbb{R} \to \mathbb{R}^2 \cap P$ and $q(t) : \mathbb{R} \to \mathbb{R}^2 \cap P$ inside a polygon P. We say that p and q are *mutually visible* in P if and only if at some time t, the line segment $p(t)q(t)$ is inside the polygon P. In this case, we could be interested in counting how many objects can be seen by the query object at some point during their movement. Note that the settings we discuss in Sections 7.2 and 7.3 lend themselves to this interpretation immediately, since either the query or the objects of A do not move. On the other hand, the setting of a query segment with a set of segments from Section 7.4 does not translate to moving objects.

Eades et al. [99] present a data structure that supports determining whether two query objects see each other at some point in time by preprocessing only the polygon. There is no obvious extension to their data structure that also preprocesses the set of objects. A possible (slow) solution would be to track time as a third dimension and construct the visibility polygon of each point $p \in A$ as it moves. Given a moving object as a query, we would then need to count the visibility polygons (that include a time dimension) that are pierced by the segment. It seems difficult to avoid double counting the points in this scenario; actually solving this problem would be an interesting continuation of the work presented in this paper.

7.5.6 Query Variations

There are many other settings that one could consider as extensions of this work, in addition to the ones we have considered in this paper. For instance, we could consider the reporting version of the problem rather than counting; this works immediately for the point query approaches of Section 7.2, but our use of inclusion–exclusion arguments for segment queries in Sections 7.3 and 7.4 prevents us from easily adapting those to reporting in time proportional to the number of reported segments. Bose et al. [34] showed how to use the arrangement-based approach for reporting, also in the case of querying with line segments. Finally, when considering segments, one can ask many other questions: how much of each segment is seen by a query segment and vice versa, for each segment or in total; these questions and more can also be considered for moving objects as in Section 7.5.5. All of these would be highly exciting directions for future work.

CHAPTER 8

Conclusions

Trajectory analysis is a broad topic, encompassing many useful tools for analysing movement with applications in many areas of science and engineering. In most of those areas, collected data are uncertain, due to the measurement methods or sampling rates. We have investigated the topic of trajectory analysis under uncertainty in this thesis, with the focus on similarity using the Fréchet distance and on further analysis tasks that heavily rely on similarity. Chapter 7, devoted to visibility questions, deviates from this pattern, hinting at the true breadth of the topic. As there has been little research into trajectory analysis under uncertainty in computational geometry, this thesis can be a start of a systematic study, but it has also barely scratched the surface of what is possible. We recall our main results in Section 8.1 and discuss the possible directions for future work in Section 8.2.

8.1 Discussion of Results

First, we studied trajectory similarity under uncertainty using the Fréchet distance, in Chapters 3 and 4. In both two dimensions and one dimension, we provided a host of hardness results and algorithms for the various variants of the (discrete) (weak) Fréchet distance. In particular, we showed NP-hardness for the upper bound discrete and continuous Fréchet distance for indecisive points and intervals in 1D (and higher), and for imprecise points modelled as disks and line segments in 2D (and higher). Following these results, we showed #P-hardness for the problem of finding the expected Fréchet distance under the uniform distribution on the uncertain points. We also showed NP-hardness for the lower bound Fréchet distance in 2D for vertical line segments, but provided polynomial-time algorithms for indecisive points in 2D (and lower) and for intervals in 1D. For the lower bound weak Fréchet distance, we provided an algorithm in 1D, but NP-hardness proofs in 2D and for the discrete weak Fréchet distance. Finally, we gave algorithms for a restricted case for the upper bound Fréchet distance and approximation algorithms for the lower bound Fréchet distance.

We further studied the problem of simplification under uncertainty in Chapter 5. In particular, we considered the variant where the output sequence of (uncertain) points is a subsequence of the input sequence, and for any realisation of the input, the corresponding realisation of the output is a valid simplification. We provided algorithms that solve the problem for the Hausdorff and the Fréchet distance on line segments, disks, and constant-complexity convex polygons.

These first chapters have been centred around the fundamental task of computing trajectory similarity. Considering the importance of similarity, it is worthwhile effort to find efficient algorithms for practical variants of the problem. Unfortunately, many variants turn out to be hard. There are still a few gaps in the various complexity tables, as discussed in Chapters 3 and 4, but perhaps filling them in is less important than obtaining practical approximation algorithms for the hard problems: maximising and especially computing the expected value of the Fréchet distance could both be used in analysis tasks, but now we only have exact solutions for a restricted setting and hardness results for general curves.

Switching to a different view of uncertainty, we discussed map matching in Chapter 6. We showed how to preprocess a realistic map to efficiently answer map-matching queries for an arbitrary trajectory. In particular, we described how to approximate the Fréchet distance to the closest path in the map and how to report a path realising this distance.

Finally, in Chapter 7, we discussed the question of visibility: assuming uncertain measurements, we may sample a collection of line segments connecting two consecutive measurements and ask how many of these are seen by a different subject that is either static or in motion.

The particular problem we studied is that of preprocessing a simple polygon and a set S of points or line segments so that, given a query point or line segment, we can efficiently count the number of objects in S that have an unobstructed line of sight to the query. This problem is likely to have applications outside of uncertainty as well.

The results of the last two chapters demonstrate how questions involving uncertainty can be answered using approaches that, at first sight, have little connection to the uncertainty models. Perhaps such connections can be found for other problems in the future.

8.2 Future Work

Every problem mentioned in the thesis can be studied both in the extremal and the probabilistic model, the latter being more useful but also more complex; in a practical sense, we would greatly benefit from efficient approximations to probabilistic problems.

Beyond the problems adjacent to those in this thesis, there are many further analysis tasks to consider. Some, like clustering, would benefit from more research into similarity measures, but others, like map matching or segmenting trajectories based on periods of (dis)similar behaviour, could be studied separately. For map matching, perhaps there are fewer useful ways to model uncertainty in measurements—as discussed in Chapter 6, implicit modelling can be made explicit with relatively little effort—but for segmentation, it can be incorporated into the algorithms.

In both cases and many more, it makes sense to consider uncertainty between measurements, a subject we barely touched upon in this thesis. Uncertainty models as developed in geographic information science and biology could serve as a base for geometric algorithms on trajectories that incorporate uncertainty. At some level of complexity, this topic may tie into using the context of movement to resolve such uncertainty. However, one can definitely benefit from studying even the simplest models—like the space–time prisms, that merely impose a bound on the speed of the subject—in the context of trajectory analysis and processing tasks such as simplification. A concerted effort in studying the uncertainty between measurements would give us a principled way to interpolate the data between such measurements in algorithms for trajectory analysis.

* * *

We should remember that uncertainty can never be ‘solved’ definitively. It is a fact of life that the data we collect is uncertain and we can only make practically useful statements to a certain degree of accuracy, no matter our techniques. The best we can do is to accept uncertainty, and deal with it openly and honestly, and there is plenty of room to do so—at least when developing algorithms for trajectory analysis.

Bibliography

[1] Amirali Abdullah, Samira Daruki, and Jeff M. Phillips. 'Range Counting Coresets for Uncertain Data'. In: *Proceedings of the 29th Annual Symposium on Computational Geometry (SoCG 2013)*. Ed. by Guilherme D. da Fonseca, Thomas Lewiner, Luis Peñaranda, Timothy M. Chan, Rolf Klein, and Alexander Kröller. New York, NY, USA: Association for Computing Machinery, 2013, pp. 223–232. ISBN: 978-1-4503-2031-3. DOI: `10.1145/2462356.2462388`.

[2] Jonathan S. Abel and James W. Chaffee. 'Existence and Uniqueness of GPS Solutions'. In: *IEEE Transactions on Aerospace and Electronic Systems* 27.6 (1991), pp. 952–956. ISSN: 0018-9251. DOI: `10.1109/7.104271`.

[3] Manuel Abellanas, Ferran Hurtado, Christian Icking, Rolf Klein, Elmar Langetepe, Lihong Ma, Belén Palop, and Vera Sacristán. 'Smallest Color-Spanning Objects'. In: *Algorithms – ESA 2001*. Ed. by Friedhelm Meyer auf der Heide. Lecture Notes in Computer Science 2161. Berlin, Germany: Springer, 2001, pp. 278–289. ISBN: 978-3-540-42493-2. DOI: `10.1007/3-540-44676-1_23`.

[4] Mikkel Abrahamsen, Mark de Berg, Kevin Buchin, Mehran Mehr, and Ali D. Mehrabi. 'Range-Clustering Queries'. In: *Proceedings of the 33rd International Symposium on Computational Geometry (SoCG 2017)*. Ed. by Boris Aronov and Matthew J. Katz. Leibniz International Proceedings in Informatics 77. Wadern, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017, 5. ISBN: 978-3-95977-038-5. DOI: `10.4230/LIPIcs.SoCG.2017.5`.

[5] Peyman Afshani, Pankaj K. Agarwal, Lars Arge, Kasper Green Larsen, and Jeff M. Phillips. '(Approximate) Uncertain Skylines'. In: *Theory of Computing Systems* 52.3 (2013), pp. 342–366. ISSN: 1432-4350. DOI: `10.1007/s00224-012-9382-7`.

[6] Pankaj K. Agarwal, Boris Aronov, Sariel Har-Peled, Jeff M. Phillips, Ke Yi, and Wuzhou Zhang. 'Nearest-Neighbor Searching Under Uncertainty II'. In: *ACM Transactions on Algorithms* 13.1, 3 (2016). ISSN: 1549-6325. DOI: `10.1145/2955098`.

[7] Pankaj K. Agarwal, Rinat Ben Avraham, Haim Kaplan, and Micha Sharir. 'Computing the Discrete Fréchet Distance in Subquadratic Time'. In: *SIAM Journal on Computing* 43.2 (2014), pp. 429–449. ISSN: 0097-5397. DOI: `10.1137/130920526`.

[8] Pankaj K. Agarwal, Siu-Wing Cheng, Yufei Tao, and Ke Yi. 'Indexing Uncertain Data'. In: *Proceedings of the 28th ACM Symposium on Principles of Database Systems (PODS 2009)*. Ed. by Jan Paredaens and Jianwen Su. New York, NY, USA: Association for Computing Machinery, 2009, pp. 137–146. ISBN: 978-1-60558-553-6. DOI: `10.1145/1559795.1559816`.

[9] Pankaj K. Agarwal, Siu-Wing Cheng, and Ke Yi. 'Range Searching on Uncertain Data'. In: *ACM Transactions on Algorithms* 8.4, 43 (2012). ISSN: 1549-6325. DOI: `10.1145/2344422.2344433`.

[10] Pankaj K. Agarwal, Alon Efrat, Swaminathan Sankararaman, and Wuzhou Zhang. 'Nearest-Neighbor Searching Under Uncertainty I'. In: *Discrete & Computational Geometry* 58.3 (2017), pp. 705–745. ISSN: 0179-5376. DOI: `10.1007/s00454-017-9903-x`.

[11] Pankaj K. Agarwal, Sariel Har-Peled, Nabil H. Mustafa, and Yusu Wang. 'Near-Linear Time Approximation Algorithms for Curve Simplification'. In: *Algorithmica* 42.3 (2005), pp. 203–219. ISSN: 0178-4617. DOI: `10.1007/s00453-005-1165-y`.

[12] Pankaj K. Agarwal, Sariel Har-Peled, Subhash Suri, Hakan Yıldız, and Wuzhou Zhang. 'Convex Hulls under Uncertainty'. In: *Algorithmica* 79.2 (2017), pp. 340–367. ISSN: 0178-4617. DOI: `10.1007/s00453-016-0195-y`.

[13] Pankaj K. Agarwal and Marc J. van Kreveld. 'Connected Component and Simple Polygon Intersection Searching'. In: *Algorithmica* 15 (1996), pp. 626–660. ISSN: 0178-4617. DOI: `10.1007/BF01940884`.

[14] Pankaj K. Agarwal, Nirman Kumar, Stavros Sintos, and Subhash Suri. 'Range-Max Queries on Uncertain Data'. In: *Journal of Computer and System Sciences* 94 (2018), pp. 118–134. ISSN: 0022-0000. DOI: `10.1016/j.jcss.2017.09.006`.

[15] Pankaj K. Agarwal and Micha Sharir. 'Applications of a New Space-Partitioning Technique'. In: *Discrete & Computational Geometry* 9 (1993), pp. 11–38. ISSN: 0179-5376. DOI: `10.1007/BF02189304`.

[16] Mahmuda Ahmed and Carola Wenk. 'Constructing Street Networks From GPS Trajectories'. In: *Algorithms – ESA 2012*. Ed. by Leah Epstein and Paolo Ferragina. Lecture Notes in Computer Science 7501. Berlin, Germany: Springer, 2012, pp. 60–71. ISBN: 978-3-642-33089-6. DOI: `10.1007/978-3-642-33090-2_7`.

[17] Hee-Kap Ahn, Sang Won Bae, and Shin-ichi Tanigawa. 'Rectilinear Covering for Imprecise Input Points'. Extended Abstract. In: *Proceedings of the 23rd International Symposium on Algorithms and Computation (ISAAC 2012)*. Ed. by Kun-Mao Chao, Tsan-sheng Hsu, and Der-Tsai Lee. Lecture Notes in Computer Science 7676. Berlin, Germany: Springer, 2012, pp. 309–318. ISBN: 978-3-642-35260-7. DOI: `10.1007/978-3-642-35261-4_34`.

[18] Hee-Kap Ahn, Christian Knauer, Marc Scherfenberg, Lena Schlipf, and Antoine Vigneron. 'Computing the Discrete Fréchet Distance with Imprecise Input'. In: *International Journal of Computational Geometry & Applications* 22.1 (2012), pp. 27–44. ISSN: 0218-1959. DOI: `10.1142/S0218195912600023`.

[19] Mohamed Ali, John Krumm, Travis Rautman, and Ankur Teredesai. 'ACM SIGSPATIAL GIS Cup 2012'. In: *Proceedings of the 20th International Conference on Advances in Geographic Information Systems (SIGSPATIAL 2012)*. Ed. by Isabel Cruz, Craig Knoblock, Peer Kröger, Egemen Tanin, and Peter Widmayer. New York, NY, USA: Association for Computing Machinery, 2012, pp. 597–600. ISBN: 978-1-4503-1691-0. DOI: `10.1145/2424321.2424426`.

[20] Sharareh Alipour, Mohammad Ghodsi, Alireza Zarei, and Maryam Pourreza. 'Visibility Testing and Counting'. In: *Information Processing Letters* 115.9 (2015), pp. 649–654. ISSN: 0020-0190. DOI: `10.1016/j.ipl.2015.03.009`.

[21] Sharareh Alipour and Salman Parsa. 'Hardness of Uncertain Segment Cover, Contiguous SAT and Visibility with Uncertain Obstacles'. In: *Discrete Mathematics, Algorithms and Applications* 15.1, 2250063 (2023). ISSN: 1793-8309. DOI: `10.1142/S179383092250063X`.

[22] Helmut Alt, Alon Efrat, Günter Rote, and Carola Wenk. 'Matching Planar Maps'. In: *Journal of Algorithms* 49.2 (2003), pp. 262–283. ISSN: 0196-6774. DOI: `10.1016/S0196-6774(03)00085-3`.

[23] Helmut Alt and Michael Godau. 'Computing the Fréchet Distance between Two Polygonal Curves'. In: *International Journal of Computational Geometry & Applications* 5.1–2 (1995), pp. 75–91. ISSN: 0218-1959. DOI: `10.1142/S0218195995000064`.

[24] Hugo Alves Akitaya, Maike Buchin, Leonie Ryvkin, and Jérôme Urhausen. 'The k-Fréchet Distance: How to Walk Your Dog While Teleporting'. In: *Proceedings of the 30th International Symposium on Algorithms and Computation (ISAAC 2019)*. Ed. by Pinyan Lu and Guochuan Zhang. Leibniz International Proceedings in Informatics 149. Wadern, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019, 50. ISBN: 978-3-95977-130-6. DOI: `10.4230/LIPIcs.ISAAC.2019.50`.

[25] Esther M. Arkin, Aritra Banik, Paz Carmi, Gui Citovsky, Matthew J. Katz, Joseph S. B. Mitchell, and Marina Simakov. 'Selecting and Covering Colored Points'. In: *Discrete Applied Mathematics* 250 (2018), pp. 75–86. ISSN: 0166-218X. DOI: `10.1016/j.dam.2018.05.011`.

[26] Boris Aronov, Kevin Buchin, Maike Buchin, Bart Jansen, Tom de Jong, Marc J. van Kreveld, Maarten Löffler, Jun Luo, Rodrigo I. Silveira, and Bettina Speckmann. 'Connect the Dot: Computing Feed-Links for Network Extension'. In: *Journal of Spatial Information Science* 3 (2011), pp. 3–31. ISSN: 1948-660X. DOI: `10.5311/JOSIS.2011.3.47`.

[27] Boris Aronov, Leonidas J. Guibas, Marek Teichmann, and Li Zhang. 'Visibility Queries and Maintenance in Simple Polygons'. In: *Discrete & Computational Geometry* 27 (2002), pp. 461–483. ISSN: 0179-5376. DOI: `10.1007/s00454-001-0089-9`.

[28] Erik Axell, Peter Johansson, Mikael Alexandersson, and Jouni Rantakokko. *Estimation of the Position Error in GPS Receivers*. Tech. rep. FOI-R–3840–SE. Totalförsvarets Forskningsinstitut, 2014. URL: `https://www.foi.se/rest-api/report/FOI-R--3840--SE` (visited on 24/06/2023).

[29] Gill Barequet, Danny Z. Chen, Ovidiu Daescu, Michael T. Goodrich, and Jack S. Snoeyink. 'Efficiently Approximating Polygonal Paths in Three and Higher Dimensions'. In: *Algorithmica* 33.2 (2002), pp. 150–167. ISSN: 0178-4617. DOI: `10.1007/s00453-001-0096-5`.

[30] Boaz Ben-Moshe, Olaf Hall-Holt, Matthew J. Katz, and Joseph S. B. Mitchell. 'Computing the Visibility Graph of Points within a Polygon'. In: *Proceedings of the 20th Annual Symposium on Computational Geometry (SoCG 2004)*. Ed. by Jack S. Snoeyink and Jean-Daniel Boissonnat. New York, NY, USA: Association for Computing Machinery, 2004, pp. 27–35. ISBN: 978-1-58113-885-6. DOI: `10.1145/997817.997825`.

[31] Mark de Berg, Elena Mumford, and Marcel Roeloffzen. *Finding Structures on Imprecise Points*. Presented at EuroCG 2010, Dortmund, Germany. 2010. URL: `https://eurocg.org/2010/proceedings.pdf` (visited on 15/06/2023).

[32] Donald J. Berndt and James Clifford. 'Using Dynamic Time Warping to Find Patterns in Time Series'. In: *AAAI-94 Workshop on Knowledge Discovery in Databases (KDD-94)*. Ed. by Usama M. Fayyad and Ramasamy Uthurusamy. Menlo Park, CA, USA: AAAI Press, 1994, pp. 359–370. URL: `https://ww.aaai.org/Papers/Workshops/1994/WS-94-03/WS94-03-031.pdf` (visited on 11/05/2023).

[33] Dimitris Bertsimas and Louis H. Howell. 'Further Results on the Probabilistic Traveling Salesman Problem'. In: *European Journal of Operational Research* 65.1 (1993), pp. 68–95. ISSN: 0377-2217. DOI: `10.1016/0377-2217(93)90145-D`.

[34] Prosenjit Bose, Anna Lubiw, and James Ian Munro. 'Efficient Visibility Queries in Simple Polygons'. In: *Computational Geometry: Theory & Applications* 23.3 (2002), pp. 313–335. ISSN: 0925-7721. DOI: `10.1016/S0925-7721(01)00070-0`.

[35] Sotiris Brakatsoulas, Dieter Pfoser, Randall Salas, and Carola Wenk. 'On Map-Matching Vehicle Tracking Data'. In: *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB 2005)*. Ed. by Kjell Bratbergsengen. Los Angeles, CA, USA: VLDB Endowment, 2005, pp. 853–864. ISBN: 978-1-59593-154-2. URL: `https://dl.acm.org/doi/10.5555/1083592.1083691` (visited on 15/06/2023).

[36] Milutin Brankovic, Kevin Buchin, Koen Klaren, André Nusser, Aleksandr Popov, and Sampson Wong. '(k, ℓ)-Medians Clustering of Trajectories Using Continuous Dynamic Time Warping'. In: *Proceedings of the 28th International Conference on Advances in Geographic Information Systems (SIGSPATIAL 2020)*. Ed. by Chang-Tien Lu, Fusheng Wang, Goce Trajcevski, Yan Huang, Shawn Newsam, and Li Xiong. New York, NY, USA: Association for Computing Machinery, 2020, pp. 99–110. ISBN: 978-1-4503-8019-5. DOI: `10.1145/3397536.3422245`.

[37] Karl Bringmann and Bhaskar Ray Chaudhury. 'Polyline Simplification Has Cubic Complexity'. In: *Proceedings of the 35th International Symposium on Computational Geometry (SoCG 2019)*. Ed. by Gill Barequet and Yusu Wang. Leibniz International Proceedings in Informatics 129. Wadern, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019, 18. ISBN: 978-3-95977-104-7. DOI: `10.4230/LIPIcs.SoCG.2019.18`.

[38] Karl Bringmann, Marvin Künnemann, and André Nusser. 'Fréchet Distance under Translation: Conditional Hardness and an Algorithm via Offline Dynamic Grid Reachability'. In: *Proceedings of the 30th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA 2019)*. Ed. by Timothy M. Chan. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2019, pp. 2902–2921. ISBN: 978-1-61197-548-2. DOI: `10.1137/1.9781611975482.180`.

[39] Karl Bringmann and Wolfgang Mulzer. 'Approximability of the Discrete Fréchet Distance'. In: *Journal of Computational Geometry* 7.2 (2016), pp. 46–76. ISSN: 1920-180X. DOI: `10.20382/jocg.v7i2a4`.

[40] Kevin Buchin, Maike Buchin, David Duran, Brittany Terese Fasy, Roel Jacobs, Vera Sacristán, Rodrigo I. Silveira, Frank Staals, and Carola Wenk. 'Clustering Trajectories for Map Construction'. In: *Proceedings of the 25th International Conference on Advances in Geographic Information Systems (SIGSPATIAL 2017)*. Ed. by Erik Hoel, Shawn Newsam, Siva Ravada, Roberto Tamassia, and Goce Trajcevski. New York, NY, USA: Association for Computing Machinery, 2017, 14. ISBN: 978-1-4503-5490-5. DOI: `10.1145/3139958.3139964`.

[41] Kevin Buchin, Maike Buchin, and Joachim Gudmundsson. 'Constrained Free Space Diagrams: A Tool for Trajectory Analysis'. In: *International Journal of Geographical Information Science* 24.7 (2010), pp. 1101–1125. ISSN: 1365-8816. DOI: `10.1080/13658810903569598`.

[42] Kevin Buchin, Maike Buchin, Joachim Gudmundsson, Aleksandr Popov, and Sampson Wong. *Map-Matching Queries under Fréchet Distance on Low-Density Spanners*. Presented at EuroCG 2023, Barcelona, Spain. 2023. URL: `https://dccg.upc.edu/eurocg23/wp-content/uploads/2023/05/Booklet_EuroCG2023.pdf` (visited on 01/07/2023).

[43] Kevin Buchin, Maike Buchin, Christian Knauer, Günter Rote, and Carola Wenk. *How Difficult Is It to Walk the Dog?* Presented at EuroCG 2007, Graz, Austria. 2007. URL: `https://page.mi.fu-berlin.de/rote/Papers/pdf/How+difficult+is+it+to+walk+the+dog.pdf` (visited on 15/06/2023).

[44] Kevin Buchin, Maike Buchin, Marc J. van Kreveld, Bettina Speckmann, and Frank Staals. 'Trajectory Grouping Structure'. In: *Journal of Computational Geometry* 6.1 (2015), pp. 75–98. ISSN: 1920-180X. DOI: `10.20382/jocg.v6i1a3`.

[45] Kevin Buchin, Maike Buchin, Wouter Meulemans, and Wolfgang Mulzer. 'Four Soviets Walk the Dog: Improved Bounds for Computing the Fréchet Distance'. In: *Discrete & Computational Geometry* 58.1 (2017), pp. 180–216. ISSN: 0179-5376. DOI: `10.1007/s00454-017-9878-7`.

[46] Kevin Buchin, Bram Custers, Ivor van der Hoog, Maarten Löffler, Aleksandr Popov, Marcel Roeloffzen, and Frank Staals. 'Segment Visibility Counting Queries in Polygons'. In: *Proceedings of the 33rd International Symposium on Algorithms and Computation (ISAAC 2022)*. Ed. by Sang Won Bae and Heejin Park. Leibniz International Proceedings in Informatics 248. Wadern, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 58. ISBN: 978-3-95977-258-7. DOI: `10.4230/LIPIcs.ISAAC.2022.58`.

[47] Kevin Buchin, Anne Driemel, Joachim Gudmundsson, Michael Horton, Irina Kostitsyna, Maarten Löffler, and Martijn Struijs. 'Approximating (k, ℓ)-Center Clustering for Curves'. In: *Proceedings of the 30th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA 2019)*. Ed. by Timothy M. Chan. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2019, pp. 2922–2938. ISBN: 978-1-61197-548-2. DOI: `10.1137/1.9781611975482.181`.

[48] Kevin Buchin, Anne Driemel, Natasja van de L'Isle, and André Nusser. 'klcluster: Center-Based Clustering of Trajectories'. In: *Proceedings of the 27th International Conference on Advances in Geographic Information Systems (SIGSPATIAL 2019)*. Ed. by Farnoush Banaei-Kashani, Goce Trajcevski, Ralf Hartmut Güting, Lars Kulik, and Shawn Newsam. New York, NY, USA: Association for Computing Machinery, 2019, pp. 496–499. ISBN: 978-1-4503-6909-1. DOI: `10.1145/3347146.3359111`.

[49] Kevin Buchin, Anne Driemel, and Martijn Struijs. 'On the Hardness of Computing an Average Curve'. In: *Proceedings of the 17th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2020)*. Ed. by Susanne Albers. Leibniz International Proceedings in Informatics 162. Wadern, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020, 19. ISBN: 978-3-95977-150-4. DOI: `10.4230/LIPIcs.SWAT.2020.19`.

[50] Kevin Buchin, Chenglin Fan, Maarten Löffler, Aleksandr Popov, Benjamin Raichel, and Marcel Roeloffzen. 'Fréchet Distance for Uncertain Curves'. In: *Proceedings of the 47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*. Ed. by Artur Czumaj, Anuj Dawar, and Emanuela Merelli. Leibniz International Proceedings in Informatics 168. Wadern, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020, 20. ISBN: 978-3-95977-138-2. DOI: `10.4230/LIPIcs.ICALP.2020.20`.

[51] Kevin Buchin, Chenglin Fan, Maarten Löffler, Aleksandr Popov, Benjamin Raichel, and Marcel Roeloffzen. 'Fréchet Distance for Uncertain Curves'. In: *ACM Transactions on Algorithms* 19.3, 29 (2023). ISSN: 1549-6325. DOI: `10.1145/3597640`.

[52] Kevin Buchin, Joachim Gudmundsson, Antonia Kalb, Aleksandr Popov, Carolin Rehs, André van Renssen, and Sampson Wong. 'Oriented Spanners'. In: *Proceedings of the 31st Annual European Symposium on Algorithms (ESA 2023)*. Ed. by Inge Li Gørtz, Martin Farach-Colton, Simon J. Puglisi, and Grzegorz Herman. Leibniz International Proceedings in Informatics 274. Wadern, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, 26. ISBN: 978-3-95977-295-2. DOI: `10.4230/LIPIcs.ESA.2023.26`.

[53] Kevin Buchin, Maximilian Konzack, and Wim Reddingius. 'Progressive Simplification of Polygonal Curves'. In: *Computational Geometry: Theory & Applications* 88, 101620 (2020). ISSN: 0925-7721. DOI: `10.1016/j.comgeo.2020.101620`.

[54] Kevin Buchin, Irina Kostitsyna, Maarten Löffler, and Rodrigo I. Silveira. 'Region-Based Approximation of Probability Distributions: For Visibility between Imprecise Points among Obstacles'. In: *Algorithmica* 81.7 (2019), pp. 2682–2715. ISSN: 0178-4617. DOI: `10.1007/s00453-019-00551-2`.

[55] Kevin Buchin, Maarten Löffler, Pat Morin, and Wolfgang Mulzer. 'Preprocessing Imprecise Points for Delaunay Triangulation: Simplified and Extended'. In: *Algorithmica* 61.3 (2011), pp. 674–693. ISSN: 0178-4617. DOI: `10.1007/s00453-010-9430-0`.

[56] Kevin Buchin, Maarten Löffler, Tim Ophelders, Aleksandr Popov, Jérôme Urhausen, and Kevin Verbeek. 'Computing the Fréchet Distance between Uncertain Curves in One Dimension'. In: *Proceedings of the 17th International Symposium on Algorithms and Data Structures (WADS 2021)*. Ed. by Anna Lubiw, Mohammad Salavatipour, and Meng He. Lecture Notes in Computer Science 12808. Berlin, Germany: Springer, 2021, pp. 243–257. ISBN: 978-3-030-83507-1. DOI: `10.1007/978-3-030-83508-8_18`.

[57] Kevin Buchin, Maarten Löffler, Tim Ophelders, Aleksandr Popov, Jérôme Urhausen, and Kevin Verbeek. 'Computing the Fréchet Distance between Uncertain Curves in One Dimension'. In: *Computational Geometry: Theory & Applications* 109, 101923 (2023). ISSN: 0925-7721. DOI: `10.1016/j.comgeo.2022.101923`.

[58] Kevin Buchin, Maarten Löffler, Aleksandr Popov, and Marcel Roeloffzen. 'Uncertain Curve Simplification'. In: *Proceedings of the 46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021)*. Ed. by Filippo Bonchi and Simon J. Puglisi. Leibniz International Proceedings in Informatics 202. Wadern, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, 26. ISBN: 978-3-95977-201-3. DOI: `10.4230/LIPIcs.MFCS.2021.26`.

[59] Kevin Buchin, André Nusser, and Sampson Wong. 'Computing Continuous Dynamic Time Warping of Time Series in Polynomial Time'. In: *Proceedings of the 38th International Symposium on Computational Geometry (SoCG 2022)*. Ed. by Xavier Goaoc and Michael Kerber. Leibniz International Proceedings in Informatics 224. Wadern, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 22. ISBN: 978-3-95977-227-3. DOI: `10.4230/LIPIcs.SoCG.2022.22`.

[60] Kevin Buchin, Tim Ophelders, and Bettina Speckmann. 'SETH Says: Weak Fréchet Distance Is Faster, but Only If It Is Continuous and in One Dimension'. In: *Proceedings of the 30th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA 2019)*. Ed. by Timothy M. Chan. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2019, pp. 2887–2901. ISBN: 978-1-61197-548-2. DOI: `10.1137/1.9781611975482.179`.

[61] Kevin Buchin, Jeff M. Phillips, and Pingfan Tang. 'Approximating the Distribution of the Median and Other Robust Estimators on Uncertain Data'. In: *Proceedings of the 34th International Symposium on Computational Geometry (SoCG 2018)*. Ed. by Bettina Speckmann and Csaba D. Tóth. Leibniz International Proceedings in Informatics 99. Wadern, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018, 16. ISBN: 978-3-95977-066-8. DOI: `10.4230/LIPIcs.SoCG.2018.16`.

[62] Kevin Buchin, Stef Sijben, T. Jean Marie Arseneau, and Erik P. Willems. 'Detecting Movement Patterns Using Brownian Bridges'. In: *Proceedings of the 20th International Conference on Advances in Geographic Information Systems (SIGSPATIAL 2012)*. Ed. by Isabel Cruz, Craig Knoblock, Peer Kröger, Egemen Tanin, and Peter Widmayer. New York, NY, USA: Association for Computing Machinery, 2012, pp. 119–128. ISBN: 978-1-4503-1691-0. DOI: `10.1145/2424321.2424338`.

[63] Maike Buchin, Anne Driemel, and Bettina Speckmann. 'Computing the Fréchet Distance with Shortcuts is NP-Hard'. In: *Proceedings of the 30th Annual Symposium on Computational Geometry (SoCG 2014)*. Ed. by Siu-Wing Cheng and Olivier Devillers. New York, NY, USA: Association for Computing Machinery, 2014, pp. 367–376. ISBN: 978-1-4503-2594-3. DOI: `10.1145/2582112.2582144`.

[64] Maike Buchin and Ross S. Purves. 'Computing Similarity of Coarse and Irregular Trajectories Using Space-Time Prisms'. In: *Proceedings of the 21st International Conference on Advances in Geographic Information Systems (SIGSPATIAL 2013)*. Ed. by Craig Knoblock, Peer Kröger, John Krumm, Markus Schneider, and Peter Widmayer. New York, NY, USA: Association for Computing Machinery, 2013, pp. 456–459. ISBN: 978-1-4503-2521-9. DOI: `10.1145/2525314.2525459`.

[65] Maike Buchin and Stef Sijben. *Discrete Fréchet Distance for Uncertain Points*. Presented at EuroCG 2016, Lugano, Switzerland. 2016. URL: `http://www.eurocg2016.usi.ch/sites/default/files/paper_72.pdf` (visited on 15/06/2023).

[66] Daniel Busto, William S. Evans, and David G. Kirkpatrick. 'Minimizing Interference Potential among Moving Entities'. In: *Proceedings of the 30th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA 2019)*. Ed. by Timothy M. Chan. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2019, pp. 2400–2418. ISBN: 978-1-61197-548-2. DOI: `10.1137/1.9781611975482.147`.

[67] Mojtaba Nouri Bygi, Shervin Daneshpajouh, Sharareh Alipour, and Mohammad Ghodsi. 'Weak Visibility Counting in Simple Polygons'. In: *Journal of Computational and Applied Mathematics* 288 (2015), pp. 215–222. ISSN: 0377-0427. DOI: `10.1016/j.cam.2015.04.018`.

[68] Leizhen Cai and Mark Keil. 'Computing Visibility Information in an Inaccurate Simple Polygon'. In: *International Journal of Computational Geometry & Applications* 7 (1997), pp. 515–538. ISSN: 0218-1959. DOI: `10.1142/S0218195997000326`.

[69] Erin Chambers, Brittany Terese Fasy, Yusu Wang, and Carola Wenk. 'Map-Matching Using Shortest Paths'. In: *ACM Transactions on Spatial Algorithms and Systems* 6.1, 6 (2020). ISSN: 2374-0353. DOI: `10.1145/3368617`.

[70] W. S. Chan and Francis Y. L. Chin. 'Approximation of Polygonal Curves with Minimum Number of Line Segments or Minimum Error'. In: *International Journal of Computational Geometry & Applications* 6.1 (1996), pp. 59–77. ISSN: 0218-1959. DOI: `10.1142/S0218195996000058`.

[71] Pingfu Chao, Yehong Xu, Wen Hua, and Xiaofang Zhou. 'A Survey on Map-Matching Algorithms'. In: *Databases Theory and Applications: Proceedings of the 31st Australasian Database Conference (ADC 2020)*. Ed. by Renata Borovica-Gajic, Jianzhong Qi, and Weiqing Wang. Lecture Notes in Computer Science 12008. Berlin, Germany: Springer, 2020, pp. 121–133. ISBN: 978-3-030-39468-4. DOI: `10.1007/978-3-030-39469-1_10`.

[72] Frederic Chazal, Daniel Chen, Leonidas J. Guibas, Xiaoye Jiang, and Christian Sommer. 'Data-Driven Trajectory Smoothing'. In: *Proceedings of the 19th International Conference on Advances in Geographic Information Systems (SIGSPATIAL 2011)*. Ed. by Isabel F. Cruz, Divyakant Agrawal, Christian S. Jensen, Eyal Ofek, and Egemen Tanin. New York, NY, USA: Association for Computing Machinery, 2011, pp. 251–260. ISBN: 978-1-4503-1031-4. DOI: `10.1145/2093973.2094007`.

[73] Bernard Chazelle. 'A Theorem on Polygon Cutting with Applications'. In: *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 1982)*. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers, 1982, pp. 339–349. DOI: `10.1109/SFCS.1982.58`.

[74] Bernard Chazelle. 'Cutting Hyperplanes for Divide-and-Conquer'. In: *Discrete & Computational Geometry* 9 (1993), pp. 145–158. ISSN: 0179-5376. DOI: `10.1007/BF02189314`.

[75] Bernard Chazelle. 'Triangulating a Simple Polygon in Linear Time'. In: *Discrete & Computational Geometry* 6 (1991), pp. 485–524. ISSN: 0179-5376. DOI: `10.1007/BF02574703`.

[76] Bernard Chazelle and Herbert Edelsbrunner. 'An Optimal Algorithm for Intersecting Line Segments in the Plane'. In: *Journal of the ACM* 39.1 (1992), pp. 1–54. ISSN: 0004-5411. DOI: `10.1145/147508.147511`.

[77] Bernard Chazelle and Leonidas J. Guibas. 'Visibility and Intersection Problems in Plane Geometry'. In: *Discrete & Computational Geometry* 4 (1989), pp. 551–581. ISSN: 0179-5376. DOI: `10.1007/BF02187747`.

[78] Bernard Chazelle, Micha Sharir, and Emo Welzl. 'Quasi-Optimal Upper Bounds for Simplex Range Searching and New Zone Theorems'. In: *Algorithmica* 8 (1992), pp. 407–429. ISSN: 0178-4617. DOI: `10.1007/BF01758854`.

[79] Daniel Chen, Anne Driemel, Leonidas J. Guibas, Andy Nguyen, and Carola Wenk. 'Approximate Map Matching with Respect to the Fréchet Distance'. In: *Proceedings of the 2011 Workshop on Algorithm Engineering and Experiments (ALENEX 2011)*. Ed. by Matthias Müller-Hannemann and Renato Werneck. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2011, pp. 75–83. ISBN: 978-1-61197-291-7. DOI: `10.1137/1.9781611972917.8`.

[80] Daniel Chen, Christian Sommer, and Daniel Wolleb. 'Fast Map Matching with Vertex-Monotone Fréchet Distance'. In: *Proceedings of the 21st Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2021)*. Ed. by Matthias Müller-Hannemann and Federico Perea. Open Access Series in Informatics 96. Wadern, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, 10. ISBN: 978-3-95977-213-6. DOI: `10.4230/OASIcs.ATMOS.2021.10`.

[81] Danny Ziyi Chen and Haitao Wang. 'Weak Visibility Queries of Line Segments in Simple Polygons'. In: *Computational Geometry: Theory & Applications* 48.6 (2015), pp. 443–452. ISSN: 0925-7721. DOI: `10.1016/j.comgeo.2015.02.001`.

[82] Lei Chen, M. Tamer Özsu, and Vincent Oria. 'Robust and Fast Similarity Search for Moving Object Trajectories'. In: *Proceedings of the 2005 ACM International Conference on Management of Data (SIGMOD 2005)*. Ed. by Fatma Özcan. New York, NY, USA: Association for Computing Machinery, 2005, pp. 491–502. ISBN: 978-1-59593-060-6. DOI: `10.1145/1066157.1066213`.

[83] Yongwan Chun, Mei-Po Kwan, and Daniel A. Griffith, eds. 33.6 (2019): *Uncertainty and Context in GIScience and Geography*. ISSN: 1365-8816. URL: `https://www.tandfonline.com/toc/tgis20/33/6` (visited on 10/07/2023).

[84] Gui Citovsky, Tyler Mayer, and Joseph S. B. Mitchell. 'TSP with Locational Uncertainty: The Adversarial Model'. In: *Proceedings of the 33rd International Symposium on Computational Geometry (SoCG 2017)*. Ed. by Boris Aronov and Matthew J. Katz. Leibniz International Proceedings in Informatics 77. Wadern, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017, 32. ISBN: 978-3-95977-038-5. DOI: `10.4230/LIPIcs.SoCG.2017.32`.

[85] Kenneth L. Clarkson. 'New Applications of Random Sampling in Computational Geometry'. In: *Discrete & Computational Geometry* 2 (1987), pp. 195–222. ISSN: 0179-5376. DOI: `10.1007/BF02187879`.

[86] Edward A. Codling, Michael J. Plank, and Simon Benhamou. 'Random Walk Models in Biology'. In: *Journal of the Royal Society Interface* 5.25 (2008), pp. 813–834. ISSN: 1742-5689. DOI: `10.1098/rsif.2008.0014`.

[87] Jacobus Conradi and Anne Driemel. 'On Computing the k-Shortcut Fréchet Distance'. In: *Proceedings of the 49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*. Ed. by Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff. Leibniz International Proceedings in Informatics 229. Wadern, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 46. ISBN: 978-3-95977-235-8. DOI: `10.4230/LIPIcs.ICALP.2022.46`.

[88] Bram Custers, Wouter Meulemans, Marcel Roeloffzen, Bettina Speckmann, and Kevin Verbeek. 'Physically Consistent Map Matching'. In: *Proceedings of the 30th International Conference on Advances in Geographic Information Systems (SIGSPATIAL 2022)*. Ed. by Matthias Renz, Mohamed Sarwat, Mario A. Nascimento, Shashi Shekhar, and Xing Xie. New York, NY, USA: Association for Computing Machinery, 2022, 56. ISBN: 978-1-4503-9529-8. DOI: `10.1145/3557915.3560991`.

[89] Bram Custers, Wouter Meulemans, Bettina Speckmann, and Kevin Verbeek. 'Route Reconstruction from Traffic Flow via Representative Trajectories'. In: *Proceedings of the 29th International Conference on Advances in Geographic Information Systems (SIGSPATIAL 2021)*. Ed. by Xiaofeng Meng, Fusheng Wang, Chang-Tien Lu, Yan Huang, Shashi Shekhar, and Xing Xie. New York, NY, USA: Association for Computing Machinery, 2021, pp. 41–52. ISBN: 978-1-4503-8664-7. DOI: `10.1145/3474717.3483650`.

[90] Sandip Das, Partha P. Goswami, and Subhas C. Nandy. 'Smallest Color-Spanning Object Revisited'. In: *International Journal of Computational Geometry & Applications* 19.5 (2009), pp. 457–478. ISSN: 0218-1959. DOI: `10.1142/S0218195909003076`.

[91] Urška Demšar, Kevin Buchin, Francesca Cagnacci, Kamran Safi, Bettina Speckmann, Nico van de Weghe, Daniel Weiskopf, and Robert Weibel. 'Analysis and Visualisation of Movement: An Interdisciplinary Review'. In: *Movement Ecology* 3, 5 (2015). ISSN: 2051-3933. DOI: `10.1186/s40462-015-0032-y`.

[92] Olivier Devillers. 'Delaunay Triangulation of Imprecise Points: Preprocess and Actually Get a Fast Query Time'. In: *Journal of Computational Geometry* 2.1 (2011), pp. 30–45. ISSN: 1920-180X. DOI: `10.20382/v2i1a3`.

[93] Thomas Devogele, Laurent Etienne, Maxence Esnault, and Florian Lardy. 'Optimized Discrete Fréchet Distance Between Trajectories'. In: *Proceedings of the 6th ACM SIGSPATIAL Workshop on Analytics for Big Geospatial Data (BigSpatial 2017)*. Ed. by Varun Chandola and Ranga Raju Vatsavai. New York, NY, USA: Association for Computing Machinery, 2017, pp. 11–19. ISBN: 978-1-4503-5494-3. DOI: `10.1145/3150919.3150924`.

[94] David H. Douglas and Thomas K. Peucker. 'Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or Its Caricature'. In: *Cartographica: The International Journal for Geographic Information and Geovisualization* 10.2 (1973), pp. 112–122. ISSN: 0317-7173. DOI: `10.3138/FM57-6770-U75U-7727`.

[95] Anne Driemel and Sariel Har-Peled. 'Jaywalking Your Dog: Computing the Fréchet Distance with Shortcuts'. In: *SIAM Journal on Computing* 42.5 (2018), pp. 1830–1866. ISSN: 0097-5397. DOI: `10.1137/120865112`.

[96] Anne Driemel, Sariel Har-Peled, and Carola Wenk. 'Approximating the Fréchet Distance for Realistic Curves in Near Linear Time'. In: *Discrete & Computational Geometry* 48.1 (2012), pp. 94–127. ISSN: 0179-5376. DOI: `10.1007/s00454-012-9402-z`.

[97] Anne Driemel, Herman Haverkort, Maarten Löffler, and Rodrigo I. Silveira. 'Flow Computations on Imprecise Terrains'. In: *Journal of Computational Geometry* 4.1 (2013), pp. 38–78. ISSN: 1920-180X. DOI: `10.20382/jocg.v4i1a3`.

[98] Anne Driemel, Amer Krivošija, and Christian Sohler. 'Clustering Time Series under the Fréchet Distance'. In: *Proceedings of the 27th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA 2016)*. Ed. by Robert Krauthgamer. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2016, pp. 766–785. ISBN: 978-1-61197-433-1. DOI: `10.1137/1.9781611974331.ch55`.

[99] Patrick Eades, Ivor van der Hoog, Maarten Löffler, and Frank Staals. 'Trajectory Visibility'. In: *Proceedings of the 17th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2020)*. Ed. by Susanne Albers. Leibniz International Proceedings in Informatics 162. Wadern, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020, 23. ISBN: 978-3-95977-150-4. DOI: `10.4230/LIPIcs.SWAT.2020.23`.

[100] Thomas Eiter and Heikki Olavi Mannila. *Computing Discrete Fréchet Distance*. Tech. rep. CD-TR 94/64. Technische Universität Wien, 1994. URL: `http://www.kr.tuwien.ac.at/staff/eiter/et-archive/cdtr9464.pdf` (visited on 12/06/2023).

[101] Hossam El Gindy and David Avis. 'A Linear Algorithm for Computing the Visibility Polygon from a Point'. In: *Journal of Algorithms* 2.2 (1981), pp. 186–197. ISSN: 0196-6774. DOI: `10.1016/0196-6774(81)90019-5`.

[102] Hossam El Gindy and David Avis. 'A Linear Algorithm for Computing the Visibility Polygon from a Point'. In: *Journal of Algorithms* 2.2 (1981), pp. 186–197. ISSN: 0196-6774. DOI: `10.1016/0196-6774(81)90019-5`.

[103] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 'A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise'. In: *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD 1996)*. Ed. by Usama Fayyad, Evangelos Simoudis, and Jiawei Han. Menlo Park, CA, USA: AAAI Press, 1996, pp. 226–231. ISBN: 978-1-57735-004-0. URL: https://ww.aaai.org/Papers/KDD/1996/KDD96-037.pdf (visited on 15/06/2023).

[104] William S. Evans, Ivor van der Hoog, David G. Kirkpatrick, and Maarten Löffler. *Towards the Minimization of Global Measures of Congestion Potential for Moving Points*. Presented at EuroCG 2022, Perugia, Italy. 2022. URL: https://eurocg2022.unipg.it/booklet/EuroCG2022-Booklet.pdf (visited on 18/06/2023).

[105] William S. Evans, David Kirkpatrick, Maarten Löffler, and Frank Staals. 'Competitive Query Strategies for Minimising the Ply of the Potential Locations of Moving Points'. In: *Proceedings of the 29th Annual Symposium on Computational Geometry (SoCG 2013)*. Ed. by Guilherme D. da Fonseca, Thomas Lewiner, Luis Peñaranda, Timothy M. Chan, Rolf Klein, and Alexander Kröller. New York, NY, USA: Association for Computing Machinery, 2013, pp. 155–164. ISBN: 978-1-4503-2031-3. DOI: 10.1145/2462356.2462395.

[106] William S. Evans, David G. Kirkpatrick, Maarten Löffler, and Frank Staals. 'Minimizing Co-location Potential of Moving Entities'. In: *SIAM Journal on Computing* 45.5 (2016), pp. 1870–1893. ISSN: 0097-5397. DOI: 10.1137/15M1031217.

[107] Chenglin Fan, Jun Luo, and Binhai Zhu. 'Tight Approximation Bounds for Connectivity with a Color-Spanning Set'. In: *Proceedings of the 24th International Symposium on Algorithms and Computation (ISAAC 2013)*. Ed. by Leizhen Cai, Siu-Wing Cheng, and Tak-Wah Lam. Lecture Notes in Computer Science 8283. Berlin, Germany: Springer, 2013, pp. 590–600. ISBN: 978-3-642-45029-7. DOI: 10.1007/978-3-642-45030-3_55.

[108] Chenglin Fan and Benjamin Raichel. 'Computing the Fréchet Gap Distance'. In: *Discrete & Computational Geometry* 65 (2020), pp. 1244–1274. ISSN: 0179-5376. DOI: 10.1007/s00454-020-00224-w.

[109] Chenglin Fan and Binhai Zhu. *Complexity and Algorithms for the Discrete Fréchet Distance Upper Bound with Imprecise Input*. 2018. arXiv: 1509.02576v2 [cs.CG].

[110] Omrit Filtser and Matthew J. Katz. 'Algorithms for the Discrete Fréchet Distance Under Translation'. In: *Proceedings of the 16th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2018)*. Ed. by David Eppstein. Leibniz International Proceedings in Informatics 101. Wadern, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018, 20. ISBN: 978-3-95977-068-2. DOI: 10.4230/LIPIcs.SWAT.2018.20.

[111] Omrit Filtser and Matthew J. Katz. *The Discrete Fréchet Gap*. 2015. arXiv: 1506.04861v1 [cs.CG].

[112] Martin Fink, John E. Hershberger, Nirman Kumar, and Subhash Suri. 'Hyperplane Separability and Convexity of Probabilistic Point Sets'. In: *Journal of Computational Geometry* 8.2 (2017), pp. 32–57. ISSN: 1920-180X. DOI: 10.20382/jocg.v8i2a3.

[113] Marco Fiore, Panagiota Katsikouli, Elli Zavou, Mathieu Cunche, Françoise Fessant, Dominique Le Hello, Ulrich Matchi Aïvodji, Baptiste Olivier, Tony Quertier, and Razvan Stanica. 'Privacy in Trajectory Micro-Data Publishing: A Survey'. In: *Transactions on Data Privacy* 13.2 (2020), pp. 91–149. ISSN: 1888-5063. URL: `http://www.tdp.cat/issues16/abs.a363a19.php` (visited on 09/06/2023).

[114] M. Maurice Fréchet. 'Sur quelques points du calcul fonctionnel'. French. In: *Rendiconti del Circolo Matematico di Palermo* 22 (1906), pp. 1–72.

[115] Giovanni Fusco, Matteo Caglioni, Karine Emsellem, Myriam Merad, Diego Moreno, and Christine Voiron-Canicio. 'Questions of Uncertainty in Geography'. In: *Environment and Planning A: Economy and Space* 49.10 (2017), pp. 2261–2280. ISSN: 0308-518X. DOI: `10.1177/0308518X17718838`.

[116] Subir Kumar Ghosh. *Visibility Algorithms in the Plane*. Cambridge, UK: Cambridge University Press, 2007. ISBN: 978-0-521-87574-5. DOI: `10.1017/CBO9780511543340`.

[117] Michael Godau. 'A Natural Metric for Curves: Computing the Distance for Polygonal Chains and Approximation Algorithms'. In: *Proceedings of the 8th Annual Symposium on Theoretical Aspects of Computer Science (STACS 1991)*. Ed. by Christian Choffrut and Matthias Jantzen. Lecture Notes in Computer Science 480. Berlin, Germany: Springer, 1991, pp. 127–136. ISBN: 978-3-540-47002-1. DOI: `10.1007/BFb0020793`.

[118] Teofilo F. Gonzalez. 'Clustering to Minimize the Maximum Intercluster Distance'. In: *Theoretical Computer Science* 38 (1985), pp. 293–306. ISSN: 0304-3975. DOI: `10.1016/0304-3975(85)90224-5`.

[119] Michael F. Goodchild. 'How Well Do We Really Know the World?: Uncertainty in GIScience'. In: *Journal of Spatial Information Science* 20 (2020), pp. 97–102. ISSN: 1948-660X. DOI: `10.5311/JOSIS.2019.20.664`.

[120] Chris Gray, Frank Kammer, Maarten Löffler, and Rodrigo I. Silveira. 'Removing Local Extrema from Imprecise Terrains'. In: *Computational Geometry: Theory & Applications* 45.7 (2012), pp. 334–349. ISSN: 0925-7721. DOI: `10.1016/j.comgeo.2012.02.002`.

[121] Chris Gray, Maarten Löffler, and Rodrigo I. Silveira. 'Smoothing Imprecise 1.5D Terrains'. In: *International Journal of Computational Geometry & Applications* 20.4 (2010), pp. 381–414. ISSN: 0218-1959. DOI: `10.1142/S0218195910003359`.

[122] Paul D. Groves. 'Shadow Matching: A New GNSS Positioning Technique for Urban Canyons'. In: *Journal of Navigation* 64.3 (2011), pp. 417–430. ISSN: 0373-4633. DOI: `10.1017/S0373463311000087`.

[123] Joachim Gudmundsson, Jyrki Katajainen, Damian Merrick, Cahya Ong, and Thomas Wolle. 'Compressing Spatio-Temporal Trajectories'. In: *Computational Geometry: Theory & Applications* 42.9 (2009), pp. 825–841. ISSN: 0925-7721. DOI: `10.1016/j.comgeo.2009.02.002`.

[124] Joachim Gudmundsson, Marc J. van Kreveld, and Bettina Speckmann. 'Efficient Detection of Patterns in 2D Trajectories of Moving Points'. In: *GeoInformatica* 11.2 (2007), pp. 195–215. ISSN: 1384-6175. DOI: `10.1007/s10707-006-0002-z`.

[125] Joachim Gudmundsson, Majid Mirzanezhad, Ali Mohades, and Carola Wenk. 'Fast Fréchet Distance between Curves with Long Edges'. In: *International Journal of Computational Geometry & Applications* 29.2 (2019), pp. 161–187. ISSN: 0218-1959. DOI: `10.1142/S0218195919500043`.

[126] Joachim Gudmundsson and Pat Morin. 'Planar Visibility: Testing and Counting'. In: *Proceedings of the 26th Annual Symposium on Computational Geometry (SoCG 2010)*. Ed. by David G. Kirkpatrick and Joseph S. B. Mitchell. New York, NY, USA: Association for Computing Machinery, 2010, pp. 77–86. ISBN: 978-1-4503-0016-2. DOI: `10.1145/1810959.1810973`.

[127] Joachim Gudmundsson, Martin P. Seybold, and Sampson Wong. 'Map Matching Queries on Realistic Input Graphs Under the Fréchet Distance'. In: *Proceedings of the 34th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA 2023)*. Ed. by Nikhil Bansal and Viswanath Nagarajan. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2023, pp. 1464–1492. ISBN: 978-1-61197-755-4. DOI: `10.1137/1.9781611977554.ch53`.

[128] Joachim Gudmundsson and Michiel Smid. 'Fast Algorithms for Approximate Fréchet Matching Queries in Geometric Trees'. In: *Computational Geometry: Theory & Applications* 48.6 (2015), pp. 479–494. ISSN: 0925-7721. DOI: `10.1016/j.comgeo.2015.02.003`.

[129] Joachim Gudmundsson and Thomas Wolle. 'Football Analysis Using Spatio-Temporal Tools'. In: *Computers, Environment and Urban Systems* 47 (2014), pp. 16–27. ISSN: 0198-9715. DOI: `10.1016/j.compenvurbsys.2013.09.004`.

[130] Leonidas J. Guibas and John Hershberger. 'Optimal Shortest Path Queries in a Simple Polygon'. In: *Journal of Computer and System Sciences* 39.2 (1989), pp. 126–152. ISSN: 0022-0000. DOI: `10.1016/0022-0000(89)90041-X`.

[131] Leonidas J. Guibas, John Hershberger, Daniel Leven, Micha Sharir, and Robert E. Tarjan. 'Linear-Time Algorithms for Visibility and Shortest Path Problems inside Triangulated Simple Polygons'. In: *Algorithmica* 2 (1987), pp. 209–233. ISSN: 0178-4617. DOI: `10.1007/BF01840360`.

[132] Leonidas J. Guibas, John E. Hershberger, Joseph S. B. Mitchell, and Jack S. Snoeyink. 'Approximating Polygons and Subdivisions with Minimum-Link Paths'. In: *International Journal of Computational Geometry & Applications* 3.4 (1993), pp. 383–415. ISSN: 0218-1959. DOI: `10.1142/S0218195993000257`.

[133] Yingqi Guo, Cheuk-Yui Yeung, Geoff C. H. Chan, Qingsong Chang, Hector W. H. Tsang, and Paul S. F. Yip. 'Mobility Based on GPS Trajectory Data and Interviews: A Pilot Study to Understand the Differences between Lower- and Higher-Income Older Adults in Hong Kong'. In: *International Journal of Environmental Research and Public Health* 19.9, 5536 (2022). ISSN: 1660-4601. DOI: `10.3390/ijerph19095536`.

[134] Prosenjit Gupta, Ravi Janardan, Yokesh Kumar, and Michiel Smid. 'Data Structures for Range-Aggregate Extent Queries'. In: *Computational Geometry: Theory & Applications* 47.2 (Part C 2014), pp. 329–347. ISSN: 0925-7721. DOI: `10.1016/j.comgeo.2009.08.001`.

[135] Prosenjit Gupta, Ravi Janardan, and Michiel H. M. Smid. 'Further Results on Generalized Intersection Searching Problems: Counting, Reporting, and Dynamization'. In: *Journal of Algorithms* 19.2 (1995), pp. 282–317. ISSN: 0196-6774. DOI: 10.1006/jagm.1995.1038.

[136] Theodor Gutschlag and Sabine Storandt. 'On the Generalized Fréchet Distance and Its Applications'. In: *Proceedings of the 30th International Conference on Advances in Geographic Information Systems (SIGSPATIAL 2022)*. Ed. by Matthias Renz and Mohamed Sarwat. New York, NY, USA: Association for Computing Machinery, 2022, 35. ISBN: 978-1-4503-9529-8. DOI: 10.1145/3557915.3560970.

[137] Sariel Har-Peled and Benjamin Raichel. 'The Fréchet Distance Revisited and Extended'. In: *ACM Transactions on Algorithms* 10.1, 3 (2014). ISSN: 1549-6325. DOI: 10.1145/2532646.

[138] Mahdi Hashemi and Hassan A. Karimi. 'A Critical Review of Real-Time Map-Matching Algorithms: Current Issues and Future Directions'. In: *Computers, Environment and Urban Systems* 48 (2014), pp. 153–165. ISSN: 0198-9715. DOI: 10.1016/j.compenvurbsys.2014.07.009.

[139] Felix Hausdorff. *Grundzüge der Mengenlehre*. German. Leipzig, Germany: Von Veit & Comp., 1914.

[140] John Hershberger. 'A New Data Structure for Shortest Path Queries in a Simple Polygon'. In: *Information Processing Letters* 38.5 (1991), pp. 231–235. ISSN: 0020-0190. DOI: 10.1016/0020-0190(91)90064-O.

[141] John Hershberger and Subhash Suri. 'A Pedestrian Approach to Ray Shooting'. In: *Journal of Algorithms* 18.3 (1995), pp. 403–431. ISSN: 0196-6774. DOI: 10.1006/jagm.1995.1017.

[142] Ivor van der Hoog, Irina Kostitsyna, Maarten Löffler, and Bettina Speckmann. 'Preprocessing Ambiguous Imprecise Points'. In: *Proceedings of the 35th International Symposium on Computational Geometry (SoCG 2019)*. Ed. by Gill Barequet and Yusu Wang. Leibniz International Proceedings in Informatics 129. Wadern, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019, 42. ISBN: 978-3-95977-104-7. DOI: 10.4230/LIPIcs.SoCG.2019.42.

[143] Ivor van der Hoog, Irina Kostitsyna, Maarten Löffler, and Bettina Speckmann. 'Preprocessing Imprecise Points for the Pareto Front'. In: *Proceedings of the 33rd Annual ACM–SIAM Symposium on Discrete Algorithms (SODA 2022)*. Ed. by Joseph Naor and Niv Buchbinder. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2022, pp. 3144–3167. ISBN: 978-1-61197-707-3. DOI: 10.1137/1.9781611977073.122.

[144] Mara Hvistendahl. 'Foreigners Run Afoul of China's Tightening Secrecy Rules'. In: *Science* 339.6118 (2013), pp. 384–385. ISSN: 0036-8075. DOI: 10.1126/science.339.6118.384.

[145] Hiroshi Imai and Masao Iri. 'Computational-Geometric Methods for Polygonal Approximations of a Curve'. In: *Computer Vision, Graphics, and Image Processing* 36.1 (1986), pp. 31–41. ISSN: 0734-189X. DOI: 10.1016/S0734-189X(86)80027-5.

[146] Patrick Jaillet. 'Probabilistic Traveling Salesman Problems'. PhD thesis. Massachusetts Institute of Technology, 1985. URL: https://www.mit.edu/~jaillet/general/jaillet-phd-mit-orc-85.pdf (visited on 18/06/2023).

[147] Minghui Jiang, Ying Xu, and Binhai Zhu. 'Protein Structure: Structure Alignment With Discrete Fréchet Distance'. In: *Journal of Bioinformatics and Computational Biology* 6.1 (2008), pp. 51–64. ISSN: 0219-7200. DOI: 10.1142/s0219720008003278.

[148] Barry Joe and Richard B. Simpson. 'Corrections to Lee's Visibility Polygon Algorithm'. In: *BIT Numerical Mathematics* 27 (1987), pp. 458–473. ISSN: 0006-3835. DOI: 10.1007/BF01937271.

[149] Allan Jørgensen, Jeff M. Phillips, and Maarten Löffler. 'Geometric Computations on Indecisive Points'. In: *Proceedings of the 12th International Symposium on Algorithms and Data Structures (WADS 2011)*. Ed. by Frank Dehne, John Iacono, and Jörg-Rüdiger Sack. Lecture Notes in Computer Science 6844. Berlin, Germany: Springer, 2011, pp. 536–547. ISBN: 978-3-642-22299-3. DOI: 10.1007/978-3-642-22300-6_45.

[150] Vahideh Keikha, Sepideh Aghamolaei, Ali Mohades, and Mohammad Ghodsi. 'Clustering Geometrically-Modeled Points in the Aggregated Uncertainty Model'. In: *Fundamenta Informaticæ* 184.3 (2021), pp. 205–231. ISSN: 0169-2968. DOI: 10.3233/FI-2021-2097.

[151] Eamonn Keogh and Chotirat Ann Ratanamahatana. 'Exact Indexing of Dynamic Time Warping'. In: *Knowledge and Information Systems* 7.3 (2005), pp. 358–386. ISSN: 0219-1377. DOI: 10.1007/s10115-004-0154-9.

[152] Mees van de Kerkhof, Irina Kostitsyna, Maarten Löffler, Majid Mirzanezhad, and Carola Wenk. 'Global Curve Simplification'. In: *Proceedings of the 27th Annual European Symposium on Algorithms (ESA 2019)*. Ed. by Michael A. Bender, Ola Svensson, and Grzegorz Herman. Leibniz International Proceedings in Informatics 144. Wadern, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019, 67. ISBN: 978-3-95977-124-5. DOI: 10.4230/LIPIcs.ESA.2019.67.

[153] David G. Kirkpatrick. 'Optimal Search in Planar Subdivisions'. In: *SIAM Journal on Computing* 12.1 (1983), pp. 28–35. ISSN: 0097-5397. DOI: 10.1137/0212002.

[154] Christian Knauer, Maarten Löffler, Marc Scherfenberg, and Thomas Wolle. 'The Directed Hausdorff Distance between Imprecise Point Sets'. In: *Theoretical Computer Science* 412.32 (2011), pp. 4173–4186. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2011.01.039.

[155] Koninklijk Nederlands Meteorologisch Instituut. *Precipitation Radar Archive*. 2023. URL: https://www.knmi.nl/nederland-nu/klimatologie/geografische-overzichten/radar (visited on 10/07/2023).

[156] Manikanta Kotaru, Kiran Joshi, Dinesh Bharadia, and Sachin Katti. 'SpotFi: Decimeter Level Localization Using WiFi'. In: *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM 2015)*. Ed. by Steve Uhlig, Olaf Maennel, Brad Karp, and Jitendra Padhye. New York, NY, USA: Association for Computing Machinery, 2015, pp. 269–282. ISBN: 978-1-4503-3542-3. DOI: 10.1145/2785956.2787487.

[157] Bart Kranstauber, Roland Kays, Scott D. LaPoint, Martin Wikelski, and Kamran Safi. 'A Dynamic Brownian Bridge Movement Model to Estimate Utilization Distributions for Heterogeneous Animal Movement'. In: *Journal of Animal Ecology* 81.4 (2012), pp. 738–746. ISSN: 0021-8790. DOI: `10.1111/j.1365-2656.2012.01955.x`.

[158] Marc J. van Kreveld and Maarten Löffler. 'Approximating Largest Convex Hulls for Imprecise Points'. In: *Journal of Discrete Algorithms* 6.4 (2008), pp. 583–594. ISSN: 1570-8667. DOI: `10.1016/j.jda.2008.04.002`.

[159] Marc J. van Kreveld, Maarten Löffler, and Joseph S. B. Mitchell. 'Preprocessing Imprecise Points and Splitting Triangulations'. In: *SIAM Journal on Computing* 39.7 (2010), pp. 2990–3000. ISSN: 0097-5397. DOI: `10.1137/090753620`.

[160] Marc J. van Kreveld, Maarten Löffler, and Lionov Wiratma. 'On Optimal Polyline Simplification Using the Hausdorff and Fréchet Distance'. In: *Proceedings of the 34th International Symposium on Computational Geometry (SoCG 2018)*. Ed. by Bettina Speckmann and Csaba D. Tóth. Leibniz International Proceedings in Informatics 99. Wadern, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018, 56. ISBN: 978-3-95977-066-8. DOI: `10.4230/LIPIcs.SoCG.2018.56`.

[161] John Krumm. 'A Survey of Computational Location Privacy'. In: *Personal and Ubiquitous Computing* 13.6 (2009), pp. 391–399. ISSN: 1617-4909. DOI: `10.1007/s00779-008-0212-5`.

[162] John Krumm. 'Maximum Entropy Bridgelets for Trajectory Completion'. In: *Proceedings of the 30th International Conference on Advances in Geographic Information Systems (SIGSPATIAL 2022)*. Ed. by Matthias Renz and Mohamed Sarwat. New York, NY, USA: Association for Computing Machinery, 2022, 79. ISBN: 978-1-4503-9529-8. DOI: `10.1145/3557915.3561015`.

[163] Joseph B. Kruskal and Mark Liberman. 'The Symmetric Time-Warping Problem: From Continuous to Discrete'. In: *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Ed. by David Sankoff and Joseph B. Kruskal. Reading, MA, USA: Addison–Wesley, 1983, pp. 125–161. ISBN: 978-0-201-07809-1.

[164] Matej Kubicka, Arben Cela, Hugues Mounier, and Silviu-Iulian Niculescu. 'Comparative Study and Application-Oriented Classification of Vehicular Map-Matching Methods'. In: *IEEE Intelligent Transportation Systems Magazine* 10.2 (2018), pp. 150–166. ISSN: 1939-1390. DOI: `10.1109/MITS.2018.2806630`.

[165] Bart Kuijpers, Bart Moelans, Walied Othman, and Alejandro A. Vaisman. 'Uncertainty-Based Map Matching: The Space-Time Prism and *k*-Shortest Path Algorithm'. In: *ISPRS International Journal of Geo-Information* 5.11, 204 (2016). ISSN: 2220-9964. DOI: `10.3390/ijgi5110204`.

[166] Heikki Laitinen, Jaakko Lähteenmäki, and Tero Nordström. 'Database Correlation Method for GSM Location'. In: *Proceedings of the 53rd IEEE VTS Vehicular Technology Conference (VTC Spring)*. Vol. 4. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers, 2001, pp. 2504–2508. ISBN: 978-0-7803-6728-9. DOI: `10.1109/VETECS.2001.944052`.

[167] Hung Le and Cuong Than. ‘Greedy Spanners in Euclidean Spaces Admit Sublinear Separators’. In: *Proceedings of the 33rd Annual ACM–SIAM Symposium on Discrete Algorithms (SODA 2022)*. Ed. by Joseph Naor and Niv Buchbinder. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2022, pp. 3287–3310. ISBN: 978-1-61197-707-3. DOI: `10.1137/1.9781611977073.130`.

[168] Der-Tsai Lee. ‘Visibility of a Simple Polygon’. In: *Computer Vision, Graphics, and Image Processing* 22.2 (1983), pp. 207–221. ISSN: 0734-189X. DOI: `10.1016/0734-189X(83)90065-8`.

[169] Jian Li and Haitao Wang. ‘Range Queries on Uncertain Data’. In: *Theoretical Computer Science* 609 (2016), pp. 32–48. ISSN: 0304-3975. DOI: `10.1016/j.tcs.2015.09.005`.

[170] Richard J. Lipton and Robert Endre Tarjan. ‘A Separator Theorem for Planar Graphs’. In: *SIAM Journal on Applied Mathematics* 36.2 (1979), pp. 177–189. ISSN: 0036-1399. DOI: `10.1137/0136016`.

[171] Maarten Löffler. ‘Data Imprecision in Computational Geometry’. PhD thesis. Universiteit Utrecht, 2009. ISBN: 978-90-8891-121-7. URL: `https://dspace.library.uu.nl/bitstream/handle/1874/36022/loffler.pdf` (visited on 15/06/2023).

[172] Maarten Löffler. ‘Existence and Computation of Tours through Imprecise Points’. In: *International Journal of Computational Geometry & Applications* 21.1 (2011), pp. 1–24. ISSN: 0218-1959. DOI: `10.1142/S0218195911003524`.

[173] Maarten Löffler and Marc J. van Kreveld. ‘Largest and Smallest Tours and Convex Hulls for Imprecise Points’. In: *Algorithm Theory – SWAT 2006*. Ed. by Lars Arge and Rusins Freivalds. Lecture Notes in Computer Science 4059. Berlin, Germany: Springer, 2006, pp. 375–387. ISBN: 978-3-540-35753-7. DOI: `10.1007/11785293_35`.

[174] Maarten Löffler and Marc J. van Kreveld. ‘Largest Bounding Box, Smallest Diameter, and Related Problems on Imprecise Points’. In: *Computational Geometry: Theory & Applications* 43.4 (2010), pp. 419–433. ISSN: 0925-7721. DOI: `10.1016/j.comgeo.2009.03.007`.

[175] Maarten Löffler and Wolfgang Mulzer. ‘Unions of Onions: Preprocessing Imprecise Points for Fast Onion Decomposition’. In: *Journal of Computational Geometry* 5.1 (2014), pp. 1–13. ISSN: 1920-180X. DOI: `10.20382/jocg.v5i1a1`.

[176] Maarten Löffler and Jeff M. Phillips. ‘Shape Fitting on Point Sets with Probability Distributions’. In: *Algorithms – ESA 2009*. Ed. by Amos Fiat and Peter Sanders. Lecture Notes in Computer Science 5757. Berlin, Germany: Springer, 2009, pp. 313–324. ISBN: 978-3-642-04128-0. DOI: `10.1007/978-3-642-04128-0_29`.

[177] Maarten Löffler and Jack Scott Snoeyink. ‘Delaunay Triangulations of Imprecise Points in Linear Time after Preprocessing’. In: *Computational Geometry: Theory & Applications* 43.3 (2010), pp. 234–242. ISSN: 0925-7721. DOI: `10.1016/j.comgeo.2008.12.007`.

[178] Paul A. Longley, Michael F. Goodchild, David J. Maguire, and David W. Rhind. *Geographic Information Systems and Science*. 4th ed. Hoboken, NJ, USA: John Wiley & Sons, 2015. ISBN: 978-1-119-12845-8.

[179] Anil Maheshwari, Jörg-Rüdiger Sack, Kaveh Shahbaz, and Hamid Zarrabi-Zadeh. 'Fréchet Distance with Speed Limits'. In: *Computational Geometry: Theory & Applications* 44.2 (2011), pp. 110–120. ISSN: 0925-7721. DOI: `10.1016/j.comgeo.2010.09.008`.

[180] Rupak Majumdar and Vinayak S. Prabhu. 'Computing the Skorokhod Distance between Polygonal Traces'. In: *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control (HSCC 2015)*. Ed. by Antoine Girard and Sriram Sankaranarayanan. New York, NY, USA: Association for Computing Machinery, 2015, pp. 199–208. ISBN: 978-1-4503-3433-4. DOI: `10.1145/2728606.2728618`.

[181] Avraham Melkman and Joseph O'Rourke. 'On Polygonal Chain Approximation'. In: *Machine Intelligence and Pattern Recognition*. Vol. 6: *Computational Morphology: A Computational Geometric Approach to the Analysis Of Form*. Ed. by Godfried T. Toussaint. Amsterdam, the Netherlands: Elsevier, 1988, pp. 87–95. ISBN: 978-0-444-70467-2. DOI: `10.1016/B978-0-444-70467-2.50012-6`.

[182] Àlex Miranda-Pascual, Patricia Guerra-Balboa, Javier Parra-Arnau, Jordi Forné, and Thorsten Strufe. 'SoK: Differentially Private Publication of Trajectory Data'. In: *Proceedings on Privacy Enhancing Technologies* 2023.2 (2023), pp. 496–516. ISSN: 2299-0984. DOI: `10.56553/popets-2023-0065`.

[183] Paul Newson and John Krumm. 'Hidden Markov Map Matching through Noise and Sparseness'. In: *Proceedings of the 17th International Conference on Advances in Geographic Information Systems (SIGSPATIAL 2009)*. Ed. by Divyakant Agrawal, Walid G. Aref, Chang-Tien Lu, Mohamed F. Mokbel, Peter Scheuermann, Cyrus Shahabi, and Ouri Wolfson. New York, NY, USA: Association for Computing Machinery, 2009, pp. 336–343. ISBN: 978-1-60558-649-6. DOI: `10.1145/1653771.1653818`.

[184] Stig Nordbeck. 'Computing Distances in Road Networks'. In: *Papers in Regional Science* 12.1 (1964), pp. 207–220. ISSN: 1435-5957. DOI: `10.1111/j.1435-5597.1964.tb01266.x`.

[185] NordNordWest. *Spreading of Homo Sapiens*. 2023. URL: `https://commons.wikimedia.org/wiki/File:Spreading_homo_sapiens_la.svg` (visited on 10/07/2023).

[186] Joseph O'Rourke. *Art Gallery Theorems and Algorithms*. International Series of Monographs on Computer Science 3. Oxford, UK: Oxford University Press, 1987. ISBN: 978-0-19-503965-8. URL: `http://www.science.smith.edu/~jorourke/books/ArtGalleryTheorems/art.html` (visited on 08/07/2023).

[187] Joseph O'Rourke. 'Visibility'. In: *Handbook of Discrete and Computational Geometry*. Ed. by Jacob E. Goodman, Joseph O'Rourke, and Csaba D. Tóth. 3rd ed. Discrete Mathematics and Its Applications. Boca Raton, FL, USA: CRC Press, 2017, pp. 875–896. ISBN: 978-1-4987-1139-5. DOI: `10.1201/9781315119601`.

[188] OpenStreetMap. *Map Data*. 2023. URL: `https://openstreetmap.org` (visited on 03/03/2023).

[189] Mark H. Overmars and Emo Welzl. 'New Methods for Computing Visibility Graphs'. In: *Proceedings of the 4th Annual Symposium on Computational Geometry (SoCG 1988)*. Ed. by Herbert Edelsbrunner. New York, NY, USA: Association for Computing Machinery, 1988, pp. 164–171. ISBN: 978-0-89791-270-9. DOI: 10.1145/73393.73410.

[190] Jian Pei, Bin Jiang, Xuemin Lin, and Yidong Yuan. 'Probabilistic Skylines on Uncertain Data'. In: *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB 2007)*. Ed. by Christoph Koch et al. Los Angeles, CA, USA: VLDB Endowment, 2007, pp. 15–26. ISBN: 978-1-59593-649-3. URL: https://dl.acm.org/doi/10.5555/1325851.1325858 (visited on 15/06/2023).

[191] Dieter Pfoser and Christian S. Jensen. 'Capturing the Uncertainty of Moving-Object Representations'. In: *Proceedings of the 6th International Symposium on Spatial Databases (SSD 1999)*. Ed. by Ralf Hartmut Güting, Dimitris Papadias, and Fred Lochovsky. Lecture Notes in Computer Science 1651. Berlin, Germany: Springer, 1999, pp. 111–131. ISBN: 978-3-540-66247-1. DOI: 10.1007/3-540-48482-5_9.

[192] Urs Ramer. 'An Iterative Procedure for the Polygonal Approximation of Plane Curves'. In: *Computer Graphics and Image Processing* 1.3 (1972), pp. 244–256. ISSN: 0146-664X. DOI: 10.1016/S0146-664X(72)80017-0.

[193] Lucas Rodríguez, Javier Palanca, Elena del Val, and Miguel Rebollo. 'Analyzing Urban Mobility Paths Based on Users' Activity in Social Networks'. In: *Future Generation Computer Systems* 102 (2020), pp. 333–346. ISSN: 0167-739X. DOI: 10.1016/j.future.2019.07.072.

[194] Hiroaki Sakoe and Seibi Chiba. 'Dynamic Programming Algorithm Optimization for Spoken Word Recognition'. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26.1 (1978), pp. 43–49. ISSN: 0096-3518. DOI: 10.1109/TASSP.1978.1163055.

[195] David Salesin, Jorge Stolfi, and Leonidas J. Guibas. 'Epsilon Geometry: Building Robust Algorithms from Imprecise Computations'. In: *Proceedings of the 5th Annual Symposium on Computational Geometry (SCG 1989)*. Ed. by Kurt Mehlhorn. New York, NY, USA: Association for Computing Machinery, 1989, pp. 208–217. ISBN: 978-0-89791-318-8. DOI: 10.1145/73833.73857.

[196] Rock Santerre. 'Impact of GPS Satellite Sky Distribution'. In: *Manuscripta Geodætica* 16 (1991), pp. 28–53. HDL: 20.500.11794/12354.

[197] Jaume Sanz Subirana, José Miguel Juan Zornoza, and Manuel Hernández-Pajares. *GNSS Data Processing*. Vol. 1: *Fundamentals and Algorithms*. ESA Training Manuals TM-23/1. Noordwijk, the Netherlands: ESA Communications, 2013. ISBN: 978-92-9221-886-7. URL: https://www.esa.int/About_Us/ESA_Publications/ESA_TM-23_GNSS_DATA_PROCESSING (visited on 09/06/2023).

[198] Neil Sarnak and Robert E. Tarjan. 'Planar Point Location Using Persistent Search Trees'. In: *Communications of the ACM* 29.7 (1986), pp. 669–679. ISSN: 0001-0782. DOI: 10.1145/6138.6151.

[199] Otfried Schwarzkopf and Jules Vleugels. 'Range Searching in Low-Density Environments'. In: *Information Processing Letters* 60.3 (1996), pp. 121–127. ISSN: 0020-0190. DOI: 10.1016/S0020-0190(96)00154-8.

[200] Jeff Sember and William S. Evans. *Guaranteed Voronoi Diagrams of Uncertain Sites*. Presented at CCCG 2008, Montreal, QC, Canada. 2008. URL: `http://cccg.ca/proceedings/2008/paper50full.pdf` (visited on 09/05/2023).

[201] Martin P. Seybold. 'Robust Map Matching for Heterogeneous Data via Dominance Decompositions'. In: *Proceedings of the 2017 SIAM International Conference on Data Mining (SDM 2017)*. Ed. by Nitesh Chawla and Wei Wang. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2017, pp. 813–821. ISBN: 978-1-61197-497-3. DOI: `10.1137/1.9781611974973.91`.

[202] Aravinda Prasad Sistla, Ouri Wolfson, Sam Chamberlain, and Son Dao. 'Querying the Uncertain Position of Moving Objects'. In: *Temporal Databases: Research and Practice*. Ed. by Opher Etzion, Sushil Jajodia, and Suryanarayana Sripada. Lecture Notes in Computer Science 1399. Berlin, Germany: Springer, 1998, pp. 310–337. ISBN: 978-3-540-64519-1. DOI: `10.1007/BFb0053708`.

[203] Tim Sodergren, Jessica Hair, Jeff M. Phillips, and Bei Wang. *Visualizing Sensor Network Coverage with Location Uncertainty*. 2017. arXiv: `1710.06925v1 [cs.HC]`.

[204] A. Frank van der Stappen. 'Motion Planning Amidst Fat Obstacles'. PhD thesis. Universiteit Utrecht, 1994. ISBN: 978-90-393-0654-3. URL: `https://webspace.science.uu.nl/~stapp101/PhDThesis_AFvanderStappen.pdf` (visited on 07/07/2023).

[205] Han Su, Shuncheng Liu, Bolong Zheng, Xiaofang Zhou, and Kai Zheng. 'A Survey of Trajectory Distance Measures and Performance Evaluation'. In: *The VLDB Journal* 29 (2020), pp. 3–32. ISSN: 1066-8888. DOI: `10.1007/s00778-019-00574-9`.

[206] Subhash Suri and Joseph O'Rourke. 'Worst-Case Optimal Algorithms for Constructing Visibility Polygons with Holes'. In: *Proceedings of the 2nd Annual Symposium on Computational Geometry (SoCG 1986)*. Ed. by Alok Aggarwal. New York, NY, USA: Association for Computing Machinery, 1986, pp. 14–23. ISBN: 978-0-89791-194-8. DOI: `10.1145/10515.10517`.

[207] Subhash Suri and Kevin Verbeek. 'On the Most Likely Voronoi Diagram and Nearest Neighbor Searching'. In: *International Journal of Computational Geometry & Applications* 26.3–4 (2016), pp. 151–166. ISSN: 0218-1959. DOI: `10.1142/S0218195916600025`.

[208] Subhash Suri, Kevin Verbeek, and Hakan Yıldız. 'On the Most Likely Convex Hull of Uncertain Points'. In: *Algorithms – ESA 2013*. Ed. by Hans L. Bodlaender and Giuseppe F. Italiano. Lecture Notes in Computer Science 8125. Berlin, Germany: Springer, 2013, pp. 791–802. ISBN: 978-3-642-40449-8. DOI: `10.1007/978-3-642-40450-4_67`.

[209] Yaguang Tao, Alan Both, Rodrigo I. Silveira, Kevin Buchin, Stef Sijben, Ross S. Purves, Patrick Laube, Dongliang Peng, Kevin Toohey, and Matt Duckham. 'A Comparative Analysis of Trajectory Similarity Measures'. In: *GIScience & Remote Sensing* 58.5 (2021), pp. 643–669. ISSN: 1548-1603. DOI: `10.1080/15481603.2021.1908927`.

[210] The CGAL Project. *Computational Geometry Algorithms Library*. 2023. URL: `https://www.cgal.org` (visited on 10/07/2023).

[211] Michail Vlachos, George Kollios, and Dimitrios Gunopulos. 'Elastic Translation Invariant Matching of Trajectories'. In: *Machine Learning* 58 (2005), pp. 301–334. ISSN: 0885-6125. DOI: 10.1007/s10994-005-5830-9.

[212] Hong Wei, Yin Wang, George Forman, and Yanmin Zhu. 'Map Matching: Comparison of Approaches Using Sparse and Noisy Data'. In: *Proceedings of the 21st International Conference on Advances in Geographic Information Systems (SIGSPATIAL 2013)*. Ed. by Craig Knoblock, Peer Kröger, John Krumm, Markus Schneider, and Peter Widmayer. New York, NY, USA: Association for Computing Machinery, 2013, pp. 444–447. ISBN: 978-1-4503-2521-9. DOI: 10.1145/2525314.2525456.

[213] Zhenzhou Xu, Ge Cui, Ming Zhong, and Xin Wang. 'Anomalous Urban Mobility Pattern Detection Based on GPS Trajectories and POI Data'. In: *ISPRS International Journal of Geo-Information* 8.7, 308 (2019). ISSN: 2220-9964. DOI: 10.3390/ijgi8070308.

[214] Lin Yan, Yusu Wang, Elizabeth Munch, Ellen Gasparovic, and Bei Wang. 'A Structural Average of Labeled Merge Trees for Uncertainty Visualization'. In: *IEEE Transactions on Visualization and Computer Graphics* 26.1 (2020), pp. 832–842. ISSN: 1077-2626. DOI: 10.1109/TVCG.2019.2934242.

[215] Man Lung Yiu, Nikos Mamoulis, Xiangyuan Dai, Yufei Tao, and Michail Vaitis. 'Efficient Evaluation of Probabilistic Advanced Spatial Queries on Existentially Uncertain Data'. In: *IEEE Transactions on Knowledge and Data Engineering* 21.1 (2009), pp. 108–122. ISSN: 1041-4347. DOI: 10.1109/TKDE.2008.135.

[216] Jianbin Zheng, Xiaolei Gao, Enqi Zhan, and Zhangcan Huang. 'Algorithm of On-Line Handwriting Signature Verification Based on Discrete Fréchet Distance'. In: *Proceedings of the 3rd International Symposium on Intelligence Computation and Applications (ISICA 2008)*. Ed. by Lishan Kang, Zhihua Cai, Xuesong Yan, and Yong Liu. Lecture Notes in Computer Science 5370. Berlin, Germany: Springer, 2008, pp. 461–469. ISBN: 978-3-540-92136-3. DOI: 10.1007/978-3-540-92137-0_51.

[217] Yu Zheng and Xiaofang Zhou, eds. *Computing with Spatial Trajectories*. Berlin, Germany: Springer, 2011. ISBN: 978-1-4614-1628-9. DOI: 10.1007/978-1-4614-1629-6.

Summary

Algorithms for Imprecise Trajectories

Movement data is ubiquitous, and measurements made with even inexpensive trackers can provide invaluable insight into behaviours of pedestrians or drivers, but also wildlife and any other moving subjects. Analysing movement data is thus of great societal importance. However, measurements made by most trackers have an inherent imprecision that may affect the results of analyses. Sparse measurements are another major source of uncertainty, since we do not explicitly know the subject's location between measurements. In this thesis, we study the algorithmic foundations of handling movement data under uncertainty. The goal is to design efficient algorithms with provable guarantees on their running time and correctness that transparently incorporate movement data uncertainty. The focus of our work lies on trajectories with a specific type of uncertainty: measurement imprecision.

In particular, we study similarity of imprecise trajectories under an established distance metric called the Fréchet distance. Computing similarity is a fundamental problem, applied as a building block in many analysis tasks. We consider different variants of the problem, depending on the desired way of modelling imprecision. We also show results for the common variants of the Fréchet distance, namely, the discrete Fréchet distance and the weak (discrete) Fréchet distance. These variants capture slightly different notions of similarity than the Fréchet distance. We study the problem for measured locations in one and two dimensions. In all of these settings, we either indicate that the problem is likely computationally difficult by showing NP-hardness; or we provide efficient algorithms, generally using dynamic programming and greedy approaches.

We also study the problem of simplifying an imprecise trajectory. For precise trajectories, typically, we select a subsequence of measurements so that the selection describes the full trajectory well in terms of some similarity measure. In our work, the goal is to select a subsequence of the imprecise locations so that for any true trajectory allowed by the uncertainty model, the selection describes a valid simplification of the true trajectory. We consider several ways to model imprecision and show how to use both the Fréchet and the Hausdorff distance within the method. In the settings we consider, we provide efficient algorithms for the problem.

Taking a different view of imprecision, we study map matching, where a measured trajectory is imprecise, but can be assumed to correspond to a path on, say, a road network. In this setting, we show how to preprocess a realistic road network so that given an arbitrary query trajectory, we can efficiently find an approximate path on the network that is closest to the query in terms of the Fréchet distance.

Finally, we consider the problem of visibility between sets of points or line segments inside a polygon, showing how to preprocess the polygon so that we can efficiently count how many fixed entities can be seen from a query object. We consider several extensions, including to entities modelled as polygons of constant complexity, and we show how to use our data structure to efficiently count pairwise visibility between two sets of entities. While not directly applicable for imprecise trajectories, this work is a step towards studying visibility under imprecision.

Curriculum Vitae

Aleksandr Popov was born on 3 October 1995 in Saint Petersburg, Russia. He completed his secondary education at Saint-Petersburg Gymnasium Alma Mater, Russia in 2012. He then followed courses in Information Security at National Research University ITMO in Saint Petersburg, Russia until 2014. He studied Computer Science and Engineering at Eindhoven University of Technology, obtaining his bachelor's degree (cum laude) in 2017 and his master's degree (cum laude) in 2019. He continued there as a PhD student; the main results of his research during that time are presented in this thesis.