Clustering Trajectories: Detecting Patterns in Spatio-Temporal Data

DISSERTATION

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Jacobus Conradi

aus

Grevenbroich, Deutschland

Bonn 2025

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

Gutachterin/Betreuerin:	Prof. Dr. Anne Driemel
Gutachter:	Prof. Dr. Heiko Röglin
Gutachter:	Prof. Dr. Pankaj Agarwal

Tag der Promotion:	28. November 2025
Erscheinungsjahr:	2025

Abstract

In this thesis, we study the detection of patterns in spatio-temporal data sets in the form of trajectory data. Many application areas collect unstructured trajectory data. Applications for the analysis of such data range from gait analysis of human movement to traffic-flow analysis, epidemiological hotspot detection, and Lagrangian analysis of particle simulations. Trajectory data is typically modeled as a sequence of points that defines a polygonal curve via linear interpolation between consecutive points. One of the most fundamental ways of detecting patterns in any kind of data is clustering, which aims to partition the data into groups where points from the same group are 'close' to one another and those from different groups are 'far apart'. The clustering of polygonal curves is the main focus of this thesis.

The first clustering problem we study is the (k, ℓ)-median problem for polygonal curves, which asks for k center curves, each with at most ℓ points, minimizing the sum of dynamic time warping (DTW) distances from each input curve to its closest center. Numerous classical results for clustering problems have been extended to many different settings. A common assumption made is that the distance function is a metric, which DTW is not. Despite this, we provide a polynomial-time approximation algorithm with an approximation factor of $\mathcal{O}(m\ell)$, where m is the maximum number of points in any input curve. This is achieved by extending known algorithms for coreset construction—i.e., problem-specific input condensations—from metric spaces to the non-metric space of polygonal curves under DTW.

The second clustering problem we consider is Subtrajectory Covering, motivated by the need to detect patterns *within* a single curve by segmenting it into its constituent semantic parts. This is formulated as a set cover problem based on the continuous Fréchet distance. Given a polygonal curve with n points, the goal is to compute the smallest set of center curves, each with at most ℓ points, such that every point on the input curve is covered by some center. A point is considered covered if it lies on a subtrajectory of the input curve whose (continuous) Fréchet distance to some center curve is at most Δ. We provide bicriteria approximation algorithms for arbitrary curves as well as for a subclass of realistic curves. Both algorithms approximate the required number k_Δ of centers by a factor of $\mathcal{O}(\log n)$ and the distance threshold Δ by a factor of $\mathcal{O}(1)$. For arbitrary curves, the algorithm runs in roughly $\mathcal{O}(\sqrt{k_\Delta} n^{5/2})$ time. For the subclass of realistic curves (or more formally, c-packed curves whose arc length inside any ball of radius r is bounded by cr), the algorithm runs in roughly $\mathcal{O}(k_\Delta c n^2)$ time. We demonstrate the viability of both the problem formulation and the algorithm on real-world data.

Additionally, we investigate two related questions. First, given a clustering of trajectory data, a natural question is how to classify a new trajectory which is not part of the input. To this end, we design a data structure that answers approximate nearest neighbor queries: Given n input curves of complexity m and a query curve q, compute the closest input curve to q w.r.t. the continuous Fréchet distance. Our data structure answers such queries in roughly $\mathcal{O}(F(m, \Phi) \log n)$ time, where $F(m, \Phi)$ depends only on m and the spread Φ of vertices and edges. Second, we analyze the detection of erroneous patterns within a curve: Given two curves, how can we optimally modify one curve by replacing pieces of the curve with line segments so as to minimize the continuous Fréchet distance to the other curve? We show that this problem is not fixed-parameter tractable in the number of modifications and give a polynomial-time $\mathcal{O}(1)$-approximation algorithm for its decision variant.

Acknowledgments

I would like to take this opportunity to express my gratitude to everyone who accompanied and supported me throughout my scientific journey over the past several years. This thesis would not have been possible without your guidance, encouragement, and friendship.

First and foremost, I am deeply grateful to my supervisor, Anne Driemel, for her invaluable mentorship and unwavering support. Her door was always open—whether to discuss ideas, provide feedback, or simply talk through challenges. Her guidance, encouragement, and numerous contributions shaped every stage of my research. She made it possible for me to participate in many conferences and workshops, which allowed me to grow as a researcher, and to connect with researchers from around the world. I could not have wished for a better supervisor.

I am equally thankful to my coauthors—Mikkel Abrahamsen, Maren Bennewitz, Lotte Blank, Florestan Brunck, Frederik Brüning, Nathan Corral, Jorge de Heuvel, Benedikt Kolbe, Benedikt Kreis, André Nusser, Ioannis Psarros, Marena Richter, and Dennis Rohde—for their collaboration, discussions, and input. I also thank the anonymous external reviewers of our manuscripts for their constructive feedback and suggestions.

Moreover, I would like to thank all members of the Algorithms and Complexity Group at the University of Bonn. Beyond the wonderful work environment, you provided an enjoyable and supportive daily atmosphere—from shared lunches and coffee breaks to after-work beers. A special thanks goes to Benedikt Kolbe, whose generosity with his time and thoughts was a constant source of support. Whether discussing research or life in general, his advice was always thoughtful, and his perspective helped me grow both professionally and personally. I also want to thank his family—Lena, Yuna, Linus, and Fiona—for their hospitality and for providing an outlet for my childish energy during our many cooking sessions.

To my family—my siblings Carolus, Martinus, and Henricus and most of all my parents Andreas and Jessica—as well as my friends: thank you for your love, encouragement, and belief in me. Your support formed the foundation upon which all of this was built. I am also grateful to the members of the 'Collegium Musicum Bonn' and the 'Akademisches Blasorchester Bonn', who made my final years as a student particularly amazing. Finally, I wish to thank Pascal Beyer, Max Kanold, and Karsten Evers of 'No Ducks Given' for their unconditional support, for always having my back and pushing me to achieve more than I could have ever imagined—you are truly awesome.

Contents

Chapter 1

Introduction

Recent technological developments have enabled affordable and widely available hardware capable of tracking a vast range of phenomena—from everyday human activity to large-scale environmental and biological systems. These range from smartwatches and smartphones collecting geographical and physiological data, such as location, temperature, heart rate, and blood oxygen levels, to scientific instruments tracking the movement of terrestrial and aquatic animals, ocean currents, and sea temperatures. The result is a huge amount of temporal data points across a wide range of domains. These datasets are analyzed with respect to various quality measures, such as estimating physiological markers based on heart rate variability [RAPJK+06], analyzing human gait in locomotion studies [CS08], estimating migratory patterns of individual animals [KSSF22], predicting human motion in multimodal settings [ZAFG21], and assessing the effects of climate change on ocean currents and marine ecosystems [WFH+16].

Most of these applications model their data as spatio-temporal sequences, where each data point is recorded in some ambient space over time. This includes GPS tracking data, where each point is a geographical coordinate; motion capture data, where each point is a set of three-dimensional joint coordinates; and heartbeat data, where each data point indicates the electrical intensity of the heart's contraction. We refer to such sequences of points as curves.

With the size of the underlying data often being large, there is a growing need to automate the analysis of these data sets. An important theoretical measure of such an automated process, or algorithm, is the notion of the asymptotic running time, which measures the number of basic operations on the data points the algorithm performs in the worst-case. Similar to the running time of an algorithm, we are also interested in the analysis of the quality of its output. Does the algorithm produce the best possible solution to the given problem, or does it only produce a rough estimate of the best possible solution? The ratio between the worst-case output and the optimal solution is the so-called approximation ratio. Alongside the running time, the approximation ratio plays a key role when analyzing algorithms and will be a recurring concept throughout this thesis.

We investigate foundational algorithmic problems related to processing data sets consisting of curves. The overarching goal in Chapter 3 and Chapter 4 is the transformation of a large collection of curves into a smaller, representative set that still preserves essential characteristics of the original data. This broad objective touches on core problems in classical machine learning, such as clustering [XW05], where the task is to find a small number of representative points that minimize the distance to the input data. Another

fundamental task that we consider in Chapter 5 is the construction of nearest neighbor data structures, where, given a set of representatives and a query point, the aim is to find the closest representative [KS13].

In the classical Euclidean setting, the distance measure (i.e., the Euclidean distance) is well-understood [OBS94]. This understanding does not translate easily to curves. There are different distance measures for curves used in practice, each with its own advantages and disadvantages, and limited understanding of them is the main roadblock to designing algorithms that mimic the qualities of state-of-the-art algorithms in the classical setting.

Two distance measures frequently used in practice—and which will be central to this thesis—are the Fréchet distance and the dynamic time warping distance (DTW). While both will be formally defined in Section 2.1, we provide an informal overview below.

Both the Fréchet distance and DTW are often introduced via the following man-and-dog metaphor: Imagine two curves, with a man walking along one and a dog, on a leash, walking along the other. Both start at the beginning of their respective curves and continuously move along their curve with varying speeds without backtracking. The *(continuous) Fréchet distance* is the length of the shortest leash required to allow both to reach the end of their respective curves—that is, the Fréchet distance is the minimum possible value of the maximum distance between man and dog at any point during their traversal. If the man and the dog are instead restricted to discrete movements, where in each discrete time step they either jump from one point to the next, or remain in place, the resulting measure is known as the *discrete Fréchet distance*. In contrast, the *dynamic time warping distance* measures the *average* distance between the man and the dog throughout such a restricted traversal. It captures the minimal possible average distance over all valid man-dog alignments constrained to input coordinates.

There are different extensions of the classical machine learning problem of clustering input points to inputs consisting of curves. The most straightforward extension we tackle in this thesis is the following:

Q 1: *Given a set of curves, can we compute few representative curves such that any input curve is close to at least one representative?*

Here, we extend clustering to curves by each curve acting like a point in the classical formulation, with distances being defined via the Fréchet distance or DTW.

A property curves exhibit, which points do not, is the fact that small pieces of curves still hold information. Consider, for example, the set of curves shown in the upper half of Figure 1.1, which track the ocean currents [LC19]. Each curve is the result of an ocean surface drifter over the period of up to two years. Often, it is natural to segment such curves into smaller pieces, especially when looking for local features, such as currents or ocean pathways, rather than global features of the data set. This motivates a different extension to clustering which we answer:

Q 2: *Given a set of curves, can we segment the curves into pieces and compute few representative curves such that any piece is close to at least one representative curve?*

The main difference between Question 1 and Question 2 is that for the latter, it is not apparent where the curve pieces start and end. As it turns out, this is also the main crux when trying to design algorithms to answer this question. Despite this, we show that Question 2 can be answered by designing efficient algorithms that, when given the aforementioned set of curves, produce the representative curves shown in Figure 1.1.

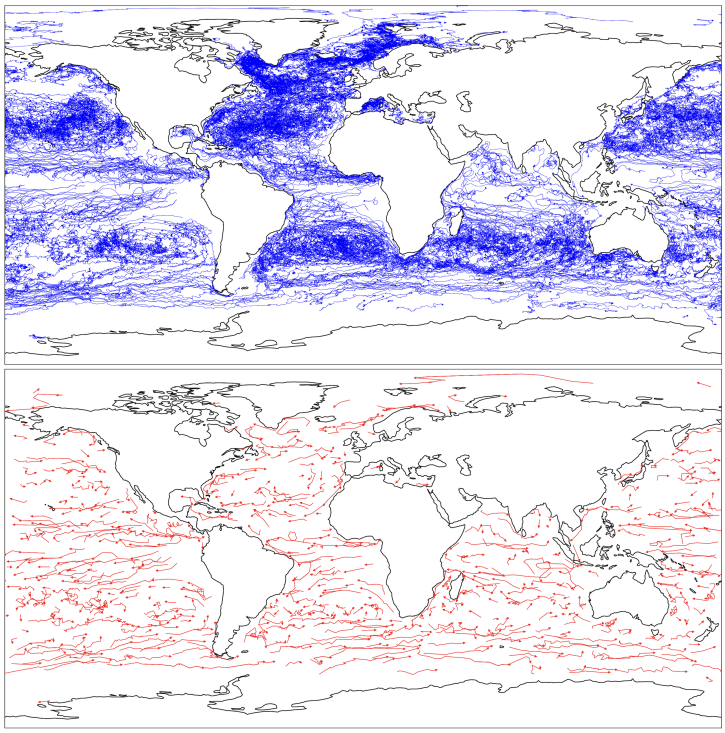


Figure 1.1: Illustration of roughly 2000 ocean surface drifters (blue) and a set of representative curves (red).

Computing a clustering of input data is usually just one of the first steps in some algorithmic pipeline. The computed set of representative curves should be made accessible to different types of queries. This can then be used to answer questions about the underlying data set. One such basic query is to compute the nearest neighbor to some query curve:

Q 3: *Given a set of curves, can we preprocess them to compute, for a given query curve, the input curve closest to the query curve?*

Questions 1–3 are concerned with detecting important patterns in the underlying data set of curves. In some applications, it may be necessary to detect and *discard* less important patterns instead. This may be the case when the data was generated by a faulty sensor, or certain assumptions on the input result in 'detours' that one would like to explicitly ignore. As the Fréchet distance is a bottleneck distance, it can be distorted by even a single outlier or detour. One extension to the Fréchet distance, which aims to measure the distance between two curves without being influenced by outliers or detours, does so by allowing simple modifications to one of the input curves, replacing pieces of the curve with a single straight edge, referred to as a *shortcut*. This gives rise to the *shortcut Fréchet distance*: Given two curves, what is the smallest Fréchet distance achievable between the two curves if one may be modified by shortcuts? Similar to Question 2, this poses the intermediate question of where optimal shortcuts should start and end. Overall, the algorithmic challenge which we aim to answer is the following:

Q 4: *Given two curves, can we efficiently compute their shortcut Fréchet distance when at most k shortcuts are allowed?*

In the following sections, we elaborate on the questions posed in Question 1 to 4, outline how they are addressed in this thesis and summarize the contributions made by the author of this thesis. Each chapter thereafter focuses on Question 1 to 4 as follows:

- Chapter 3 investigates coreset constructions for a k-median-like problem for curves under DTW, relating to Question 1,
- Chapter 4 addresses two problems called Subtrajectory Covering and Subtrajectory Coverage Maximization, relating to Question 2,
- Chapter 5 explores approximate nearest neighbor data structures for curves under the continuous Fréchet distance, relating to Question 3, and
- Chapter 6 considers the computability of the Fréchet distance with shortcuts, relating to Question 4.

1.1 Coresets for k-Median of Curves

Clustering is arguably one of the most fundamental problems in theoretical computer science and computational geometry. Broadly speaking, clustering deals with the identification of structure in a set of data points. A powerful approach to clustering problems involving massive data sets is data reduction, and the construction of ε-coresets is an approach that has received substantial attention. An ε-coreset is a problem-specific condensate of the given input set. It captures the core properties of the underlying input set w.r.t. the problem at hand and can be used as a proxy set for slower algorithms producing solutions with a small relative error of $(1 \pm \varepsilon)$.

Clustering in general, and the k-median problem in particular, represent fundamental computational problems that have been extensively studied in many classical settings. With the growing availability of geospatial tracking data, clustering problems for curves have received growing attention both from a theoretical and applied perspective. In practice, curve classification largely relies on the dynamic time warping (DTW) distance and is widely used in the area of data mining. Simple nearest neighbor classifiers under DTW are considered hard to beat [Kat16, TLNH19], and much effort has been put into making classification using DTW computationally efficient [JJO11, PFW$^+$14, PFW$^+$16, RCM$^+$13]. In contrast to its cousin, the Fréchet distance, DTW is less sensitive to outliers, but is also less well understood, owing to the fact that it is not a metric. In particular, the wealth of research surrounding k-median clustering for metric spaces does not directly apply to clustering problems under DTW.

For curves, k-median is often studied in the form of the (k, ℓ)-median problem, where the sought-after center curves are restricted to have a complexity (i.e., number of points in the sequence) of at most ℓ [DKS16, BDvG$^+$22].

Results In Chapter 3, we give the first construction of ε-coresets for the (k, ℓ)-median problem under DTW. We adapt the framework of sensitivity sampling introduced in [FL11] to our setting, develop fast approximation algorithms solving (k, ℓ)-median clustering under DTW, and use coresets to improve existing (k, ℓ)-median algorithms for curves under DTW.

Our approach relies on approximations of nearly all objects involved in our inquiry, but most importantly on curve simplifications and an approximation of DTW based on

its shortest-path metric. This allows us to apply state-of-the-art techniques for the k-median problem in this metric space, circumventing the use of heavy k-median machinery in non-metric spaces which would incur exponential dependence on k [BDvG$^+$22]. Our main ingredient is a new insight into the notion of relaxed triangle inequalities for DTW. We then construct a coreset based on the approximation algorithm. For this, we bound the so-called 'sensitivity' of the elements of the given data set, as well as their sum. An element's sensitivity measures its importance and determines its sample probability in the coreset construction. We construct an ε-coreset for the (k, ℓ)-median clustering problem, with size quadratic in $1/\varepsilon$ and k, logarithmic in the number n of input curves, and depending on the complexity m of the input curves as well as ℓ. We achieve this by upper bounding the VC dimension of a range space defined by all approximate DTW balls, with the bound depending logarithmically on m. Informally, the VC dimension measures the largest set of curves such that every partition into two sets can be realized by the intersection of the set of curves with some approximate DTW ball.

[DNPP21, BD23] observed that the VC dimension of the discrete and continuous Fréchet distance exhibits a near-linear dependency on the complexity of the sequences used as centers of the ranges, yet it depends only logarithmically on the size of the curves within the set of curves. This distinction holds significant implications in the analysis of real data sets, where queries may involve simple, short sequences, but the data set itself may consist of complex, lengthy sequences. We show that there is an approximation of DTW, for which similar bounds apply. More precisely, we show that for any given set of input sequences requiring DTW-based analysis, there is an approximation of DTW whose associated range space of bounded VC dimension. This is sufficient to enable a broad array of algorithmic techniques that leverage the VC dimension, particularly in scenarios where approximate computations are allowed.

Bibliographical Notes The results presented in Chapter 3 are based on the following work:

> Jacobus Conradi, Benedikt Kolbe, Ioannis Psarros, and Dennis Rohde. Fast Approximations and Coresets for (k, ℓ)-Median Under Dynamic Time Warping. In *40th International Symposium on Computational Geometry (SoCG 2024)*, pages 42:1–42:17, 2024. doi:10.4230/LIPICS.SOCG.2024.42.

This paper resulted from a dynamic and collaborative development process, with all authors contributing equally. The author of this thesis contributed the adaptation of the analysis of the sensitivity-bounds under DTW and the approximation of DTW via its shortest-path metric to obtain a fast initial approximation for (k, ℓ)-median. The detailed analysis was carried out jointly by all authors.

1.2 Clustering Subtrajectories

In the (k, ℓ)-median problem we cluster curves as if they were points in a space endowed with a complex distance measure. When clustering subtrajectories, we instead want to find clusters of *subcurves*, i.e., small pieces of the input curves, that represent complex patterns that reoccur along the curves [AFM$^+$18, BKK20, LYW$^+$24], such as commuting patterns in traffic data.

We study two variants of a problem posed in [ABCD23], which are based on the well-known set cover and coverage maximization problems. Both rely on the definition

of a geometric set system, which is built using metric balls under the Fréchet distance. In both variants one is provided with a polygonal curve, and is asked to compute a set of 'center' curves, each of complexity at most ℓ, that either

1. is of minimal cardinality and covers the entirety of the input curve, or
2. covers as much as possible of the input curve under cardinality constraints.

The first problem we refer to as Subtrajectory Covering, the second problem we refer to as Subtrajectory Coverage Maximization. In either setting a point p of the input curve is considered as 'covered', if there is a subcurve π_p of the input curve that contains p and has small Fréchet distance to some center curve. Note that a point p may be covered by several center curves. For the precise formulation, refer to Section 4.1. This formulation extends immediately to a set of input curves—instead of one curve—where one is now tasked to cover the entirety (respectively as much as possible) of all points on all input curves combined.

Results for Subtrajectory Covering Given a curve defined by n points, let k_Δ be the minimal cardinality of any set of curves that covers (in the sense described above) the entirety of the given curve. In Chapter 4, we first show that there is an algorithm with running time in $\widetilde{\mathcal{O}}(k_\Delta n^3)$ which computes a set of $\mathcal{O}(k_\Delta \log n)$ center curves that cover the input curve, where $\widetilde{\mathcal{O}}(\cdot)$ hides polylogarithmic factors in n. For this, we show that the Subtrajectory Covering problem can be reduced to a *discretized* set cover instance, where the set family consists of $\widetilde{\mathcal{O}}(n^2)$ sets of $\mathcal{O}(n)$ intervals in $[0,1]$ and the ground set consists of the induced arrangement of all $\widetilde{\mathcal{O}}(n^3)$ intervals.

We extend the analysis of this algorithm to the class of c-packed curves. A curve is c-packed, if its total arc length inside any ball of radius r is at most cr. Notably, curves encountered in practice often are c-packed for small c [GSW20]. We bound the running time of the algorithm provided with a c-packed curve by roughly $\widetilde{\mathcal{O}}(k_\Delta c n^2)$. This result is complemented with experimental results that highlight the versatility and robustness of the approach, but also suggest that the provided bounds of the running time are not necessarily tight.

Lastly, we improve the worst-case running time of $\widetilde{\mathcal{O}}(k_\Delta n^3)$. For this we show that the set family consisting of $\widetilde{\mathcal{O}}(n^2)$ sets of $\mathcal{O}(n)$ intervals can be partitioned into few ordered lists where the sets of intervals of consecutive elements from the lists are 'structurally similar'. We exploit this structure by reusing the computations performed to compute the intervals of preceding elements in the lists to compute the intervals of later elements more efficiently. This structure together with an intricate two-staged sampling of the ground set $[0,1]$ results in an algorithm with running time in $\widetilde{\mathcal{O}}(\sqrt{k_\Delta} n^{5/2})$ which computes a set of $\mathcal{O}(k_\Delta \log n)$ center curves that cover the input curve.

Results for Subtrajectory Coverage Maximization Given a curve P defined by n points as well as a cardinality constraint k, let Λ be the maximal total arc length of the pieces of P that can be covered by k center curves. In Section 4.9, we combine the insights of the set family with a linearization of the free space (region in the parameter space of the two curves where the corresponding distances stay below a threshold) which yields an algorithm that can compute the element from the set system with (approximately) maximal arc length of newly covered pieces in $\widetilde{\mathcal{O}}(n^2)$ time. With this at hand, we present an algorithm with running time in $\widetilde{\mathcal{O}}(kn^2)$ which computes k center curves that cover pieces of P of length at least $\frac{e-1}{16e}\Lambda$.

Bibliographic Notes The results presented in Chapter 4 are based on the following three works:

> Frederik Brüning, Jacobus Conradi, and Anne Driemel. Faster Approximate Covering of Subcurves Under the Fréchet Distance. In *30th Annual European Symposium on Algorithms (ESA 2022)*, pages 28:1–28:16, 2022. doi:10.4230/LIPICS.ESA.2022.28.
>
> Jacobus Conradi and Anne Driemel. Finding Complex Patterns in Trajectory Data via Geometric Set Cover. *arXiv preprint arXiv:2308.14865*, 2023. doi:10.48550/ARXIV.2308.14865.
>
> Jacobus Conradi and Anne Driemel. Subtrajectory Clustering and Coverage Maximization in Cubic Time, or Better. In *33rd Annual European Symposium on Algorithms (ESA 2025)*, pages 12:1–12:18 2025. doi:10.4230/LIPICS.ESA.2025.12.

These papers resulted from a dynamic developing process that all authors contributed to equally. In the first paper the author of this thesis contributed the analysis of a preliminary version of the set system for both general and c-packed curves. Other results from the paper not presented in this thesis include a combination of the set system with probabilistic techniques to yield an algorithm that computes an approximate solution to the Subtrajectory Covering problem which only consists of edges. In the second paper the author of this thesis contributed the extension of the probabilistic approach to a deterministic approach which computes centers of non-constant complexity and validated this approach with an experimental work. In the third paper, the author of this thesis contributed a linearization of the free space, a new approximation of the set system, and insights into the set system presented in [vdHvdHO25], which yields the first deterministic approximation algorithm with cubic running time and, in certain cases, even subcubic running time.

1.3 Approximate Nearest Neighbor for Curves

Nearest neighbor searching is one of the most fundamental problems in theoretical computer science and computational geometry. Formally, it asks: Given a set S of n points in a space M and a query point q in M, what is the nearest neighbor in S to q? This question has been studied since the 1960s [MP69] and finds applications in many different areas such as RNA sequencing [BS06], disease diagnosing [STS12], motion pattern detection [GvKS04], shape indexing [BL97], or handwritten digit recognition [Lee91]. Classically, M is the Euclidean plane $\mathbb{R}^2$ and the distance between two points is given by the Euclidean distance. In this case, classical results such as point location in a Voronoi diagram achieve a query time of $\mathcal{O}(\log n)$ while using $\mathcal{O}(n \log n)$ space in $\mathbb{R}^2$ [SH75], which was later improved upon to only require linear space and preprocessing time with logarithmic query time [Kir83].

While these classical results provide efficient exact solutions in low-dimensional Euclidean spaces, their performance degrades in higher dimensions, motivating the study of approximate versions [HPIM12]. This is formalized in the *c-approximate* nearest neighbor problem (c-ANN) for $c > 1$: Given a set S of n points in a space M and a query point q in M, identify a point in S whose distance to q is at most c times the distance

from q to its exact nearest neighbor. One complexity measure often used to generalize approaches for Euclidean spaces to more general metric spaces is the notion of doubling dimension [GKL03, KR02, Tal04]. The doubling dimension is the smallest number d such that any metric ball inside the metric space can be covered by 2^d many balls of half the radius. It is a well-known fact that the doubling dimension of the Euclidean space corresponds roughly to its dimension d. More precisely, the doubling dimension of $\mathbb{R}^d$ is in $\Theta(d)$. The approximate nearest neighbor problem in spaces with low doubling dimension has been studied extensively [KR02] and results are known that roughly match the bounds known for $\mathbb{R}^d$ [HPM06].

Unfortunately, the metric space of curves under the continuous Fréchet distance has been shown to have unbounded doubling dimension [DKS16]. As a curve may consist of arbitrarily short and arbitrarily long edges, the ratio between the longest and shortest edge length is unbounded. This arbitrarily large ratio is the central property enabling the construction in [DKS16] which shows that the doubling dimension is unbounded.

Results In Chapter 5, we first consider the space of curves in $\mathbb{R}^d$ consisting of at most k edges whose shortest edge has length at least λ and the longest edge has at most length Λ. We show that the doubling dimension of this subspace depends only on Λ/λ, d and k. We further show that this subspace is very close (their Gromov-Hausdorff distance is at most λ) to the space of all curves consisting of at most k edges whose longest edge has at most length Λ. We combine this bound of the doubling dimension of the close-by space together with the construction of a $(1+\varepsilon)$-ANN data structure given a bounded doubling dimension from [HPM06]. The result is a $(1+\varepsilon)$-ANN data structure for a set S of n curves in d-dimensional space, by choosing the parameters Λ and λ based on the set S. These choices depend on ε and the spread of the points defining the curves in S, i.e., the ratio between the largest and smallest point-to-point distance. Overall, the data structure has linear size in n. The query time has logarithmic dependence on n.

Our approach is in line with the more general idea of using results for spaces of bounded doubling dimension in the context of spaces that are close to a space of bounded doubling dimension, even if they do not themselves have this property [SS22].

We complement our upper bound of the doubling dimension of the subspace of curves with bounded edge lengths with a lower bound construction, showing that our analysis is not far from being tight.

Bibliographical Notes The results presented in Chapter 5 are based on the following work:

> Jacobus Conradi, Anne Driemel, and Benedikt Kolbe. $(1+\varepsilon)$-ANN Data Structure for Curves via Subspaces of Bounded Doubling Dimension. *Computing in Geometry and Topology*, 3(2):6:1–6:22, 2024. `doi:10.57717/CGT.V3I2.45`.

This paper resulted from a dynamic developing process that all authors contributed to equally. The author of this thesis contributed the main volumetric bounds allowing the bounds on the doubling dimension. The detailed analysis was mainly carried out by the author of the thesis under the supervision and consultation of Anne Driemel and in close cooperation with Benedikt Kolbe.

1.4 Shortcut Fréchet Distance

The Fréchet distance is a bottleneck distance, and may be heavily influenced by outliers, measurement errors or local distortions, limiting the practical use of the classical Fréchet distance. This has motivated a growing body of work on partial and robust variants of the Fréchet distance. These include a partial Fréchet distance as defined in [BBW09], which maximizes the portions of the two curves matched to one another within some given distance threshold, and the shortcut Fréchet distance [DHP12], which allows replacing subcurves with straight-line shortcuts in the spirit of edit distances, as well as a variant introduced in [ABRU19] which permits discrete 'jumps' (backwards and forwards) in the curve traversal.

It is conceivable that computing a partial dissimilarity based on the Fréchet distance should be more difficult than the standard Fréchet distance because of the structure of the optimization problems involved. While the (discrete or continuous) Fréchet distance can be computed in roughly quadratic (in n) time for two polygonal curves of n vertices [AG95, AHPK$^+$06, BBMM17, Bri14, BM16, BOS19], the overall picture on the computational complexity of the partial variants is very heterogeneous.

On the one hand, the problem of computing the partial Fréchet distance is not solvable by radicals over $\mathbb{Q}$ and the degree of the polynomial equations involved is unbounded in general [DGM$^+$14]. On the other hand, some variants of the partial Fréchet distance can be computed exactly in polynomial time [BBW09]. Similarly, the shortcut Fréchet distance was shown to be NP-hard [BDS14] when shortcuts are allowed anywhere along the curve. The discrete Fréchet distance with shortcuts was shown to be computable in strictly sub-quadratic time, however, which is faster than computing the discrete Fréchet distance without shortcuts [AFK$^+$15]. The variant defined in [ABRU19] turns out to be NP-hard, but allows for fixed-parameter tractable algorithms.

Results In Chapter 6, we study a parameterized version of the shortcut Fréchet distance, where the number of allowed shortcuts is bounded by a parameter k. This model offers a trade-off between the classical and fully shortcut-based versions. Our main contributions are threefold.

First, we provide an exact exponential-time algorithm for deciding whether the k-shortcut Fréchet distance between two polygonal curves in the plane is at most a given threshold δ. The running time of the algorithm is in $\mathcal{O}(kn^{2k+2}\log n)$ and has a space requirement in $\mathcal{O}(kn^{2k+2})$. It proceeds in k rounds over the classical free space diagram [AG95], each round corresponding to one additional shortcut. To identify valid shortcut intervals, we use line-stabbing wedges [GHMS94], as also employed in [BDS14].

Second, we prove a conditional lower bound. Assuming the Exponential Time Hypothesis (ETH), there is no algorithm with running time $n^{o(k)}$ for deciding the k-shortcut Fréchet distance in $\mathbb{R}^d$ for $d \geq 2$. Our reduction is based on a k-SUM variant called k-Table-SUM, adapting previous constructions from [BDS14] via a novel gadget to reduce the number of shortcuts per decision step to a constant independent of n.

Third, we develop an approximation algorithm for c-packed curves. For two c-packed curves [DHW12] of total complexity n, and given parameters $\delta > 0$ and $\varepsilon \in (0, 1]$, we provide an algorithm whose running time is in $\mathcal{O}\left(kcn\varepsilon^{-5}\log^2(n\varepsilon^{-1})\right)$ with a space requirement in $\mathcal{O}\left(kcn\varepsilon^{-4}\log^2(\varepsilon^{-1})\right)$. The algorithm returns either (i) $d_{\mathcal{S}}^k(T, B) \leq (3+\varepsilon)\delta$ or (ii) $d_{\mathcal{S}}^k(T, B) > \delta$, in either case it is correct. Since any polygonal curve of complexity n is $2n$-packed, the result implies an $\mathcal{O}(kn^2\varepsilon^{-5}\log^2(n\varepsilon^{-1}))$ time algorithm for general

curves in the plane.

Our approximation algorithm exploits the approximate monotonicity of the Fréchet distance between a shortcut and a subcurve [DHP12], enabling it to safely use the shortest feasible shortcut in each round. Instead of evaluating the Fréchet distance of shortcuts to subcurves directly via line-stabbing wedges, we rely on a data structure that computes the Fréchet distance from a segment to a subcurve, using a convex hull approximation. To ensure near-linear running time, we apply curve simplification techniques that bound the complexity of the free space diagram between c-packed curves.

Bibliographical Notes The results presented in Chapter 6 are based on the following work:

> Jacobus Conradi and Anne Driemel. On Computing the k-Shortcut Fréchet distance. *ACM Transactions on Algorithms*, 20(4):29:1–29:37, 2024. doi:10.1145/3663762.

This paper builds on the author's Master's thesis supervised by Anne Driemel. The research was collaborative, with all authors contributing equally to the overall design and insights. The main analysis was performed by the author of this thesis under the supervision of Anne Driemel.

Chapter 2

Basic Notation, Concepts and Techniques

We begin by formally defining and discussing the most important objects, their basic properties, and the notation we use in this thesis. For a natural number n we denote the set $\{1, \ldots, n\}$ by $[n]$. For a set X we denote its power set—the set containing all subsets of X—by $\mathcal{P}(X)$. Our primary focus in this thesis lies on curves, which we usually think of as lying in an ambient Euclidean space $\mathbb{R}^d$. In Euclidean space we measure the distance of points via the Euclidean norm $\|x\| = \sqrt{\langle x, x\rangle} = \sqrt{\sum_{i\in[d]} x_i^2}$, where the Euclidean distance between two points $p, q \in \mathbb{R}^d$ is then given by $\|p - q\|$. For a metric space $\mathcal{M} = (M, d)$, $p \in M$ and $r \in \mathbb{R}_{\geq 0}$ we define the closed ball $\mathrm{D}_r^{\mathcal{M}}(p)$, sometimes referred to as a disk, as the set $\{q \in M \mid d(p, q) \leq r\}$. Whenever the metric space is clear from context, we may simply write $\mathrm{D}_r(p)$.

2.1 Polygonal Curves and Similarity Measures

We call an ordered set $(p_1, \ldots, p_n) \subset \mathbb{R}^d$ of points in d-dimensional Euclidean space a ***point sequence***. We call the cardinality of a point sequence P its ***complexity***, and denote it by $|P|$. A ***polygonal curve*** is a map $P : [0, 1] \to \mathbb{R}^d$ defined by a point sequence of complexity n together with n values $0 = t_1 < t_2 < \ldots < t_n = 1$, where $P(t) = p_i + (p_{i+1} - p_i)\frac{t-t_i}{t_{i+1}-t_i}$ if $t_i \leq t \leq t_{i+1}$. We call the set $(t_1, \ldots, t_n)$ the ***vertex parameters*** of P and the points $p_1, \ldots, p_n$ the ***vertices*** of P. The complexity $|P|$ of a polygonal curve is the complexity of the underlying point sequence. A polygonal curve of complexity 2 is called an ***edge***. An edge defined by the point sequence $(p, q) \subset \mathbb{R}^d$ we denote by $\overline{p\,q}$. Conversely, a polygonal curve of complexity n can be thought of as the concatenation of the $n - 1$ edges $\overline{p_1\,p_2} \oplus \overline{p_2\,p_3} \oplus \ldots \oplus \overline{p_{n-1}\,p_n}$. These edges we call the edges of P. A point $P(t)$ is said to lie on the i^{th} edge of P if $t_i \leq t \leq t_{i+1}$. For any polygonal curve P and values $0 \leq s \leq t \leq 1$ we define $P[s, t]$ to be the ***subcurve*** of P from s to t, which itself is a polygonal curve of complexity at most $|P|$, where $P[s, t](t') = P(s + (1 - t')(t - s))$ for any $t' \in [0, 1]$. We further define the ***reversal*** $\mathrm{rev}(P)$ of P as the polygonal curve resulting from P by reversing the parametrization, i.e., $\mathrm{rev}(P)(t) = P(1 - t)$ for any $t \in [0, 1]$. Observe the relationship $\mathrm{rev}(P[s, t]) = \mathrm{rev}(P)[1 - t, 1 - s]$ for every curve P, and $0 \leq s \leq t \leq 1$.

The arc length $\|e\|$ of an edge $e = \overline{p\,q}$ is the Euclidean distance $\|q - p\|$. The arc length $\|P\|$ of a polygonal curve P is the sum of arc lengths of its edges. A curve P is

said to be c-packed, if its arc length inside any disk $\mathrm{D}_r(p)$, denoted by $\|P \cap \mathrm{D}_r(p)\|$, is at most cr.

The set of all polygonal curves in $\mathbb{R}^d$ of complexity at most m we denote by $\mathbb{X}^{d,m}$. The subset of curves whose complexity is exactly m we denote by $\mathbb{X}^{d,=m}$. The subset of curves in $\mathbb{X}^{d,m}$ whose edges have length at most Λ we denote by $\mathbb{X}^{d,m}_{\Lambda}$.

The ***continuous Fréchet distance*** of two curves P and Q, is defined as

$$\mathrm{d}_{\mathcal{F}}(P,Q) = \inf_{f,g} \max_{t\in[0,1]} \|P(f(t)) - Q(g(t))\|,$$

where f and g range over all surjective non-decreasing functions. Recall the metaphor from Chapter 1, in which the Fréchet distance was introduced via a man and a dog traversing the two curves P and Q. Here, f and g play the roles of the man and the dog, respectively, with their parameterizations chosen to minimize the maximal distance between them over time. We may refer to a pair (f, g) of such curves as a traversal or a reparameterization. The discrete Fréchet distance has been proposed as a discretization of the continuous Fréchet distance. Let $p_1, \ldots, p_n$ and $q_1, \ldots, q_m$ be the vertices of P and Q respectively. Define the space $\mathcal{T}_{n,m}$ of ***(n, m)-traversals*** as the set of sequences $((a_1, b_1), (a_2, b_2), \ldots, (a_l, b_l))$, such that

1. $a_1 = 1$ and $b_1 = 1$; and $a_l = n$ and $b_l = m$,
2. for all $i \in [l-1]$ it holds that $(a_{i+1}, b_{i+1}) - (a_i, b_i) \in \{(1,0), (0,1), (1,1)\}$.

The space of (n, m)-traversals plays the role of the discretization of the set of all reparameterizations in the continuous case. The ***discrete Fréchet distance*** is defined as

$$\mathrm{d}_{\mathrm{d}\mathcal{F}}(P,Q) = \min_{T\in\mathcal{T}_{n,m}} \max_{(i,j)\in T} \|p_i - q_j\|.$$

Both the continuous and discrete Fréchet distances are bottleneck distances, i.e., they minimize the maximal Euclidean distance between points matched during a traversal, and as such can be heavily influenced by outliers. In practice, we sometimes prefer distances that are more robust to such outliers. One such distance measure is the ***dynamic time warping*** distance (DTW), which can be thought of as the 'average discrete Fréchet distance', and is defined as

$$\mathrm{dtw}(P,Q) = \min_{T\in\mathcal{T}_{n,m}} \sum_{(i,j)\in T} \|p_i - q_j\|.$$

The dynamic time warping distance can be thought of as the minimum 1-norm of the distance vector induced by any traversal. Similarly, the discrete Fréchet distance can be thought of as the minimum ∞-norm of the distance vector induced by any traversal. We study a generalization of which both the discrete Fréchet distance and DTW are special cases, namely the ***p-dynamic time warping*** distance

$$\mathrm{dtw}_p(\sigma,\tau) = \min_{T\in\mathcal{T}_{n,m}} \left(\sum_{(i,j)\in T} \|\sigma_i - \tau_j\|^p \right)^{1/p}.$$

Various extensions of the continuous Fréchet distance which are less sensitive to outliers have been proposed. Broadly, they fall into two categories: Extensions of the dynamic time warping distance to the continuous domain and partial similarity measures. We will focus on the latter, studying the shortcut Fréchet distance. Given a curve P, we say P' is a shortcut curve of P if P' results from P by replacing subcurves of P with edges.

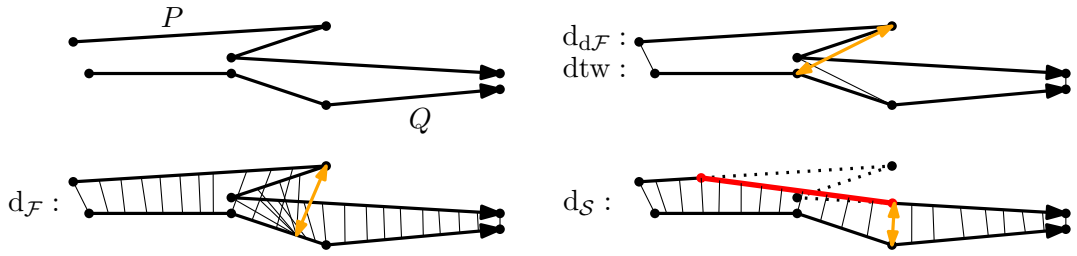


Figure 2.1: Two curves P and Q in $\mathbb{X}^{2,4}$. Further illustrated are the traversals and reparametrizations realizing the discrete Fréchet distance, dynamic time warping, continuous Fréchet distance and shortcut Fréchet distance. Additionally highlighted are the point-to-point distances realizing the bottleneck distances: the discrete Fréchet distance, continuous Fréchet distance and shortcut Fréchet distance.

More precisely, P' is defined via P and values $0 \leq s_1 \leq t_1 \leq s_2 \leq \ldots \leq s_m \leq t_m \leq 1$, with P' being the concatenation of $P[0, s_1]$, $\overline{P}[s_1, t_1]$, $P[t_1, s_2]$, …, $\overline{P}[s_m, t_m]$ and $P[t_m, 1]$, where $\overline{P}[s, t]$ is the ***shortcut*** $\overline{P(s)\,P(t)}$. Define the ***shortcut Fréchet distance*** $\mathrm{d}_{\mathcal{S}}(P, Q)$ of P and Q as the minimum Fréchet distance of any shortcut curve P' and Q. Note that the shortcut Fréchet distance is directional, that is, $\mathrm{d}_{\mathcal{S}}(P, Q) \neq \mathrm{d}_{\mathcal{S}}(Q, P)$. For an overview of the distances refer to Figure 2.1. In Chapter 6 we study a parametrized version of the shortcut Fréchet distance: the ***k-shortcut Fréchet distance*** $\mathrm{d}^k_{\mathcal{S}}(P, Q)$ is defined as the the minimum Fréchet distance of any shortcut curve P' with at most k shortcuts and Q. Similarly to the non-parametrized shortcut Fréchet distance, the k-shortcut Fréchet distance in general is not symmetric.

Note that for curves of complexity 2 (i.e., edges) the distances are trivial.

Observation 2.1.1. *Given two curves P and Q in $\mathbb{R}^d$ each defined by sequences of two points (a, b) and (c, d) respectively. Then*

- $\mathrm{d}_{\mathcal{F}}(P, Q) = \max(\|a - c\|, \|b - d\|)$,
- $\mathrm{d}_{d\mathcal{F}}(P, Q) = \max(\|a - c\|, \|b - d\|)$
- $\forall k \in \mathbb{N} : \mathrm{d}^k_{\mathcal{S}}(P, Q) = \max(\|a - c\|, \|b - d\|)$, *and*
- $\mathrm{dtw}(P, Q) = \|a - c\| + \|b - d\|$.

Lastly, observe the following relation between the distance measures.

Observation 2.1.2. *Given two curves P and Q in $\mathbb{R}^d$. Then for any k*

$$\mathrm{d}_{\mathcal{S}}(P, Q) \leq \mathrm{d}^k_{\mathcal{S}}(P, Q) \leq \mathrm{d}_{\mathcal{F}}(P, Q) \leq \mathrm{d}_{d\mathcal{F}}(P, Q) \leq \mathrm{dtw}(P, Q).$$

2.2 Computational Model

Throughout this thesis we formulate and analyze algorithms in the `realRAM` model. In this model the input to any algorithm consists of reals which we may store, and on which we may perform the basic arithmetic operations $+$, $-$, $\times$ and $/$ as well as the binary comparison operations $<$, $\leq$, $=$, $\geq$ and $>$ in constant time. We additionally assume that the we may perform the square-root operation $\sqrt{\cdot}$ computing the square root of a given (non-negative) real in constant time.

2.3 Free Space Diagram

[AG95] introduced the so-called ***free space diagram*** as an algorithmic concept to compute the continuous Fréchet distance of two given polygonal curves.

Definition 2.3.1 (Free Space Diagram)**.** Let P and Q be two polygonal curves. The free space diagram of Q and P is defined as the joint parametric space $[0,1]^2$ together with a not necessarily uniform grid, where each vertical grid line corresponds to a vertex parameter of P and each horizontal grid line corresponds to a vertex parameter of Q. The ***Δ-free space*** (refer to Figure 2.2) of Q and P is defined as

$$\mathcal{D}_\Delta(Q,P) = \left\{(x,y) \in [0,1]^2 \mid \|P(x) - Q(y)\| \leq \Delta\right\}.$$

This is the set of points in the parametric space, whose corresponding points on Q and P are at a distance of at most Δ. The edges of Q and P segment the free space diagram into cells. Denote by $\mathcal{D}_\Delta^{(i,j)}(Q,P) = \mathcal{D}_\Delta(Q,P) \cap C_{i,j}$ the Δ-free space inside the cell $C_{i,j}$ corresponding to the i^{th} edge of P and the j^{th} edge of Q. We call the intersection of $\mathcal{D}_\Delta(Q,P)$ with the boundary of cells the ***free space intervals***.

[AG95] noted that the Fréchet distance of Q and P is at most Δ, iff there is a monotone (in both x- and y-direction) path from $(0,0)$ to $(1,1)$ in the Δ-free space of Q and P. They further observed that the free space inside any cell is described by an ellipse intersected with the cell, and thus is convex and has constant complexity. Hence, the Δ-free space can be used to decide whether the Fréchet distance of two curves of complexity n and m is at most Δ in $\mathcal{O}(mn)$ time.

In [DHW12] it was shown that the number of cells with non-empty Δ-free space of suitable simplifications of two c-packed curves is in $\mathcal{O}(c \cdot \max(m,n))$, which enables linear time approximation algorithms for the decision of whether the Fréchet distance is at most Δ for two c-packed curves. This will be an important and reoccurring concept, as curves encountered in practice are often c-packed for small c [GSW20].

We now present a generalization of the Δ-free space, the *approximate* free space. The approximate free space will be an important tool in Chapter 4.

Definition 2.3.2 (Approximate Free Space)**.** Let P and Q be two polygonal curves parametrized over $[0,1]$. Let $\Delta \geq 0$ and $\alpha \geq 1$ be given. We say A is an α-approximate Δ-free space (or (α,Δ)-free space) of Q and P, if

1. $\mathcal{D}_\Delta(Q,P) \subset A \subset \mathcal{D}_{\alpha\Delta}(Q,P)$,
2. A is convex inside the interior of every cell of the free space diagram and
3. any point p on the boundary of a cell of the free space diagram is in A iff for every cell p lies in it is in the closure of the restriction of A to the interior of that cell.

The $(1,\Delta)$-free space of Q and P is unique and coincides with $\mathcal{D}_\Delta(Q,P)$. By definition any approximate free space of Q and P also defines an approximate free space of $\text{rev}(Q)$ and P by mirroring it along the x-axis and translating it by $(0,1)$.

Any monotone path from (a,b) to (c,d) in an (α,Δ)-free space of Q and P implies that $\mathrm{d}_{\mathcal{F}}(P[a,c], Q[b,d]) \leq \alpha\Delta$.

Definition 2.3.3 (Extremal Points)**.** Let A be an (α,Δ)-free space. As A is convex in any cell C, the set of points of A minimizing the x-coordinate in C are described by a vertical line segment of length at least 0. We call the start- and endpoint of this

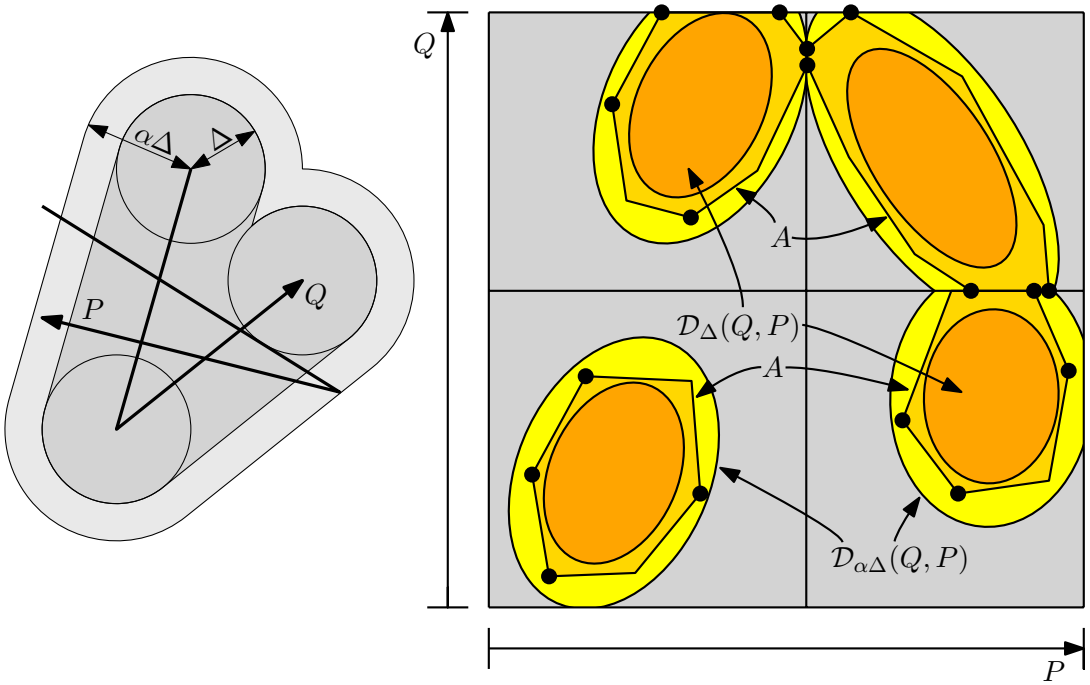


Figure 2.2: The Δ-free space and the $\alpha\Delta$-free space of two curves P and Q, as well as an α-approximate Δ-free space A of P and Q. Additionally marked are the extremal points of A.

line segment the leftmost points of A in C. Similarly A has at most two bottommost, rightmost and topmost points in C. The union over all left-, bottom-, right- and topmost points of A in every cell defines the set of extremal points of A (refer to Figure 2.2).

We impose a total order on the y-coordinates of extremal points via symbolic perturbation, where extremal points inside a cell are ordered such that the topmost points lie above the left- and rightmost points, which in turn lie above all bottommost points in that cell. If points from different cells have the same y-coordinate, we say that points from cells with lexicographically smaller index lie below points from cells with higher index. Lastly, comparing two topmost (resp. bottommost) points from the same cell we order them such that the left topmost point is below the right topmost point.

Aside from the $(1, \Delta)$-free space we will make use of an approximate Δ-free space in this work, which we describe in the following.

Piecewise Linear Approximate Δ-Free Space

We provide an approximation to the Δ-free space that is *independent* of the dimension of the ambient space that the polygonal curves live in. This is constructed by projecting every pair of edges e_1 and e_2 (corresponding to some cell of the Δ-free space) into $\mathbb{R}^3$ resulting in $\hat{e}_1$ and $\hat{e}_2$ such that for any $(x, y) \in [0, 1]^2$ the distances $\|e_1(x) - e_2(y)\|$ and $\|\hat{e}_1(x) - \hat{e}_2(y)\|$ coincide. Importantly, in $\mathbb{R}^3$ we can $(1 + \varepsilon)$-approximate the ball with radius Δ by a polytope represented as the intersection of at most $\mathcal{O}(\varepsilon^{-2})$ halfspaces in $\mathbb{R}^3$. This polytope defines a polygon in $[0, 1]^2$ which $(1 + \varepsilon)$-approximates the Δ-free space of $\hat{e}_1$ and $\hat{e}_2$ which coincides with the Δ-free space of e_1 and e_2.

Theorem 2.3.4. *Let P and Q be two polygonal curves in $\mathbb{R}^d$. Let $\varepsilon \in (0,1]$ be given. In $\mathcal{O}(|P||Q|\varepsilon^{-2}\log(\varepsilon^{-1}))$ time one can compute a $(1+\varepsilon,\Delta)$-free space of P and Q whose restriction to the interior of any cell of the free space diagram is described by a convex polygon of complexity $\mathcal{O}(\varepsilon^{-2})$.*

We prove this theorem via the following folklore lemmas. For the sake of completeness, we provide proofs for these lemmas.

Lemma 2.3.5. *Let $p,q,r,s \in \mathbb{R}^d$ be given. One can compute in $\mathcal{O}(d)$ time four points $\hat{p},\hat{q},\hat{r},\hat{s} \in \mathbb{R}^3$ such that for any $(x,y) \in [0,1]^2$ it holds that*

$$\|p + y(q-p) - (r + y(s-r))\| = \|\hat{p} + x(\hat{q}-\hat{p}) - (\hat{r} + y(\hat{s}-\hat{r}))\|.$$

Proof. Assume $d > 3$. Without loss of generality assume $p = 0$ by translating all points by $-p$. Then q, r, and s lie on a three-dimensional subspace of $\mathbb{R}^d$. Let e_1, e_2 and e_3 be pairwise orthogonal vectors of length 1 spanning the same subspace, which may be computed from q, r, and s by, for example, the Gram-Schmidt process. Then $\hat{q} = (\langle q,e_1\rangle, \langle q,e_2\rangle, \langle q,e_3\rangle)$ and similarly $\hat{r} = (\langle r,e_1\rangle, \langle r,e_2\rangle, \langle r,e_3\rangle)$, and $\hat{s} = (\langle s,e_1\rangle, \langle s,e_2\rangle, \langle s,e_3\rangle)$. As q, r and s are in the subspace spanned by e_1, e_2 and e_3 we also have that $q = \langle q,e_1\rangle e_1 + \langle q,e_2\rangle e_2 + \langle q,e_3\rangle e_3$ and similarly for r and s. Now, as e_1, e_2 and e_3 are pairwise orthogonal, i.e., $\langle e_i,e_j\rangle$ is 0 if $i \neq j$, and $\langle e_i,e_i\rangle = 1$ for $i \leq 3$ we have that

$$\begin{aligned}\langle q,q\rangle &= \big\langle \langle q,e_1\rangle e_1 + \langle q,e_2\rangle e_2 + \langle q,e_3\rangle e_3, \langle q,e_1\rangle e_1 + \langle q,e_2\rangle e_2 + \langle q,e_3\rangle e_3 \big\rangle \\ &= \langle q,e_1\rangle\langle q,e_1\rangle + \langle q,e_2\rangle\langle q,e_2\rangle + \langle q,e_2\rangle\langle q,e_2\rangle = \langle \hat{q},\hat{q}\rangle,\end{aligned}$$

and similarly $\langle q,r\rangle = \langle \hat{q},\hat{r}\rangle$, $\langle q,s\rangle = \langle \hat{q},\hat{s}\rangle$, $\langle r,r\rangle = \langle \hat{r},\hat{r}\rangle$, $\langle r,s\rangle = \langle \hat{r},\hat{s}\rangle$, and $\langle s,s\rangle = \langle \hat{s},\hat{s}\rangle$. Hence for any $(x,y) \in [0,1]^2$ it holds that

$$\langle xq - (r + y(s-r)), xq - (r + y(s-r))\rangle = \langle x\hat{q} - (\hat{r} + y(\hat{s}-\hat{r})), x\hat{q} - (\hat{r} + y(\hat{s}-\hat{r}))\rangle\,,$$

and thus

$$\|p + x(q-p) - (r + y(s-r))\| = \|\hat{p} + x(\hat{q}-\hat{p}) - (\hat{r} + y(\hat{s}-\hat{r}))\|. \qquad \square$$

Lemma 2.3.6. *Let $\varepsilon \in (0,1/5]$. One can compute a convex polytope D of complexity $\mathcal{O}(\varepsilon^{-2})$ in $\mathbb{R}^3$ such that $\mathrm{D}_1(0) \subset D \subset \mathrm{D}_{1+4\varepsilon}(0)$ in time $\mathcal{O}(\varepsilon^{-2}\log\varepsilon^{-1})$.*

Proof. Consider a grid with side length ε^{-1} in $\mathbb{R}^3$. Let G be all points of the grid that lie in $\mathrm{D}_1(0)$. One can compute in $\mathcal{O}(\varepsilon^{-2}\log\varepsilon^{-1})$ a set of $\mathcal{O}(\varepsilon^{-2})$ points S such that the convex hull of S and G coincide. For this compute for every $-\lceil\varepsilon^{-1}\rceil \leq i,j \leq \lceil\varepsilon^{-1}\rceil$ the minimum k^- and the maximum k^+ of the set $G_{i,j} = \{k \in \mathbb{Z} \mid |k| \leq \lceil\varepsilon^{-1}\rceil \;\&\; \sqrt{(i^2+j^2+k^2)}\varepsilon \leq 1\}$ via binary search. These two values define two points $(i\varepsilon, j\varepsilon, k^-\varepsilon)$ and $(i\varepsilon, j\varepsilon, k^+\varepsilon)$ whose convex hull contains all other points $(i\varepsilon, j\varepsilon, k\varepsilon)$ for $k \in G_{i,j}$ and hence all $(i\varepsilon, j\varepsilon, k\varepsilon) \in G$ for $k \in \mathbb{Z}$. Thus, the union of all these $\mathcal{O}(n^2)$ points defines S.

Let now C be the convex hull of S represented as the intersection of $\mathcal{O}(\varepsilon^{-2})$ halfspaces, which can be computed in $\mathcal{O}(\varepsilon^{-2}\log\varepsilon^{-1})$ time. Observe that $G \subset C \subset \mathrm{D}_1(0)$.

Now, for any point $x \in \mathrm{D}_1(0)$ it holds that $\min_{y\in C}\|x-y\| < 2\varepsilon^{-1}$, as otherwise there is a disk of radius ε^{-1} contained in $\mathrm{D}_1(0) \setminus C$. As the side length of the grid is ε^{-1} there would then be a grid point p in this disk of radius ε^{-1} and thus also in $\mathrm{D}_1(0) \setminus C$ contradicting the fact that $G \subset C$. Hence $\mathrm{D}_{1-2\varepsilon}(0) \subset C \subset \mathrm{D}_1(0)$.

Scaling C by a factor of $(1+4\varepsilon)$ proves the claim, as $1 \leq 1+2\varepsilon-8\varepsilon^2 = (1+4\varepsilon)(1-2\varepsilon)$ and $(1+4\varepsilon)(1-2\varepsilon) \leq (1+4\varepsilon)$. $\square$

Lemma 2.3.7. *Let e_1 and e_2 be two edges in $\mathbb{R}^3$ and let D be a polytope in $\mathbb{R}^3$ described by the intersection of C halfspaces. Then the set*

$$\{(x, y) \in [0, 1]^2 \mid e_1(x) - e_2(y) \in D\}$$

is a polygon of complexity $\mathcal{O}(C)$ and can be computed in $\mathcal{O}(C \log C)$ time.

Proof. Let $H_1, \ldots, H_C$ be the halfspace describing D represented via their normal vector n_i and value d_i (that is $H_i = \{x \in \mathbb{R}^3 \mid \langle n_i, x\rangle \leq d_i\}$) in $\mathbb{R}^3$ describing the polytope D. Then the set $\{(x, y) \in [0, 1]^2 \mid e_1(x) - e_2(y) \in D\}$ is described as the intersection of the C halfspaces in $\mathbb{R}^2$ (intersected with $[0, 1]^2$) given by $\langle n_i, e_1(x) - e_2(y)\rangle \leq d_i$. □

Proof of Theorem 2.3.4. This is a consequence of Lemma 2.3.5, Lemma 2.3.6 as well as Lemma 2.3.7. Let $\hat{\varepsilon} = \varepsilon/5 \leq 1/5$. First, compute in time $\mathcal{O}(\varepsilon^{-2} \log(\varepsilon^{-1}))$ time a polytope $D \subset \mathbb{R}^3$ such that $\mathrm{D}_\Delta(0) \subset D \subset \mathrm{D}_{(1+4\hat{\varepsilon})\Delta}(0)$. With the polytope D at hand, in $\mathcal{O}(|P||Q|\varepsilon^{-2} \log \varepsilon^{-1})$ time compute for every cell $C_{i,j}$ corresponding to the edges $e_i = \overline{a\,b}$ of Q and $e'_j = \overline{c\,d}$ of P first the four points $\hat{a}$, $\hat{b}$, $\hat{c}$, and $\hat{d}$ according to Lemma 2.3.5, and then the polygon $D_{i,j} = \{(x, y) \in [0, 1]^2 \mid \hat{e}_i(x) - \hat{e}'_j(y) \in D\}$ where $\hat{e}_i$ and $\hat{e}'_j$ are the two edges in $\mathbb{R}^3$ defined by $\hat{a}$ and $\hat{b}$, and $\hat{c}$ and $\hat{d}$ respectively. Now observe that $\mathcal{D}_\Delta(e_i, e'_j) \subset D_{i,j} \subset \mathcal{D}_{1+4\varepsilon\Delta}(e_i, e'_j)$, and $D_{i,j}$ define a $(1 + \varepsilon)$-approximate Δ-free space in the interior of every cell. These polygons define a $(1 + \varepsilon, \Delta)$-free space, where on the shared boundary of a set of cells the $(1 + \varepsilon)$-approximate Δ-free space is exactly the intersection of all polygons associated to cells in that set concluding the proof. □

Observe that the extremal points and polygon vertices of the approximate free space computed via Theorem 2.3.4 may also be endowed with a total order, where topmost points of any cell lie above all left- and rightmost points, which in turn lie above all bottommost points.

2.4 Dimensionality Measures

Both the running time and the approximation ratio of many geometric algorithms in Euclidean space $\mathbb{R}^d$ depend on the dimension. In this thesis, we focus on the space of curves under various distance measures. While we typically think of curves as lying in an ambient Euclidean space, the notion of 'dimension' for the space $\mathbb{X}^{d,m}$ is less immediate. For more general spaces different measures of dimensionality have been proposed which generalize the dimension of the Euclidean space. Let $\mathcal{M} = (M, \mathrm{d})$ be a metric space. The ***doubling constant*** of $\mathcal{M}$ is the smallest number D such that any metric ball of radius r can be covered by D balls of radius $r/2$. The ***doubling dimension*** of $\mathcal{M}$ is the logarithm $\log_2(D)$ of the doubling constant. It is well known that the doubling dimension of the d-dimensional Euclidean space $\mathbb{R}^d$ is in $\Theta(d)$. For the space of all curves $\mathbb{X}^{d,m}$ endowed with the discrete Fréchet distance observe that a curve P has discrete Fréchet distance at most r to Q if every vertex of P is at distance at most r to some vertex of Q. Choosing for every vertex q of Q a set of $2^{\Theta(d)}$ disks with radius $r/2$ covering $\mathrm{D}_r(v)$ and enumerating all sequences of length at most m of centers of such disks guarantees that the Fréchet distance of P and at least one such sequence is at most $r/2$. This implies that the doubling dimension of $(\mathbb{X}^{d,m}, \mathrm{d}_{\mathrm{d}\mathcal{F}})$ is in $\mathcal{O}(m(d+\log(m))$. For the space of all curves $\mathbb{X}^{d,m}$ under the *continuous* Fréchet distance, [DKS16] showed that

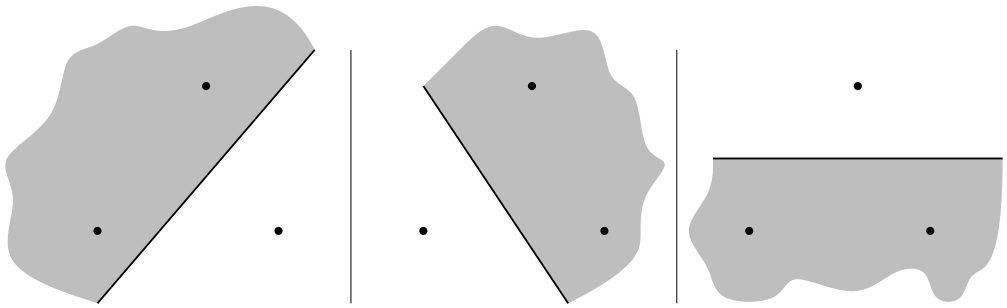


Figure 2.3: Illustration of the VC-dimension for the range space whose ground set is $\mathbb{R}^2$ and whose set system is the set of all halfspaces in $\mathbb{R}^2$. Any set of three points, that are not colinear, can be shattered, while no set of four points can be shattered by Radon's Lemma.

the doubling dimension is unbounded, even if $m = 3$ (refer to Theorem 5.4.3 for a proof of a stronger statement).

A different dimensionality measure we will work with is the so-called VC-dimension. It measures the richness of a range space. A ***range space*** is a pair (X, R), where X is a set and $R \subset \mathcal{P}(X)$ is a set of subsets of X. The set X is called the ***ground set*** of (X, R), and R is called the ***set system*** of (X, R). For a given range space (X, R) a set $A \subset X$ is said to be ***shattered***, if for any subset $A' \subset A$ there is an $r \in R$ such that $A' = A \cap r$. The ***VC-dimension*** is the cardinality of the biggest subset of X that is shattered. Simple examples include $(\mathbb{R}^2, \{\mathrm{D}_r(x) | x \in \mathbb{R}^2, r > 0\})$, whose VC-dimension is 3, $(\mathbb{R}^2, \{\text{all halfspaces in } \mathbb{R}^2\})$, whose VC-dimension is 3 (refer to Figure 2.3), $(\mathbb{R}^2, \{\text{all triangles in } \mathbb{R}^2\})$, whose VC-dimension is 7, and $([n], \mathcal{P}([n]))$, whose VC-dimension is n.

Chapter 3

(k, ℓ)-Median under Dynamic Time Warping

In this chapter, we study the p-dynamic time warping distance and whether the (k, ℓ)-median problem under p-DTW admits coresets and, based on such a coreset, an efficient algorithm.

Problem 1 (**(k, ℓ)-Median under p-DTW**). Let $T = \{\tau_1, \ldots, \tau_n\} \subset \mathbb{X}^{d,m}$, and let $k, \ell, p \in \mathbb{N}$ be given. Compute k center curves $C = \{c_1, \ldots, c_k\} \subset \mathbb{X}^{d,\ell}$ that minimize

$$\mathrm{cost}_p(T, C) = \sum_{\tau \in T} \min_{c \in C} \mathrm{dtw}_p(\tau, c).$$

Our results improve upon the algorithm in [BDvG+22] in that we provide the first polynomial-time algorithm with constant success probability.

The main content of this chapter previously appeared as the paper *Fast Approximations and Coresets for (k, ℓ)-Median under Dynamic Time Warping* [CKPR24] by Jacobus Conradi, Benedikt Kolbe, Ioannis Psarros, and Dennis Rohde which was published in the *Proceedings of the 40^{th} International Symposium on Computational Geometry (SoCG 2024)*. A full version of the paper is available on arXiv [CKPR23]. An initial version of the work has also been presented at the 40^{th} *European Workshop on Computational Geometry (EuroCG 2024)* based on an extended abstract without formal publication.

3.1 Introduction

In the last decade, the problems of (k, ℓ)-median and (k, ℓ)-center clustering for time series in $\mathbb{R}^d$ under the Fréchet distance have gained significant attention. The problem is NP-hard [BDG+19, BDS20, DKS16], even if $k = 1$ and $d = 1$, and the (k, ℓ)-center problem is even NP-hard to approximate within a factor of $(3-\varepsilon)$ for $d \geq 2$, and $(1.5-\varepsilon)$ for $d = 1$ [BDG+19, BCD+25].

For the (k, ℓ)-median problem, all presently known $(1+\varepsilon)$-approximation algorithms are based on an approximation scheme [BDR23, CH23b, DKS16] which has been generalized several times [ABS10, KSS04]. The most recent version of this scheme [BDR23, Theorem 7.2] can be used to approximate any k-median-type problem in an arbitrary space X with a distance function. All it needs is a plugin algorithm that, when given a set T of elements from some (problem-specific) subset $Y \subseteq X$, computes a set of candidates

C such that for any set $T' \subseteq T$ with $|T'| \geq |T|/k$ the set C contains an approximate median of T' with high probability. The resulting approximation quality depends on the approximation factor of the plugin algorithm. The running time depends on the size $|C|$ of the candidate set and k with a factor of $\mathcal{O}(|C|^k)$.

For the Fréchet distance, plugin algorithms exist that yield $(1+\varepsilon)$-approximations [BDR23, CH23b]. For DTW, however, the best plugin algorithm [BDvG+22] has running time exponential in k—roughly with a dependency of $\widetilde{\mathcal{O}}((32k^2\varepsilon^{-1})^{k+2}n)$—and approximation guarantee of $(8+\varepsilon)(m\ell)^{1/p}$ with constant success probability. In principle, some of the ideas from plugins for the Fréchet distance could be adapted, but the more involved plugins, i.e., the ones yielding $(1+\varepsilon)$-approximations, crucially make use of the metric properties of the distance function.

In practice, an adaption of Gonzalez algorithm for (k, ℓ)-center clustering under the Fréchet distance performs well [BDvdLN19]. Similar ideas have also been used for clustering under (a continuous variant of) DTW [BBK+20], but there are no approximation guarantees, and the usual analysis is based on repeated application of the triangle inequality.

An influential approach to solving k-median-like problems is to construct a point set that acts as a proxy on which to run computationally more expensive algorithms that yield solutions with approximation guarantees. The condensed input set is known as a coreset (refer to Figure 3.1).

Definition 3.1.1 (ε-Coreset)**.** Let $T \subset \mathbb{X}^{d,m}$ be a finite set and $\varepsilon \in (0,1)$. Then a weighted multiset $S \subset \mathbb{X}^{d,m}$ with weight function $w : S \to \mathbb{R}_{>0}$ is a weighted ε-coreset for the (k, ℓ)-median problem of T under dtw_p if for all $C \subset \mathbb{X}^{d,\ell}$ with $|C| = k$

$$(1-\varepsilon)\mathrm{cost}_p(T, C) \leq \sum_{s \in S} w(s) \min_{c \in C} \mathrm{dtw}_p(s, c) \leq (1+\varepsilon)\mathrm{cost}_p(T, C).$$

For the Fréchet distance, ε-coresets can be constructed [BCJ+22, BR22] that help facilitate the practicability of available algorithms. Using such ε-coresets w.r.t. the Fréchet distance, a $(5+\varepsilon)$-approximation algorithm for the 1-median problem was recently analyzed [BR22], yielding a total running time of roughly $nm^{\mathcal{O}(1)} + (m/\varepsilon)^{\mathcal{O}(\ell)}$, in contrast to a running time of $n(m/\varepsilon)^{\mathcal{O}(\ell)}$ without the use of coresets [BDR23].

For DTW, no coreset construction is known to this point. This is at least partially due to prominent coreset frameworks assuming a normed or at least a metric space [FL11, LS10]. Also, recently a coreset construction relying solely on uniform sampling was developed that greatly simplifies existing coreset constructions [BCJ+22], including the aforementioned coresets under the Fréchet distance. Unfortunately, the construction again relies on different incarnations of the triangle inequality, limiting its use for DTW.

3.1.1 Results

To construct ε-coresets, we use approximations of the range space defined by balls under p-DTW and bound their VC dimension. Assuming that the input is a set of n curves of complexity at most m, we present an approximation algorithm (Theorem 3.5.2) for (k, ℓ)-median with running time in roughly $\widetilde{\mathcal{O}}(n)$ (hiding other factors) that improves upon existing work in terms of running time, with comparable approximation guarantees. Our approach relies on curve simplifications and a new insight into the notion of relaxed triangle inequalities for p-DTW (Lemma 3.4.6). We use this relaxed triangle inequality to give an algorithm which computes a bicriteria approximation for the (k, ℓ)-median

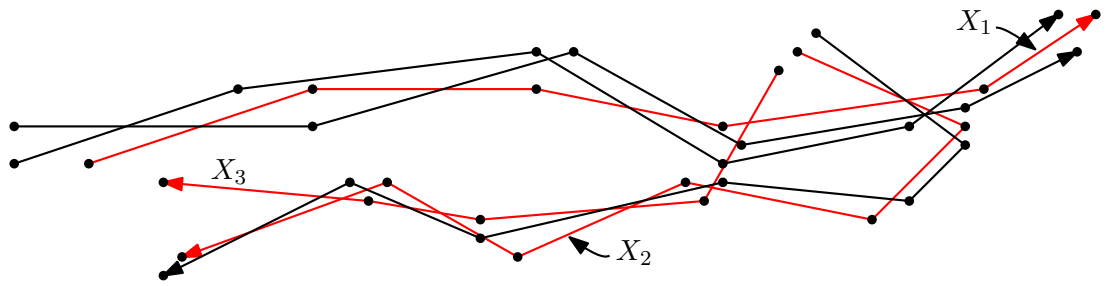


Figure 3.1: Illustration of a coreset (red), i.e., a weighted sparse representation of the original set of curves (in red and black). The weights in this case are $w(X_1) = 3$, $w(X_2) = 2$ and $w(X_3) = 1$.

problem via the finite metric space consisting of all input curves and their simplifications where the distances are defined by the shortest-path metric w.r.t. p-DTW.

Definition 3.1.2 ((α, β)-approximation)**.** Let a set $T = \{\tau_1, \ldots, \tau_n\} \subset \mathbb{X}^{d,m}$ of n curves be given. A set $\hat{C} \subset \mathbb{X}^{d,\ell}$ is called an (α, β)-approximation of (k, ℓ)-median, if $|\hat{C}| \leq \beta k$ and

$$\mathrm{cost}_p\left(T, \hat{C}\right) \leq \alpha \mathrm{cost}_p(T, C)$$

for any $C \subset \mathbb{X}^{d,\ell}$ of size k.

We use the approximation computed by the bicriteria approximation algorithm to bound the sensitivity of each input curve. The bound of the sensitivities together with the aforementioned bound on the VC dimension of the approximate range space of p-DTW balls enables application of a modified version of sensitivity sampling as introduced in [FL11]. Overall, this results in an ε-coreset for (k, ℓ)-median clustering of size quadratic in $1/\varepsilon$ and k, logarithmic in n, and depending on $(m\ell)^{1/p}$ and ℓ (Corollary 3.5.4).

3.2 VC Dimension of DTW

We now derive bounds on the VC dimension of a range space that approximates the range space of p-DTW balls in $\mathbb{X}^{d,m}$:

$$\left(\mathbb{X}^{d,m}, \left\{\{x \in \mathbb{X}^{d,m} \mid \mathrm{dtw}_p(x, \tau) \leq r\} \subset \mathbb{X}^{d,m} \middle| \tau \in \mathbb{X}^{d,\ell}, r > 0\right\}\right).$$

Our reasoning for bounding the VC dimension exclusively relies on establishing the prerequisites of Theorem 3.2.2 below.

Definition 3.2.1 ([AB99])**.** Let H be a class of $\{0, 1\}$-valued functions defined on a set X, and F a class of real-valued functions defined on $\mathbb{R}^d \times X$. We say that H is a k-combination of $\mathrm{sign}(F)$ if there is a function $g : \{-1, 1\}^k \to \{0, 1\}$ and functions $f_1, \ldots, f_k \in F$ so that for all $h \in H$ there is a parameter vector $\alpha \in \mathbb{R}^d$ such that for all x in X,

$$h(x) = g(\mathrm{sign}(f_1(\alpha, x)), \ldots, \mathrm{sign}(f_k(\alpha, x))).$$

The definition of the sign function we use is that $\mathrm{sign}(x) = 1$ for $\mathbb{R} \ni x \geq 0$ and $\mathrm{sign}(x) = -1$ for $x < 0$. Observe that the class H of functions corresponds to a set system consisting of subsets of X.

Theorem 3.2.2 (Theorem 8.3 [AB99]). *Let F be a class of maps from $\mathbb{R}^s \times X$ to $\mathbb{R}$, so that for all $x \in X$ and $f \in F$, the function $\alpha \mapsto f(\alpha, x)$ is a polynomial on $\mathbb{R}^s$ of degree δ. Let H be a κ-combination of* $\operatorname{sign}(F)$. *Then the VC dimension of H is less than $2s \log_2(12\delta\kappa)$.*

Theorem 3.2.2 implies a bound on the VC dimension of range spaces defined by p-DTW for even values of p which was independently observed for $p = 2$ in [BD23]: The decision of whether p-DTW exceeds a given threshold can be formulated as a $|\mathcal{T}_{m,\ell}|$-combination of signs of polynomial functions; each one realizing the cost of a traversal. This yields an upper bound of $\mathcal{O}(d\ell^2 \log(mp))$. The situation becomes more intriguing in the general case, since for any odd p, and in particular $p = 1$, the cost of each traversal is no longer a root of a polynomial. To overcome this, we investigate range spaces defined by approximate p-DTW balls and show that we get bounds that do not depend on p.

The following lemma shows that one can determine (approximately) the p-DTW distance between two sequences, based solely on the signs of certain polynomials that are designed to provide a sketchy view of all pointwise distances.

Lemma 3.2.3. *Let $\tau \in \mathbb{X}^{d,=\ell}$, $\sigma \in \mathbb{X}^{d,=m}$, $r > 0$ and $\varepsilon \in (0, 1]$. For each $i \in [\ell]$, $j \in [m]$ and $z \in [\lfloor \varepsilon^{-1} + 1 \rfloor]$, define*

$$f_{i,j,z}(\tau, r, \sigma) = \|\tau_i - \sigma_j\|^2 - (z \cdot \varepsilon r)^2.$$

There is an algorithm that, given as input the values of $\operatorname{sign}(f_{i,j,z}(\tau, r, \sigma))$, *for all $i \in [\ell]$, $j \in [m]$ and $z \in [\lfloor \varepsilon^{-1} + 1 \rfloor]$, outputs a value in $\{0, 1\}$ such that:*

- *if* $\mathrm{dtw}_p(\tau, \sigma) \leq r$ *then it outputs* 1,
- *if* $\mathrm{dtw}_p(\tau, \sigma) > (1 + (m + \ell)^{1/p}\varepsilon)r$ *then it outputs* 0 *and*
- *if* $\mathrm{dtw}_p(\tau, \sigma) \in (r, (1 + (m + \ell)^{1/p}\varepsilon)r]$ *the output is either* 0 *or* 1.

Proof. The algorithm first iterates over all i and j. For each i and j we assign a variable $\phi_{i,j}$ as follows: if $Z_{i,j} := \{z \in [\lfloor \varepsilon^{-1} + 1 \rfloor] \mid \operatorname{sign}(f_{i,j,z}(\tau, r, \sigma)) = -1\} \neq \emptyset$, then $\phi_{i,j} := \min(Z_{i,j})\varepsilon r$, otherwise $\phi_{i,j} := \infty$. After having computed all $\phi_{i,j}$, we return a value as follows: if

$$\Phi(\tau, \sigma) := \min_{T \in \mathcal{T}_{\ell,m}} \left(\sum_{(i,j) \in T} (\phi_{i,j})^p \right)^{1/p} \leq (1 + (m + \ell)^{1/p}\varepsilon)r,$$

then the output is 1. Otherwise, the output is 0.

We now prove the correctness of the algorithm. For this let us first observe that for all $i \in [\ell]$ and $j \in [m]$ it holds that $\|\tau_i - \sigma_j\| < \phi_{i,j}$. Further if $\|\tau_i - \sigma_j\| \leq r$ then $\phi_{i,j} - \varepsilon r \leq \|\tau_i - \sigma_j\|$. This follows from the fact that $Z_{i,j}$ coincides with the set $\{z \in [\lfloor \varepsilon^{-1} + 1 \rfloor] \mid z\varepsilon r \geq ||\tau_i - \sigma_j||\}$.

Now, the fact that for all $i \in [\ell]$ and $j \in [m]$ it holds that $\|\tau_i - \sigma_j\| < \phi_{i,j}$, wee see that $\Phi(\tau, \sigma) \geq \mathrm{dtw}_p(\tau, \sigma)$, as for any (ℓ, m)-traversal T we see that

$$\left(\sum_{(i,j) \in T} (\phi_{i,j})^p \right)^{1/p} \geq \left(\sum_{(i,j) \in T} \|\tau_i - \sigma_j\|^p \right)^{1/p} \geq \mathrm{dtw}_p(\tau, \sigma).$$

In particular, if $\mathrm{dtw}_p(\tau, \sigma) > (1 + (m + \ell)^{1/p}\varepsilon)r$ then $\Phi(\sigma, \tau) > (1 + (m + \ell)^{1/p}\varepsilon)r$, and as such the algorithm outputs 0. It remains to show that if $\mathrm{dtw}_p(\tau, \sigma) \leq r$, then $\Phi(\sigma, \tau) \leq$

$(1+(m+\ell)^{1/p}\varepsilon)r$. To show this, let $\mathrm{dtw}_p(\tau,\sigma) \leq r$ and let T^* be an (ℓ,m)-traversal realizing $\mathrm{dtw}_p(\tau,\sigma)$. In particular, for all $(i,j) \in T^*$, $\|\tau_i - \sigma_j\| \leq r$, so $\phi_{i,j} \leq \|\tau_i - \sigma_j\| + \varepsilon r$ for all $(i,j) \in T^*$. We conclude that

$$\begin{aligned}\Phi(\sigma,\tau) &\leq \left(\sum_{(i,j)\in T^*} (\phi_{i,j})^p\right)^{1/p} \leq \left(\sum_{(i,j)\in T^*} |\|\tau_i-\sigma_j\|+\varepsilon r|^p\right)^{1/p} \\ &\leq \left(\sum_{(i,j)\in T^*} \|\tau_i-\sigma_j\|^p\right)^{1/p} + \left(\sum_{(i,j)\in T^*} (\varepsilon r)^p\right)^{1/p} \\ &\leq r + (|T^*|)^{1/p}\cdot \varepsilon r \leq r + (m+\ell)^{1/p}\,\varepsilon r,\end{aligned}$$

where the inequalities hold by the Minkowski inequality and $1 \leq |T^*| \leq m+\ell$. □

The algorithm of Lemma 3.2.3 essentially defines a function that implements approximate p-DTW ball memberships, and satisfies the requirements set by Theorem 3.2.2. However, it is only defined on curves in $\mathbb{X}^{d,=\ell}$ and $\mathbb{X}^{d,=m}$. We extend the approach to all curves in $\mathbb{X}^{d,m}$.

Lemma 3.2.4. *Let $\varepsilon \in (0,1]$, and let $m,\ell \in \mathbb{N}$ be given. There are injective functions $\pi_\ell : \mathbb{X}^{d,\ell} \to \mathbb{R}^{(d+1)\ell}$ and $\pi_m : \mathbb{X}^{d,m} \to \mathbb{R}^{(d+1)m}$ and a class of functions F_ε mapping from $\left(\mathbb{R}^{(d+1)\ell} \times \mathbb{R}\right) \times \mathbb{R}^{(d+1)m}$ to $\mathbb{R}$, such that for any $f \in F_\varepsilon$ the function $\alpha \mapsto f(\alpha, x)$ is a polynomial function of degree 2. Furthermore, there is a function $g\colon \{-1,1\}^k \to \{0,1\}$ and functions $f_1,\ldots,f_k \in F_\varepsilon$, where $k = m\ell\lfloor\varepsilon^{-1}+1\rfloor + m + \ell$, such that for any $\tau \in \mathbb{X}^{d,\ell}$, $r > 0$ and $\sigma \in \mathbb{X}^{d,m}$ it holds that*

- *if* $\mathrm{dtw}_p(\sigma,\tau) \leq r$ *then*

$$g(\mathrm{sign}(f_1(\pi_\ell(\tau), r, \pi_m(\sigma))), \ldots, \mathrm{sign}(f_k(\pi_\ell(\tau), r, \pi_m(\sigma)))) = 1,$$

- *if* $\mathrm{dtw}_p(\sigma,\tau) > (1+(m+\ell)^{1/p}\varepsilon)r$ *then*

$$g(\mathrm{sign}(f_1(\pi_\ell(\tau), r, \pi_m(\sigma))), \ldots, \mathrm{sign}(f_k(\pi_\ell(\tau), r, \pi_m(\sigma)))) = 0.$$

Proof. For $\sigma \in \mathbb{X}^{d,=m'} \subset \mathbb{X}^{d,m}$ define $\widetilde{\sigma} = \pi_m(\sigma)$. The curve $\widetilde{\sigma}$ consists of m points in $\mathbb{R}^{d+1}$, where the first m' points consist of the points in σ together with a 1 in the $(d+1)^{\text{th}}$ coordinate. The $(m'+1)^{\text{th}}$ to m^{th} point is defined to be the point $(-1,\ldots,-1) \in \mathbb{R}^{d+1}$. The points of $\widetilde{\tau} = \pi_\ell(\tau)$ are defined similarly padding τ to a length of ℓ similar to σ. Let τ_i and σ_j denote the first d coordinates of the i^{th} and j^{th} point in $\widetilde{\tau}$ and $\widetilde{\sigma}$. That is, for $j \leq m'$ the point σ_j is exactly the j^{th} point of σ, and for $j > m'$ the point σ_j is the point $(-1,\ldots,-1) \in \mathbb{R}^d$. Let τ_i^{d+1} and σ_j^{d+1} denote the $(d+1)^{\text{th}}$ coordinate of the i^{th} and j^{th} $(d+1)$-dimensional point in $\widetilde{\tau}$ and $\widetilde{\sigma}$ respectively.

The set F_ε consists of all functions $f_{i,j,z}(\widetilde{\tau}, r, \widetilde{\sigma}) = \|\tau_i - \sigma_j\|^2 - (z\varepsilon r)^2$, where $i \in [\ell]$, $j \in [m]$ and $z \in \left[\lfloor\varepsilon^{-1}+1\rfloor\right]$. It further contains the functions $g_i(\widetilde{\tau}, r, \widetilde{\sigma}) = \tau_i^{d+1}$ and $h_j(\widetilde{\tau}, r, \widetilde{\sigma}) = \sigma_j^{d+1}$. The function g has $k = \ell m \cdot \lfloor\varepsilon^{-1}+1\rfloor + m + l$ arguments, corresponding to the signs of the functions $f_{i,j,z}$, g_i, and h_j always ordered in the same way. To compute g we first use the sign of g_i and h_j to infer the values of m' and ℓ', that is, the complexities of σ and τ, as $\mathrm{sign}(g_i) = 1$ if and only if $i \leq \ell'$, and similarly $\mathrm{sign}(h_j) = 1$ if and only if $j \leq m'$. It then invokes and outputs the result from the algorithm of Lemma 3.2.3 with input $\mathrm{sign}(f_{1,1,1}(\widetilde{\tau}, r, \widetilde{\sigma})), \ldots, \mathrm{sign}(f_{\ell',m',\lfloor\varepsilon^{-1}+1\rfloor}(\widetilde{\tau}, r, \widetilde{\sigma}))$, concluding the proof. □

Using the previous lemmas, we define a distance function $\widetilde{\mathrm{dtw}}_p$ between elements of $\mathbb{X}^{d,m}$ and $\mathbb{X}^{d,\ell}$, which we will use throughout the chapter as an approximate function of dtw_p. To get an estimate of the VC dimension of the range space induced by balls under $\widetilde{\mathrm{dtw}}_p$ and decide membership of points to these balls, the approximate distance will only take discrete values.

Definition 3.2.5. Let $\varepsilon \in (0,1]$ and define the set of radii $R_\varepsilon = \{(1+\varepsilon/3)^z \mid z \in \mathbb{Z}\}$. Lemma 3.2.4 defines an approximation of $\mathrm{dtw}_p(\sigma,\tau)$ for any $\sigma \in \mathbb{X}^{d,\ell}$ and $\tau \in \mathbb{X}^{d,m}$, by virtue of the functions g and $f_1, ..., f_k$ for $F_{\varepsilon/6(m+\ell)^{1/p}}$, as

$$\widetilde{\mathrm{dtw}}_p(\sigma,\tau) = (1+\varepsilon/3)\cdot \min\{r \in R_\varepsilon \mid g(\mathrm{sign}(f_1(\pi_\ell(\tau), r, \pi_m(\sigma))), \ldots) = 1\}.$$

In the following lemma, we formally show that $\widetilde{\mathrm{dtw}}_p(\sigma,\tau)$ approximates p-DTW between σ and τ within a factor of $1+\varepsilon$.

Lemma 3.2.6. *Let $\varepsilon \in (0,1]$. For any $\sigma \in \mathbb{X}^{d,m}$ and $\tau \in \mathbb{X}^{d,\ell}$ it holds that*

$$\mathrm{dtw}_p(\sigma,\tau) < \widetilde{\mathrm{dtw}}_p(\sigma,\tau) < (1+\varepsilon)\,\mathrm{dtw}_p(\sigma,\tau).$$

Proof. Let $r^* = \widetilde{\mathrm{dtw}}_p(\sigma,\tau)$ and let $\hat{\varepsilon} = \varepsilon/3$. By definition of $\widetilde{\mathrm{dtw}}_p$ the function g (with the class of functions with $F_{\varepsilon/6(m+\ell)^{1/p}} = F_{\hat{\varepsilon}/2(m+\ell)^{1/p}}$) from Definition 3.2.5 outputs 0 with σ, $r^*/(1+\hat{\varepsilon})^2$, and τ. However, the function g outputs 1 with σ, $r^*/(1+\hat{\varepsilon})$, and τ. Hence $r^*/(1+\hat{\varepsilon})^2 < \mathrm{dtw}_p(\sigma,\tau)$ and $r^*/(1+\hat{\varepsilon}) \geq \mathrm{dtw}_p(\sigma,\tau)/(1+\hat{\varepsilon}/2)$. This in turn implies that

$$\mathrm{dtw}_p(\sigma,\tau) \leq (1+\hat{\varepsilon})r^*/(1+\hat{\varepsilon}/2) < r^* < (1+\hat{\varepsilon})^2\,\mathrm{dtw}_p(\sigma,\tau) < (1+\varepsilon)\,\mathrm{dtw}_p(\sigma,\tau). \quad \square$$

From the definition of $\widetilde{\mathrm{dtw}}_p$, we conclude that g serves as a membership predicate for balls defined by $\widetilde{\mathrm{dtw}}_p$.

Lemma 3.2.7. *Let $\varepsilon \in (0,1]$, $\tau \in \mathbb{X}^{d,\ell}$ and $r \in R_\varepsilon$. For any $\sigma \in \mathbb{X}^{d,m}$ the output of the function g of Definition 3.2.5 with σ, τ and r is 1 iff $\widetilde{\mathrm{dtw}}_p(\sigma,\tau) \leq (1+\varepsilon/3)r$, i.e., iff $\sigma \in \{x \in \mathbb{X}^{d,m} \mid \widetilde{\mathrm{dtw}}_p(x,\tau) \leq (1+\varepsilon/3)r\}$.*

Proof. Let $r^* = \widetilde{\mathrm{dtw}}_p(\sigma,\tau)$ and $\hat{\varepsilon} = \varepsilon/3$. Observe that g outputs 0 for σ, τ and any $r \in R_\varepsilon$ if $r \leq r^*/(1+\hat{\varepsilon})^2$, and 1 for $r = r^*/(1+\hat{\varepsilon})$. Now by Lemma 3.2.6 we know that $\mathrm{dtw}_p(\sigma,\tau) < r^*$. Hence g outputs 1 for σ, τ and any $r \geq r^*$. Thus we conclude that g outputs 1 for σ, τ and $r \in R_\varepsilon$ if and only if $r \geq \widetilde{\mathrm{dtw}}_p(\sigma,\tau)/(1+\hat{\varepsilon}) = \widetilde{\mathrm{dtw}}_p(\sigma,\tau)/(1+\varepsilon/3)$. $\square$

We conclude with the main result of this section, namely an upper bound on the VC dimension of the range space that approximates the p-DTW range space.

Theorem 3.2.8. *Let $\varepsilon \in (0,1]$. The VC dimension of the range space*

$$\left(\mathbb{X}^{d,m}, \left\{\{x \in \mathbb{X}^{d,m} \mid \widetilde{\mathrm{dtw}}_p(x,\tau) \leq r\} \subset \mathbb{X}^{d,m} \middle| \tau \in \mathbb{X}^{d,\ell}, r > 0\right\}\right)$$

is at most

$$2(d+1)\ell \log_2(12\ell m\lfloor (m+\ell)^{1/p}\varepsilon^{-1}+1\rfloor + 12m + 12\ell) = \mathcal{O}(d\ell\log(\ell m\varepsilon^{-1})).$$

Proof. This follows from Theorem 3.2.2, Lemma 3.2.4 and Lemma 3.2.7, and the fact that any ball of radius $r > 0$ under $\widetilde{\mathrm{dtw}}_p$ coincides with some ball with radius $\widetilde{r} \in R_\varepsilon$ under $\widetilde{\mathrm{dtw}}_p$. Finally, the statement is implied by the injectivity of the functions π_m and π_ℓ. □

In this section, we defined a distance function $\widetilde{\mathrm{dtw}}_p$ between curves in $\mathbb{X}^{d,m}$ and those in $\mathbb{X}^{d,\ell}$, which $(1+\varepsilon)$-approximates dtw_p. We also derived an upper bound on the VC dimension of the range space induced by balls of $\widetilde{\mathrm{dtw}}_p$, thereby producing an approximation of the p-DTW range space that we make use of below. We emphasize that the sole purpose of $\widetilde{\mathrm{dtw}}_p$ is to obtain bounds on the size of a sample constituting a coreset through the knowledge of the VC dimension. At no point do we compute $\widetilde{\mathrm{dtw}}_p$.

3.3 Sensitivity Bounds and Coresets for DTW

To make use of the sensitivity sampling framework for coresets by Feldman and Langberg [FL11], we recast the input set $T \subset \mathbb{X}^{d,m}$ to a set of functions. Consider for any $y \in \mathbb{X}^{d,m}$ the real valued function f_y defined on (finite) subsets of $\mathbb{X}^{d,\ell}$ where $f_y(C) = \min_{c\in C} \mathrm{dtw}_p(y,c)$ for $C \subset \mathbb{X}^{d,\ell}$, transforming T into $F_T = \{f_\tau \mid \tau \in T\}$. To construct a coreset, one draws elements from T according to a fixed probability distribution over T, and reweighs each drawn element. Both the weight and sampling probability are expressed in terms of the ***sensitivity*** of the drawn element t, which describes the maximum possible relative contribution of t to the cost of any set of curves $C \subset \mathbb{X}^{d,\ell}$. In our case, as we restrict a solution to a size of k, it turns out that it suffices to analyze the sensitivity with respect to inputs of size k.

Definition 3.3.1 (sensitivity)**.** Let F be a finite set of functions from $\mathcal{P}\left(\mathbb{X}^{d,\ell}\right) \setminus \{\emptyset\}$ to $\mathbb{R}$. For any $f \in F$ define the sensitivity

$$\mathfrak{s}(f,F) = \sup_{C=\{c_1,\ldots,c_k\}\subset Z:\sum_{g\in F} g(C)>0} \frac{f(C)}{\sum_{g\in F} g(C)}.$$

The total sensitivity $\mathfrak{S}(F)$ of F is defined as $\sum_{f\in F} \mathfrak{s}(f,F)$.

A crucial step in our approach is to show that any (α,β)-approximation for (k,ℓ)-median under dtw_p can be used to obtain a bound on the total sensitivity associated to approximate distances. This is facilitated by the following lemma that is a weaker version of the triangle inequality, as in general dtw_p is not a metric (see Figure 3.2).

Lemma 3.3.2 (weak triangle inequality [Lem09])**.** *For two curves $x, z \in \mathbb{X}^{d,m}$ and any curve $y \in \mathbb{X}^{d,\ell}$ it holds that*

$$\mathrm{dtw}_p(x,z) \le m^{1/p}(\mathrm{dtw}_p(x,y) + \mathrm{dtw}_p(y,z)).$$

Note that the distance function we bounded the VC dimension of is not dtw_p, but the $(1+\varepsilon)$-approximation $\widetilde{\mathrm{dtw}}_p$ of dtw_p. Hence we also analyze the functions $\widetilde{f}_y$ corresponding to the approximation $\widetilde{\mathrm{dtw}}_p$; for any $y \in \mathbb{X}^{d,m}$ and $\varepsilon > 0$, let $\widetilde{f}_y : \mathcal{P}(\mathbb{X}^{d,\ell}) \setminus \{\emptyset\} \to \mathbb{R}$ where $\widetilde{f}_y(C) = \min_{c\in C} \widetilde{\mathrm{dtw}}_p(y,c)$. Similarly, let $\widetilde{F}_T$ be the set $\{\widetilde{f}_\tau \mid \tau \in T\}$ for any $T \subset \mathbb{X}^{d,m}$. Note that by Lemma 3.2.6 $f_\tau \le \widetilde{f}_\tau \le (1+\varepsilon) f_\tau$. We now analyze the sensitivity of the corresponding functions w.r.t. $\widetilde{\mathrm{dtw}}_p$ only in terms of dtw_p.

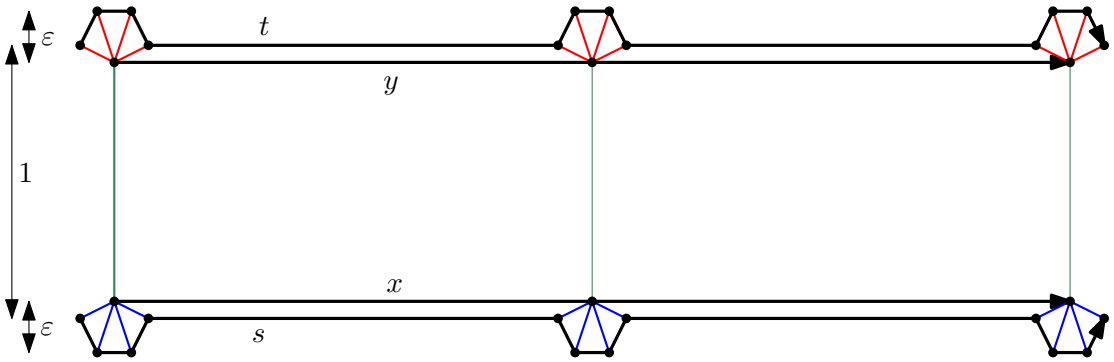


Figure 3.2: Violated triangle inequality as $\mathrm{dtw}(s,t) \approx 12$, but $\mathrm{dtw}(s,x) \approx 0$ (matching in blue), $\mathrm{dtw}(y,t) \approx 0$ (red matching) and $\mathrm{dtw}(x,y) \approx 3$ (green matching).

Lemma 3.3.3. *Let $\varepsilon \in (0,1]$ and let $T \subset \mathbb{X}^{d,m}$ be the input of size n for (k,ℓ)-median and let $\hat{C} = \{\hat{c}_1, \ldots, \hat{c}_{\hat{k}}\} \subset \mathbb{X}^{d,\ell}$ be an (α,β)-approximation to the (k,ℓ)-median problem of T with cost $\hat{\Delta} = \sum_{\tau\in T} \min_{\hat{c}\in\hat{C}} \mathrm{dtw}_p(\tau,\hat{c})$, of size $\hat{k} \leq \beta k$. For any $i \in [\hat{k}]$ let $\hat{V}_i = \{\tau \in T \mid \mathrm{dtw}_p(\tau,\hat{c}_i) = \min_{\hat{c}\in\hat{C}} \mathrm{dtw}_p(\tau,\hat{c})\}$ be the Voronoi region of $\hat{c}_i$, where ties are broken arbitrarily. These sets partitions T. Let $\hat{\Delta}_i = \sum_{\tau\in\hat{V}_i} \mathrm{dtw}_p(\tau,\hat{c}_i)$ be the cost of $\hat{V}_i$. For all $\tau \in \hat{V}_i$ define*

$$\gamma(\widetilde{f}_\tau) := (m\ell)^{1/p}\left(\frac{2\alpha\,\mathrm{dtw}_p(\tau,\hat{c}_i)}{\hat{\Delta}} + \frac{4}{|\hat{V}_i|} + \frac{8\alpha\hat{\Delta}_i}{\hat{\Delta}|\hat{V}_i|}\right).$$

Then $\mathfrak{s}(\widetilde{f}_\tau, \widetilde{F}_T) \leq \gamma(\widetilde{f}_\tau)$ for any $\tau \in T$, and $\mathfrak{S}(\widetilde{F}_T) \leq \sum_{\tau\in T} \gamma(\widetilde{f}_\tau) \leq (m\ell)^{1/p}(4\hat{k} + 10\alpha)$.

Proof. Fix an arbitrary set $C = \{c_1, \ldots, c_k\} \subseteq \mathbb{X}^{d,\ell}$, $i \in [\hat{k}]$. Let further $\hat{B}_i = \{\sigma \in \hat{V}_i \mid \mathrm{dtw}_p(\sigma,\hat{c}_i) \leq 2\hat{\Delta}_i/|\hat{V}_i|\}$. Observe that for any $\sigma \in \hat{B}_i$ it holds that $\widetilde{f}_\sigma(\hat{C}) \leq (1+\varepsilon)2\hat{\Delta}_i/|\hat{V}_i| \leq 4\hat{\Delta}_i/|\hat{V}_i|$. Breaking ties arbitrarily, let $c(x) \in C$ be the nearest neighbor of $x \in X$ among C.

We observe that $|\hat{B}_i| \geq |\hat{V}_i|/2$, since otherwise $\sum_{\sigma\in\hat{V}_i\setminus\hat{B}_i} \mathrm{dtw}_p(\sigma,\hat{c}_i) > \hat{\Delta}_i$. Additionally note that $\sum_{\widetilde{f}_\sigma\in\widetilde{F}_T} \widetilde{f}_\sigma(C) \geq \hat{\Delta}/\alpha$.

For any $\sigma \in \hat{B}_i$ Lemma 3.3.2 implies that $\widetilde{f}_\sigma(C) \geq \mathrm{dtw}_p(\sigma,c(\sigma)) \geq \frac{\mathrm{dtw}_p(\hat{c}_i,c(\sigma))}{\ell^{1/p}} - \frac{4\hat{\Delta}_i}{|\hat{V}_i|}$. Now as $\mathrm{dtw}_p(\hat{c}_i,c(\sigma)) \geq \mathrm{dtw}_p(\hat{c}_i,c(\hat{c}_i))$ it holds that

$$\sum_{\widetilde{f}_\sigma\in\widetilde{F}_T} \widetilde{f}_\sigma(C) \geq \max\left\{\sum_{\sigma\in\hat{B}_i} \widetilde{f}_\sigma(C), \frac{\hat{\Delta}}{\alpha}\right\} \geq \max\left\{\frac{|\hat{V}_i|}{2}\left(\frac{\mathrm{dtw}_p(\hat{c}_i,c(\hat{c}_i))}{\ell^{1/p}} - \frac{4\hat{\Delta}_i}{|\hat{V}_i|}\right), \frac{\hat{\Delta}}{\alpha}\right\}.$$

Let us now fix some $\tau \in \hat{V}_i$. Let $(2\ell^{1/p})\frac{\hat{\Delta}+2\alpha\hat{\Delta}_i}{\alpha|\hat{V}_i|} =: \delta_i$ and consider the function

$$h_{i,\tau} : [\delta_i,\infty) \to \mathbb{R}_{>0}, x \mapsto \frac{2m^{1/p}(\ell^{1/p}\,\mathrm{dtw}_p(\tau,\hat{c}_i) + x)}{\frac{|\hat{V}_i|x}{2\ell^{1/p}} - 2\hat{\Delta}_i}$$

and observe that it is monotone and thus its maximum is either $h_{i,\tau}(\delta_i)$ or $\lim_{x\to\infty} h_{i,\tau}(x)$. Assume now that $\frac{|\hat{V}_i|}{2}\left(\frac{\mathrm{dtw}_p(\hat{c}_i,c(\hat{c}_i))}{\ell^{1/p}} - \frac{4\hat{\Delta}_i}{|\hat{V}_i|}\right) \geq \frac{\hat{\Delta}}{\alpha}$, i.e., $\mathrm{dtw}_p(\hat{c}_i,c(\hat{c}_i)) \geq \delta_i$. Lemma 3.3.2

implies that $\widetilde{f}_\tau(C) \leq (1+\varepsilon)m^{1/p}(\mathrm{dtw}_p(\tau,\hat{c}_i) + \mathrm{dtw}_p(\hat{c}_i, c(\hat{c}_i)))$ and hence

$$\frac{\widetilde{f}_\tau(C)}{\sum_{\widetilde{f}_\sigma \in \widetilde{F}_T} \widetilde{f}_\sigma(C)} \leq \frac{2m^{1/p}(\mathrm{dtw}_p(\tau,\hat{c}_i) + \mathrm{dtw}_p(\hat{c}_i, c(\hat{c}_i)))}{\frac{|\hat{V}_i|}{2}\left(\frac{\mathrm{dtw}_p(\hat{c}_i,c(\hat{c}_i))}{\ell^{1/p}} - \frac{4\hat{\Delta}_i}{|\hat{V}_i|}\right)} \leq h_{i,\tau}(\mathrm{dtw}_p(\hat{c}_i, c(\hat{c}_i))).$$

If instead $\mathrm{dtw}_p(\hat{c}_i, c(\hat{c}_i)) < \delta_i$, then

$$\frac{\widetilde{f}_\tau(C)}{\sum_{\widetilde{f}_\sigma \in \widetilde{F}_T} \widetilde{f}_\sigma(C)} \leq \frac{\widetilde{f}_\tau(C)}{\hat{\Delta}/\alpha} < \frac{(1+\varepsilon)m^{1/p}(\mathrm{dtw}_p(\tau,\hat{c}_i) + \delta_i)}{\hat{\Delta}/\alpha} \leq h_{i,\tau}(\delta_i).$$

Thus it follows that

$$\begin{aligned}\mathfrak{s}(\widetilde{f}_\tau, \widetilde{F}_T) &= \sup_{C=\{c_1,\ldots,c_k\}\subseteq Z} \frac{\widetilde{f}_\tau(C)}{\sum_{\widetilde{f}_\sigma \in \widetilde{F}_T} \widetilde{f}_\sigma(C)} \leq \max\left\{h_{i,\tau}(\delta_i), \lim_{x\to\infty} h_{i,\tau}(x)\right\} \\ &= \max\left\{(m\ell)^{1/p}\left(\frac{2\alpha\,\mathrm{dtw}_p(\tau,\hat{c}_i)}{\hat{\Delta}} + \frac{4}{|\hat{V}_i|} + \frac{8\alpha\hat{\Delta}_i}{\hat{\Delta}|\hat{V}_i|}\right), \frac{4(m\ell)^{1/p}}{|\hat{V}_i|}\right\} = \gamma(\widetilde{f}_\tau).\end{aligned}$$

Overall it follows that

$$\begin{aligned}\mathfrak{S}(\widetilde{F}_T) &\leq (m\ell)^{1/p}\sum_{i=1}^{\hat{k}}\sum_{\sigma\in\hat{V}_i}\left(\frac{2\alpha\,\mathrm{dtw}_p(\tau,\hat{c}_i)}{\hat{\Delta}} + \frac{4}{|\hat{V}_i|} + \frac{8\alpha\hat{\Delta}_i}{\hat{\Delta}|\hat{V}_i|}\right) \\ &= (m\ell)^{1/p}\sum_{i=1}^{\hat{k}}\left(\frac{2\alpha\hat{\Delta}_i}{\hat{\Delta}} + 4 + \frac{8\alpha\hat{\Delta}_i}{\hat{\Delta}}\right) = (m\ell)^{1/p}\left(2\alpha + 4\hat{k} + 8\alpha\right). \qquad \square\end{aligned}$$

Lemma 3.3.4. *Let $\varepsilon \in (0,1]$. A weighted ε-coreset S for (k,ℓ)-median of T under the approximate distance $\widetilde{\mathrm{dtw}}_p$ is a weighted 3ε-coreset for (k,ℓ)-median of T under dtw_p.*

Proof. Let $C \subset \mathbb{X}^{d,\ell}$ with $|C| = k$ be given. Let $w(\widetilde{f}_\tau)$ for every $\tau \in S$ be the weights of S. In particular

$$(1-\varepsilon)\sum_{\tau\in T}\widetilde{f}_\tau(C) \leq \sum_{\tau\in S} w(\widetilde{f}_\tau)\widetilde{f}_\tau(C) \leq (1+\varepsilon)\sum_{\tau\in T}\widetilde{f}_\tau(C).$$

Then by Lemma 3.2.6

$$\begin{aligned}(1-2\varepsilon)\sum_{\tau\in T} f_\tau(C) &\leq \frac{1-\varepsilon}{1+\varepsilon}\sum_{\tau\in T} f_\tau(C) \leq \frac{1-\varepsilon}{1+\varepsilon}\sum_{\tau\in T}\widetilde{f}_\tau(C) \\ &\leq \frac{1}{1+\varepsilon}\sum_{\tau\in S} w(\widetilde{f}_\tau)\widetilde{f}_\tau(C) \leq \sum_{\tau\in S} w(\widetilde{f}_\tau) f_\tau(C),\end{aligned}$$

and

$$\begin{aligned}\sum_{\tau\in S} w(\widetilde{f}_\tau) f_\tau(C) &\leq \sum_{\tau\in S} w(\widetilde{f}_\tau)\widetilde{f}_\tau(C) \leq (1+\varepsilon)\sum_{\tau\in T}\widetilde{f}_\tau(C) \\ &\leq (1+\varepsilon)^2\sum_{\tau\in T} f_\tau(C) \leq (1+3\varepsilon)\sum_{\tau\in T} f_\tau(C). \qquad \square\end{aligned}$$

Definition 3.3.5 ([HPS11, Definition 2.3])**.** Let $\varepsilon, \eta \in (0,1)$ and $(X, \mathcal{R})$ be a range space with finite non-empty ground set. An (η, ε)-approximation of $(X, \mathcal{R})$ is a set $S \subseteq X$, such that for all $R \in \mathcal{R}$

$$\left| \frac{|R \cap X|}{|X|} - \frac{|R \cap S|}{|S|} \right| \leq \begin{cases} \varepsilon \cdot \frac{|R \cap X|}{|X|}, & \text{if } |R \cap X| \geq \eta \cdot |X| \\ \varepsilon \cdot \eta, & \text{else.} \end{cases}$$

We employ the following theorem to obtain (η, ε)-approximations.

Theorem 3.3.6 ([HPS11, Theorem 2.11])**.** *Let $(X, \mathcal{R})$ be a range space with finite non-empty ground set and VC dimension $\mathcal{D}$. Let $\varepsilon, \delta, \eta \in (0,1)$. There is an absolute constant $c \in \mathbb{R}_{>0}$ such that a sample of*

$$\frac{c}{\eta \cdot \varepsilon^2} \cdot \left(\mathcal{D} \log\left(\frac{1}{\eta}\right) + \log\left(\frac{1}{\delta}\right) \right)$$

elements drawn independently and uniformly at random with replacement from X is a (η, ε)-approximation for $(X, \mathcal{R})$ with probability at least $1 - \delta$.

Theorem 3.3.7. *For $\widetilde{f} \in \widetilde{F}$, let $\lambda(\widetilde{f}) = 2^{\lceil \log_2(\gamma(\widetilde{f})) \rceil}$, with $\gamma(\widetilde{f})$ the sensitivity bound of Lemma 3.3.3, associated to an (α, β)-approximation consisting of $\hat{k} \leq \beta k$ curves, for (k,ℓ)-median for curves in $\mathbb{X}^{d,m}$ under dtw_p, $\Lambda = \sum_{\widetilde{f} \in \widetilde{F}} \lambda(\widetilde{f})$, $\psi(\widetilde{f}) = \frac{\lambda(\widetilde{f})}{\Lambda}$ and $\delta, \varepsilon \in (0,1)$. A sample S of*

$$\Theta\Big(\varepsilon^{-2} \alpha \hat{k} (m\ell)^{1/p} \left((d\ell \log(\ell m \varepsilon^{-1})) k \log(k) \log(\alpha n) \log(\alpha \hat{k} (m\ell)^{1/p}) + \log(1/\delta) \right) \Big)$$

elements $\tau_i \in T$, drawn independently with replacement with probability $\psi(\widetilde{f}_i)$ and endowed with the weight $w(\widetilde{f}_i) = \frac{\Lambda}{|S| \lambda(\widetilde{f}_i)}$ is a weighted ε-coreset for (k,ℓ)-median clustering of T under dtw_p with probability at least $1 - \delta$.

Let $\widetilde{\mathrm{cost}}_p(T, C) = \sum_{\tau \in T} \widetilde{f}_\tau(C)$ for $T = \{\tau_1, \ldots, \tau_n\} \subseteq \mathbb{X}^{d,m}$, and let $\widetilde{F} = \{\widetilde{f}_1, \ldots, \widetilde{f}_n\}$, with $\widetilde{f}_i = \widetilde{f}_{\tau_i}$.

Our proof relies on the reduction to uniform sampling, introduced by [FL11] and improved by [BFL+16], allowing us to apply Theorem 3.3.6. In the following, we adapt and modify the proof of [FSS20, Theorem 31] and combine it with results from [MSSW18] to handle the involved scaling, similarly to [BR22, Theorem 4].

The proof is structured as follows: We first show that S can be thought of as a uniform sample S' from a multiset G of functions. We secondly show that we can think of the functions $g \in G$ as integrals of membership functions of a range space with ground set G. Next we show that a (η, ε)-approximation S' of the range space is a coreset under $\widetilde{\mathrm{dtw}}_p$ before finally bounding the VC dimension of the range space and with it the required size of S'.

Proof. We begin by analyzing different estimators for $\widetilde{\mathrm{cost}}_p(T, C)$ for $C \subseteq \mathbb{X}^{d,\ell}$ arbitrary with $|C| = k$. Consider first the estimator

$$\widehat{\mathrm{cost}}_p(S, C) = \sum_{\tau_i \in S} w(\widetilde{f}_i) \cdot \min_{c \in C} \widetilde{\mathrm{dtw}}_p(\tau_i, c) = \sum_{\tau_i \in S} w(\widetilde{f}_i) \cdot \widetilde{f}_i(C) = \sum_{\tau_i \in S} \frac{\Lambda}{|S| \lambda(\widetilde{f}_i)} \widetilde{f}_i(C)$$

for $\widetilde{\mathrm{cost}}_p(T,C)$. We see that $\widehat{\mathrm{cost}}_p(S,C)$ is unbiased by virtue of

$$\mathrm{E}\left[\widehat{\mathrm{cost}}_p(S,C)\right] = \sum_{i=1}^{|S|}\sum_{\tau_j\in T}\psi(\widetilde{f}_j)\frac{\Lambda}{|S|\lambda(\widetilde{f}_j)}\widetilde{f}_j(C) = \sum_{i=1}^{|S|}\sum_{\tau_j\in T}\frac{\widetilde{f}_j(C)}{|S|} = \widetilde{\mathrm{cost}}_p(T,C).$$

We next reduce the sensitivity sampling to uniform sampling by letting G be a multiset that is a copy of $\widetilde{F}$, where each $\widetilde{f}\in\widetilde{F}$ is contained $|\widetilde{F}|\lambda(\widetilde{f})$ times and is scaled by $\frac{1}{|\widetilde{F}|\lambda(\widetilde{f})}$, so that $|G| = |\widetilde{F}|\Lambda$ and $\psi(\widetilde{f}) = \frac{|\widetilde{F}|\lambda(\widetilde{f})}{|G|}$. We clearly have

$$\sum_{g\in G} g(C) = \sum_{\widetilde{f}\in\widetilde{F}}\frac{|\widetilde{F}|\lambda(\widetilde{f})}{|\widetilde{F}|\lambda(\widetilde{f})}\widetilde{f}(C) = \sum_{\widetilde{f}\in\widetilde{F}}\widetilde{f}(C) = \widetilde{\mathrm{cost}}_p(T,C).$$

For a sample S', with $|S'| = |S|$, drawn independently and uniformly at random with replacement from G, consider the estimator for $\widetilde{\mathrm{cost}}_p(T,C)$ defined by

$$\overline{\mathrm{cost}}_p\big(S',C\big) = \frac{|G|}{|S'|}\sum_{g\in S'} g(C),$$

where again $C\subseteq\mathbb{X}^{d,\ell}$ with $|C| = k$. We see that $\overline{\mathrm{cost}}_p(S',C)$ is unbiased by virtue of

$$\mathrm{E}\left[\overline{\mathrm{cost}}_p\big(S',C\big)\right] = \frac{|G|}{|S'|}\sum_{i=1}^{|S'|}\sum_{\widetilde{f}\in\widetilde{F}}\frac{\widetilde{f}(C)}{|\widetilde{F}|\lambda(\widetilde{f})}\frac{|\widetilde{F}|\lambda(\widetilde{f})}{|G|} = \frac{1}{|S'|}\sum_{i=1}^{|S'|}\sum_{\widetilde{f}\in\widetilde{F}}\widetilde{f}(C) = \frac{|S'|}{|S'|}\widetilde{\mathrm{cost}}_p(T,C).$$

We now assume that $S' = \left\{\frac{1}{|\widetilde{F}|\lambda(\widetilde{f}_i)}\cdot\widetilde{f}_i\middle|\tau_i\in S\right\}$, which yields

$$\overline{\mathrm{cost}}_p\big(S',C\big) = \frac{|\widetilde{F}|\Lambda}{|S'|}\sum_{g\in S'} g(C) = \sum_{\tau_i\in S}\frac{\Lambda}{|S|\lambda(\widetilde{f}_i)}\widetilde{f}_i(C) = \widehat{\mathrm{cost}}_p(S,C). \tag{I}$$

For any subset $H\subseteq G$, $C\subseteq\mathbb{X}^{d,\ell}$ with $|C| = k$ and $r\in\mathbb{R}_{\geq 0}$, define the set $\mathrm{range}(H,C,r) = \{g\in H\mid g(C)\geq r\}$. Observe that $\mathrm{range}(H,C,r) = \mathrm{range}(G,C,r)\cap H$. For all such $C\subseteq Z$ and all $H\subseteq G$, we have that

$$\sum_{g\in H} g(C) = \sum_{g\in H}\int_0^\infty \mathbb{1}(g(C)\geq r)\,\mathrm{d}r = \int_0^\infty |\mathrm{range}(H,C,r)|\,\mathrm{d}r, \tag{II}$$

where all of the involved functions are integrable. Consider now the range space $(G,\mathcal{R})$ over G, where $\mathcal{R} = \{\mathrm{range}(G,C,r)\mid r\in\mathbb{R}_{\geq 0}, C\subseteq Z, |C| = k\}$. For the following, we apply Theorem 3.3.6 with the given δ, $\varepsilon/2$ and $\eta = 1/\Lambda$, so as to guarantee that S' is a $(1/\Lambda,\varepsilon/2)$-approximation of $(G,\mathcal{R})$. Given $C\subseteq Z$ with $|C| = k$, we compute that

$$\begin{aligned}
\left|\widetilde{\mathrm{cost}}_p(T,C) - \widehat{\mathrm{cost}}_p(S,C)\right| &\overset{\text{(I)}}{=} \left|\widetilde{\mathrm{cost}}_p(T,C) - \overline{\mathrm{cost}}_p\big(S',C\big)\right| = \left|\sum_{g\in G} g(C) - \frac{|G|}{|S'|}\sum_{g\in S'} g(C)\right| \\
&= \left|\int_0^\infty |\mathrm{range}(G,C,r)|\,\mathrm{d}r - \frac{|G|}{|S'|}\int_0^\infty |\mathrm{range}\big(S',C,r\big)|\,\mathrm{d}r\right| \\
&= \left|\int_0^\infty |\mathrm{range}(G,C,r)| - \frac{|G|}{|S'|}|\mathrm{range}\big(S',C,r\big)|\,\mathrm{d}r\right| \\
&\leq \int_0^\infty \left||\mathrm{range}(G,C,r)| - \frac{|G|}{|S'|}|\mathrm{range}\big(S',C,r\big)|\right|\mathrm{d}r.
\end{aligned}$$

As $r \mapsto |\text{range}(G,C,r)|$ is monotone, $R_1(C) = \{r \in \mathbb{R}_{\geq 0} \mid |\text{range}(G,C,r)| \geq \eta \cdot |G|\}$ and $R_2(C) = \mathbb{R}_{\geq 0} \setminus R_1(C)$ are intervals. Denoting $r_u(C) = \max_{g \in G} g(C)$, we have that for $r \in (r_u(C), \infty)$, it holds that $|\text{range}(G,C,r)| = 0$. Therefore,

$$\left| \frac{|\text{range}(G,C,r)|}{|G|} - \frac{|\text{range}(S',C,r)|}{|S'|} \right| \leq \frac{\varepsilon}{2} \frac{|\text{range}(G,C,r)|}{|G|},$$

for $r \in R_1(C)$, since S' is a $(1/\Lambda, \varepsilon/2)$-approximation of the range space $(G, \mathcal{R})$ and similarly for $r \in R_2(C)$. Thus,

$$\begin{aligned}
\left| \widetilde{\text{cost}}_p(T,C) - \widehat{\text{cost}}_p(S,C) \right| &\leq \int_{R_1(C)} \left| |\text{range}(G,C,r)| - \frac{|G|}{|S'|} |\text{range}\big(S',C,r\big)| \right| \mathrm{d}r \\
&\quad + \int_{R_2(C)} \frac{\varepsilon}{2} \eta |G| \mathrm{d}r \\
&\leq \frac{\varepsilon}{2} \int_0^\infty |\text{range}(G,C,r)| \ \mathrm{d}r + \frac{\varepsilon \eta |G|}{2} \int_0^{r_u(C)} \mathrm{d}r \\
&= \frac{\varepsilon}{2} \sum_{g \in G} g(C) + \frac{\varepsilon \eta |G| r_u(C)}{2}, \qquad \text{(III)}
\end{aligned}$$

where the last equality is due to (II). We now bound the last term in (III) with the help of the sensitivity bounds derived in Lemma 3.3.3, where we use that $\gamma(\widetilde{f}) \leq \lambda(\widetilde{f})$. For each $g \in G$, we have

$$\frac{g(C)}{\sum_{g \in G} g(C)} = \frac{\frac{1}{|\widetilde{F}|\lambda(\widetilde{f})} \widetilde{f}(C)}{\sum_{\widetilde{f} \in \widetilde{F}} \widetilde{f}(C)} \leq \frac{1}{|\widetilde{F}|\lambda(\widetilde{f})} \lambda(\widetilde{f}) = \frac{1}{|\widetilde{F}|},$$

where $\widetilde{f} \in \widetilde{F}$ is the function that g is a copy of, implying that $r_u(C) \leq \frac{1}{|\widetilde{F}|} \sum_{g \in G} g(C)$. Thus,

$$\frac{\varepsilon \eta |G| r_u(C)}{2} \leq \frac{\varepsilon}{2} \frac{1}{\Lambda} |\widetilde{F}| \Lambda \frac{1}{|\widetilde{F}|} \sum_{g \in G} g(C) = \frac{\varepsilon}{2} \sum_{g \in G} g(C).$$

All in all, (III) implies that $|\widetilde{\text{cost}}_p(T,C) - \widehat{\text{cost}}_p(S,C)| \leq \varepsilon \cdot \sum_{g \in G} g(C) = \varepsilon \cdot \widetilde{\text{cost}}_p(T,C)$ for all $C \subseteq Z$ with $|C| = k$, with probability at least $1 - \delta$, so S is an ε-coreset for the approximate distance function.

By Lemma 3.3.4, upon rescaling ε by $1/3$, it remains to show the asserted bounds on the size of S'. Observe that $\gamma(\widetilde{f}) \leq \lambda(\widetilde{f}) \leq 2 \cdot \gamma(\widetilde{f})$ so that $\Lambda \leq 2\Gamma(\widetilde{F}) = \mathcal{O}(\alpha \hat{k}(m\ell)^{1/p})$, by Lemma 3.3.3. Thus, assuming the VC dimension of the range space $(G, \mathcal{R})$ is in $\mathcal{O}(d\ell \log(\ell m \varepsilon^{-1}) k \log(k) \log(\alpha n))$, Theorem 3.3.6 implies the claim.

To bound the VC dimension, let us first consider the simple case that for all $\widetilde{f} \in \widetilde{F}$, $\lambda(\widetilde{f}) = \widetilde{c}$ for some $\widetilde{c}$, so that the scaling of the elements of G is uniform and can be ignored in the context of the VC dimension. For given $r \geq 0$ and $c \in \mathbb{X}^{d,\ell}$, let $\mathrm{D}_r(c) = \{\sigma \in \mathbb{X}^{d,m} | \, \widetilde{\text{dtw}}_p(\sigma, c) \leq r\} \cap T$. The range space $(G, \mathcal{R})$ can then alternatively be described as $(T, \{T \setminus \bigcup_{c \in C} \widetilde{B}_{r,T}(c)) | C \subset \mathbb{X}^{d,\ell}, |C| = k, r \in \mathbb{R}_{\geq 0}\})$, which in turn has VC dimension at most equal to that of $(\mathbb{X}^{d,m}, \{\mathbb{X}^{d,m} \setminus \bigcup_{i \in [k]} \mathrm{D}_i | \mathrm{D}_1, ..., \mathrm{D}_k\})$, with each D_i of the form $\mathrm{D}_r(c)$ for a $c \in \mathbb{X}^{d,\ell}$. The last range space has the same VC as its complementary range

space, which by the k-fold union theorem [BEHW89, Lemma 3.2.3] has VC dimension at most $2\mathcal{D}k\log_2(3k) \leq c\mathcal{D}k\log k \in \mathcal{O}(\mathcal{D}k\log k)$, where $\mathcal{D} = \mathcal{O}(d\ell\log(\ell m\varepsilon^{-1}))$ is the VC dimension of balls in $\mathbb{X}^{d,m}$ under $\widetilde{\mathrm{dtw}}_p$ by Theorem 3.2.8.

Let now t denote the number of distinct values $\{c_1,...,c_t\}$ of $\lambda(\widetilde{f})$, as $\widetilde{f}$ ranges over $\widetilde{F}$ and partition G into the sets $\{G_1,...,G_t\}$ such that for all $g \in G_i$ there is a $\widetilde{f} \in \widetilde{F}$ with $g = \frac{1}{|\widetilde{F}|\lambda(\widetilde{f})}\widetilde{f} = \frac{1}{|\widetilde{F}|c_i}\widetilde{f}$. Assume, for the sake of contradiction, that $G' \subset G$ is a set with $|G'| > t \cdot c\mathcal{D}k\log k$ that is shattered by $\mathcal{R}$. Consider the sets $G'_i = G' \cap G_i$ as well as induced range spaces $\mathcal{R}_i = G_i \cap \mathcal{R}$ on each G_i for $i \in [t]$. Since the G_i are disjoint, each G'_i is shattered by $\mathcal{R}_i$ and there must exist at least one $j \in [t]$ such that $|G'_j| \geq \frac{|G'|}{t} > \frac{t\cdot c\mathcal{D}k\log k}{t} = c\mathcal{D}k\log k$. However, this contradicts the VC dimension of $(G_j,\mathcal{R}_j)$ in the case that the scaling of the functions in G is uniform, established above. We now derive explicit bounds on t. By Lemma 3.3.3, for $\tau \in \hat{V}_i$ with $i \in [k']$, we have

$$(m\ell)^{1/p}\frac{4}{|\hat{V}_i|} \leq \gamma(\widetilde{f}) \leq (m\ell)^{1/p}\left(2\alpha + \frac{4}{|\hat{V}_i|} + \frac{8\alpha}{|\hat{V}_i|}\right),$$

implying that the number of distinct values of $\lambda(\widetilde{f})$ is that number for $\lceil\log_2(\gamma(\widetilde{f}))\rceil$, which is

$$\begin{aligned}&\log_2\left((m\ell)^{1/p}\left(2\alpha + \frac{4}{|\hat{V}_i|} + \frac{8\alpha}{|\hat{V}_i|}\right)\right) - \log_2\left((m\ell)^{1/p}\frac{4}{|\hat{V}_i|}\right)\\ =&\log_2\left(\left(2\alpha + \frac{8\alpha}{|\hat{V}_i|} + \frac{4}{|\hat{V}_i|}\right)\left(\frac{4}{|\hat{V}_i|}\right)^{-1}\right) \leq \log_2\left(\frac{n\alpha}{2} + 2\alpha + 1\right).\end{aligned}$$

Thus, the VC dimension of $(G,\mathcal{R})$ is at most $2\mathcal{D}k\log_2(3k)\log_2(\frac{n\alpha}{2} + 2\alpha + 1)$ concluding the proof. □

We remark that in the limit $p \to \infty$, the constructed coreset has a very similar size as a recent construction for coresets for the Fréchet distance [BR22].

3.4 Linear Time $(\mathcal{O}((m\ell)^{1/p}), 1)$-Approximation Algorithm for (k,ℓ)-Median

In this section, we develop approximation algorithms for (k,ℓ)-median for a set $T \subset \mathbb{X}^{d,m}$ of n curves. For this, we approximate DTW on T by a metric using a new inequality for DTW (Lemma 3.4.6). This allows the use of any approximation algorithm for k-median in metric spaces, leading to a first approximation algorithm of the original problem. However, computing the whole metric space would take $\mathcal{O}(n^3)$ time. We circumvent this by in turn using the DTW distance to approximate the metric space. Combined with a k-median algorithm in metric spaces [Ind99], we obtain a linear-time $(\mathcal{O}((m\ell)^{1/p}), 1)$-approximation algorithm.

3.4.1 Approximate ℓ-Simplifications under p-DTW

We briefly discuss simplifications for curves under p-DTW. These simplifications will serve as candidates, to which we restrict possible solutions of the (k,ℓ)-median problem incurring a constant approximation factor only.

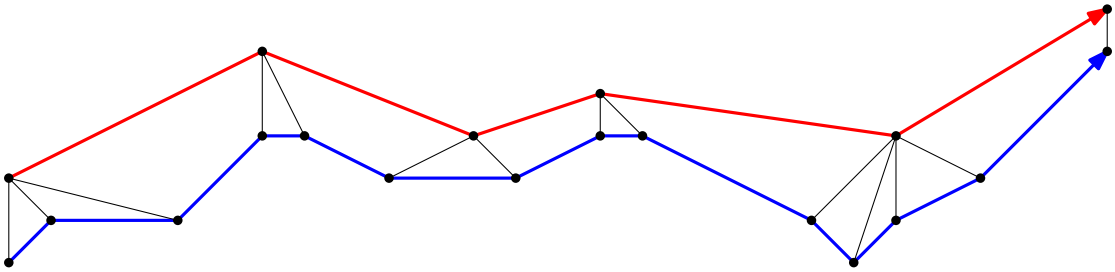


Figure 3.3: Illustration of a lopsided traversal.

Definition 3.4.1 ($(1+\varepsilon)$-approximate ℓ-simplifications)**.** Let $\sigma \in \mathbb{X}^{d,m}$, $\ell \in \mathbb{N}$ and $\varepsilon > 0$. We call $\sigma^* \in \mathbb{X}^{d,\ell}$ an ***$(1+\varepsilon)$-approximate ℓ-simplification*** if

$$\inf_{\sigma_\ell \in \mathbb{X}^{d,\ell}} \mathrm{dtw}_p(\sigma_\ell, \sigma) \leq \mathrm{dtw}_p(\sigma^*, \sigma) \leq (1+\varepsilon) \inf_{\sigma_\ell \in \mathbb{X}^{d,\ell}} \mathrm{dtw}_p(\sigma_\ell, \sigma).$$

We reduce the problem of finding a $(1+\varepsilon)$-approximate simplification to finding a $(1+\varepsilon)$-approximation of a center point for a set of $\leq m$ points, where the objective is to minimize the sum of the individual distances to the center point raised to the p^{th} power. Note that for $p = \infty$, the problem reduces to finding a minimum enclosing ball, and for $p = 2$, it reduces to finding the center of gravity of the set of discrete points. Both problems can be solved exactly. We show (Lemma 3.4.5) that for all dtw_p, there is a deterministic 2-approximation which is a crucial ingredient for our approximation algorithms of (k,ℓ)-median under p-DTW.

Definition 3.4.2. A traversal $((a_1,b_1),(a_2,b_2),\ldots,(a_l,b_l)) \in \mathcal{T}_{m,\ell}$ is said to be ***lopsided*** (refer to Figure 3.3) if for all i we have that $(a_{i+1},b_{i+1}) - (a_i,b_i) \in \{(0,1),(1,1)\}$.

Lemma 3.4.3. *Let $\sigma \in \mathbb{X}^{d,m}$ and $\tau \in \mathbb{X}^{d,\ell}$. There is an optimal traversal $T^* \in \mathcal{T}_{m,\ell}$ that realizes $\mathrm{dtw}(\sigma,\tau)$ and for all $1 \leq i \leq |T^*| - 2$ it holds that $(a_{i+2},b_{i+2}) - (a_i,b_i) \neq (1,1)$.*

Proof. Let T be an arbitrary traversal for which there is an index i such that $(a_{i+2},b_{i+2}) - (a_i,b_i) = (1,1)$. Observe that we can remove the index pair (a_{i+1},b_{i+1}) from the traversal without increasing the distance, which implies the claim. □

Lemma 3.4.4. *Let $\tau \in \mathbb{X}^{d,m}$. Then there is a curve $\sigma \in \mathbb{X}^{d,\ell}$ with*

$$\mathrm{dtw}_p(\sigma,\tau) = \inf_{\sigma' \in \mathbb{X}^{d,\ell}} \mathrm{dtw}_p(\sigma',\tau).$$

such that $\mathrm{dtw}_p(\sigma,\tau)$ can be realized with a lopsided traversal.

Proof. Refer to Figure 3.4. Assume that $\mathrm{dtw}_p(\sigma,\tau)$ is not attained by a lopsided traversal. It suffices to show that there is then a curve $\sigma' \in \mathbb{X}^{d,\ell}$ with $\mathrm{dtw}_p(\sigma',\tau) \leq \mathrm{dtw}_p(\sigma,\tau)$.

Hence, fix some (m,ℓ)-traversal that is not lopsided, i.e., there is an index i such that $(a_{i+1},b_{i+1}) - (a_i,b_i) = (1,0)$. Observe that removing vertex $\sigma_{a_{i+1}}$ from σ does not increase $\mathrm{dtw}_p(\sigma,\tau)$. Indeed, by Lemma 3.4.3, we can assume that $(a_{i+2},b_{i+2}) - (a_{i+1},b_{i+1}) \neq (0,1)$, so $a_{i+2} = a_{i+1} + 1$. Therefore, discarding $\sigma_{a_{i+1}}$ from σ removes exactly one summand from the sum in $\mathrm{dtw}_p(\sigma,\tau)$, concluding the proof. □

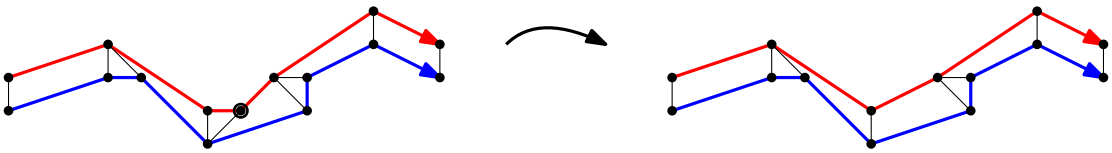


Figure 3.4: Illustration of Proof of Lemma 3.4.4. The vertex that can safely be deleted is marked on the left, and removed on the right.

Lemma 3.4.5. *For $\sigma = (\sigma_1, \ldots, \sigma_m) \in \mathbb{X}^{d,m}$ and integer $\ell > 0$, one can compute in $\mathcal{O}(m^2(d+\ell+m))$ time a curve $\sigma^* \in \mathbb{X}^{d,\ell}$ such that*

$$\inf_{\sigma_\ell \in \mathbb{X}^{d,\ell}} \mathrm{dtw}_p(\sigma_\ell, \sigma) \leq \mathrm{dtw}_p(\sigma^*, \sigma) \leq 2 \inf_{\sigma_\ell \in \mathbb{X}^{d,\ell}} \mathrm{dtw}_p(\sigma_\ell, \sigma).$$

Proof. First compute in $\mathcal{O}(m^2 d)$ time all pairwise distances of vertices of σ, and store them for later use. Next, compute $C(a,b,i) = \sum_{a \leq j \leq b} \|\sigma_i - \sigma_j\|^p$ for all $1 \leq a \leq i \leq b \leq m$ in total time $\mathcal{O}(m^3)$ via the recurrence $C(a,a,a) = 0$, $C(a-1,b,i) = C(a,b,i) + \|\sigma_i - \sigma_{a-1}\|^p$ and $C(a,b+1,i) = C(a,b,i) + \|\sigma_i - \sigma_{b+1}\|^p$. With $C(a,b,i)$ at hand for every $1 \leq a \leq i \leq b \leq m$, we compute $C(a,b) = \min_{a \leq i \leq b} \sum_{a \leq j \leq b} \|\sigma_i - \sigma_j\|^p$ for all $1 \leq a \leq b \leq m$, again in total time $\mathcal{O}(m^3)$.

Consider now for $1 \leq a \leq b \leq m$,

$$D(a,b) = \min_{\substack{\sigma' = (\sigma_1, \ldots, \sigma_b) \\ S = (s_1, \ldots, s_a) \text{ subsequence of } \sigma' \\ s_a = \sigma_b}} \sum_{i=1}^{a} C(s_{i-1}+1, s_i),$$

with $s_0 := 0$, which corresponds to the optimal partitioning of the first b indices into a contiguous disjoint intervals under the cost function C. Computing the value $D(\ell, m)$ takes $\mathcal{O}(m^2\ell)$ time via the recurrence $D(a,b) = \min_{b' \leq b} D(a-1,b') + C(b'+1,b)$. The subsequence realizing $D(\ell,m)$ defines σ^*. For correctness, observe that the optimal simplification also yields a partition of the vertices of σ into ℓ contiguous disjoint intervals. Let T be the partition computed, and let $T^{\mathrm{opt}} = ([a_1,b_1], \ldots, [a_\ell, b_\ell])$ be an optimal partition of $[m]$ that realizes $\inf_{\sigma_\ell \in \mathbb{X}^{d,\ell}} \mathrm{dtw}_p(\sigma_\ell, \sigma)$. For a vertex v_i in a fixed optimal simplification, let π_i be a closest point among $\{\sigma_{a_i}, \ldots, \sigma_{b_i}\}$. Then

$$\begin{aligned}
\left(\sum_{[a,b] \in T} C(a,b) \right)^{1/p} &= \left(\sum_{[a,b] \in T} \min_{a \leq i \leq b} \sum_{a \leq j \leq b} \|\sigma_i - \sigma_j\|_2^p \right)^{1/p} \\
&\leq \left(\sum_{[a,b] \in T^{\mathrm{opt}}} \min_{a \leq i \leq b} \sum_{a \leq j \leq b} \|\sigma_i - \sigma_j\|_2^p \right)^{1/p} \\
&\leq \left(\sum_{[a,b] \in T^{\mathrm{opt}}} \sum_{a \leq j \leq b} \|\pi_i - \sigma_j\|_2^p \right)^{1/p} \\
&\leq \left(\sum_{[a,b] \in T^{\mathrm{opt}}} \sum_{a \leq j \leq b} (\|\pi_i - v_i\|_2 + \|v_i - \sigma_j\|_2)^p \right)^{1/p}
\end{aligned}$$

$$\leq \left(\sum_{[a,b]\in T^{\text{opt}}} \sum_{a\leq j\leq b} (2\|v_i - \sigma_j\|_2)^p \right)^{1/p}$$
$$\leq 2 \left(\sum_{[a,b]\in T^{\text{opt}}} \sum_{a\leq j\leq b} \|v_i - \sigma_j\|_2^p \right)^{1/p} = 2 \inf_{\sigma_\ell \in \mathbb{X}^{d,\ell}} \text{dtw}_p(\sigma_\ell, \sigma). \quad \square$$

3.4.2 Dynamic Time Warping Approximating Metric

We begin with the following more general triangle inequality for dtw_p, which motivates analyzing the metric closure of the input set. While dtw_p does not satisfy the triangle inequality (see Figure 3.2), the inequality shows it is never 'too far off'. Remarkably, the inequality does not depend on the complexity of the curves 'visited'.

Lemma 3.4.6 (Iterated triangle inequality)**.** *Let $s \in \mathbb{X}^{d,\ell}$, $t \in \mathbb{X}^{d,\ell'}$ and $X = (x_1, \ldots, x_r)$ be an arbitrary ordered set of curves in $\mathbb{X}^{d,m}$. Then*

$$\text{dtw}_p(s,t) \leq (\ell + \ell')^{1/p} \left(\text{dtw}_p(s, x_1) + \sum_{i<r} \text{dtw}_p(x_i, x_{i+1}) + \text{dtw}_p(x_r, t) \right).$$

Proof. To ease exposition, assume that $r = 2$, that is, $X = (x, y)$. Let W_{sx} be an optimal traversal of s and x realizing $\text{dtw}_p(s, x)$. Similarly define W_{xy} and W_{yt}. From this we now construct a traversal of s and t endowed with additional information on which vertices of x and y were used to match the vertices of s and t. More precisely, we will construct an ordered set W of indices $((\alpha_1, \beta_1, \gamma_1, \delta_1), (\alpha_2, \beta_2, \gamma_2, \delta_2), \ldots)$, such that for any $i \geq 2$ it holds that $(\alpha_i, \delta_i) - (\alpha_{i-1}, \delta_{i-1}) \in \{(0,1), (1,0), (1,1)\}$, and for any $i \geq 1$ it holds that $(\alpha_i, \beta_i) \in W_{sx}$, $(\beta_i, \gamma_i) \in W_{xy}$, and $(\gamma_i, \delta_i) \in W_{yt}$. Refer to Figure 3.5 for a schematic view of the constructed set W.

We begin with $W = ((1,1,1,1))$, which clearly has the stated properties as $(1,1)$ is in any traversal. Now recursively define the next element in W based on the last element $(\alpha, \beta, \gamma, \delta)$ of W.

If $\alpha = \ell$ and $\delta = \ell'$, we stop adding elements to W. Otherwise, if $\delta < \ell'$ let $\delta' = \delta + 1$. From this let $\gamma' = \min\{j \geq \gamma \mid (j, \delta') \in W_{yt}\}$, which exists, because W_{yt} itself is a traversal. Similarly from γ' define β' and from β' define α'. If $\alpha' \leq \alpha + 1$, then $(\alpha', \beta', \gamma', \delta')$ is added to W, and the steps are recursively repeated. Observe that $\alpha' \geq \alpha$, which implies that the properties of W are preserved. If instead $\alpha' > \alpha + 1$, let $\alpha'' = \alpha + 1$. From this define $\beta'' = \min\{b \geq \beta \mid (\alpha'', b) \in W_{sx}\}$. Clearly $\beta'' < \beta'$, as $\alpha'' < \alpha'$. Similarly from this define γ'' for which it holds that $\gamma'' < \gamma'$ and from this define δ'' for which it holds that $\delta'' < \delta' = \delta + 1$. But by definition $\delta \leq \delta''$, thus $\delta = \delta''$. Thus we add $(\alpha'', \beta'', \gamma'', \delta'')$ preserving the properties of W. From here recursively repeat the steps above.
In the case where $\delta = \ell'$, we set $\alpha' = \alpha + 1$. From this we similarly define $\beta' = \min\{b \geq \beta \mid (\alpha', b) \in W_{sx}\}$, from which we similarly define γ', from which we define δ'. But as $\ell' = \delta \leq \delta' \leq \ell'$ it follows that $\delta' = \ell'$ and thus adding $(\alpha', \beta', \gamma', \delta')$ to W also preserves its properties. From here recursively repeat the steps above.

Observe now that

$$\left(\sum_{(\alpha,\beta,\gamma,\delta)\in W} \|s_\alpha - x_\beta\|_2^p \right)^{1/p} \leq \left(\sum_{(\alpha,\beta)\in W_{sx}} |W| \cdot \|s_\alpha - x_\beta\|_2^p \right)^{1/p} = |W|^{1/p} \, \text{dtw}(s,x)_p.$$

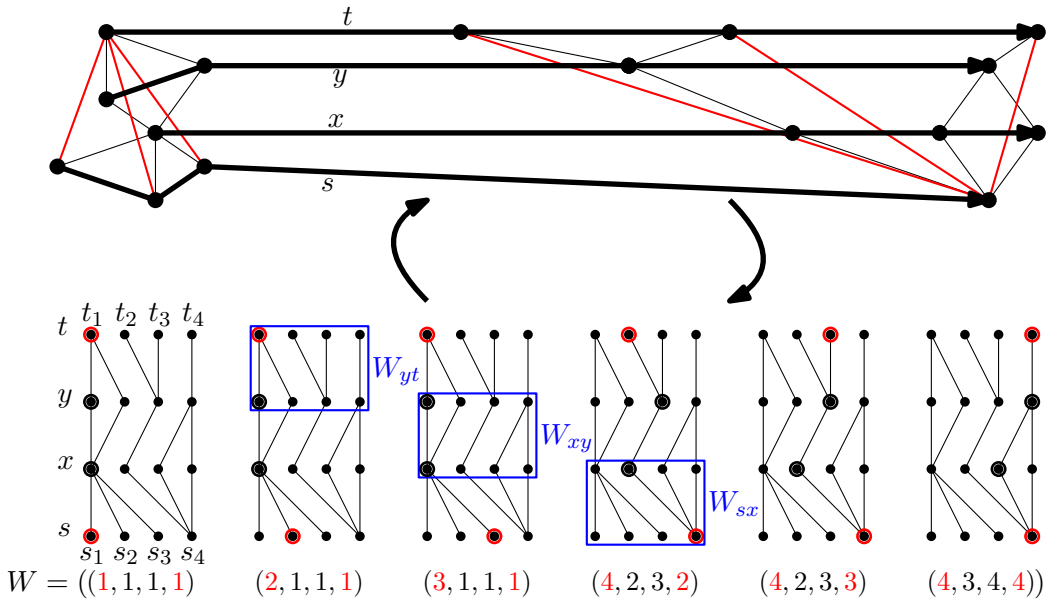


Figure 3.5: Illustration of how the optimal traversals W_{sx}, W_{xy} and W_{yt} of visited curves can be 'composed' to yield a set W that induces a traversal $\widetilde{W}$ (in red) of s and t. Any single matched pair of vertices in W_{sx}, W_{xy} or W_{yt} is at most $|W| \leq \ell + \ell'$ times a part of W.

This similarly holds for x and y, and y and t. Further, we acquire a traversal $\widetilde{W} = ((\alpha_1, \delta_1), \ldots)$ of s and t from W by dropping the middle two indices of each element of W. Now overall

$$\begin{aligned}
\operatorname{dtw}_p(s,t) &\leq \left(\sum_{(\alpha,\delta)\in\widetilde{W}} \|s_\alpha - t_\delta\|_2^p \right)^{1/p} = \left(\sum_{(\alpha,\beta,\gamma,\delta)\in W} \|s_\alpha - t_\delta\|_2^p \right)^{1/p} \\
&\leq \left(\sum_{(\alpha,\beta,\gamma,\delta)\in W} (\|s_\alpha - x_\beta\|_2 + \|x_\beta - y_\gamma\|_2 + \|y_\gamma - t_\delta\|_2)^p \right)^{1/p} \\
&\leq |W|^{1/p} \operatorname{dtw}(s,x)_p + |W|^{1/p} \operatorname{dtw}(x,y)_p + |W|^{1/p} \operatorname{dtw}(y,t)_p,
\end{aligned}$$

which concludes the proof, as $|W| = |\widetilde{W}| \leq \ell + \ell'$.

Observe that this analysis immediately extends to $r > 2$. In this case W consists of tuples of length $r + 2$, while $|W| = |\widetilde{W}| \leq \ell + \ell'$ still holds. □

Definition 3.4.7 (metric closure)**.** Let (X, ϕ) be a finite set endowed with a distance function $\phi : X \times X \to \mathbb{R}_{\geq 0}$. The metric closure $\overline{\phi}$ of ϕ (refer to Figure 3.6) is the function

$$\overline{\phi} : X \times X \to \mathbb{R}, (s,t) \mapsto \min_{\substack{r\geq 2, \{\tau_1,\ldots,\tau_r\}\subset X \\ s=\tau_1, t=\tau_r}} \sum_{i<r} \phi(\tau_i, \tau_{i+1}).$$

The metric closure of a distance function is a semi-metric and can be extended to a metric by removing duplicates or small perturbations.

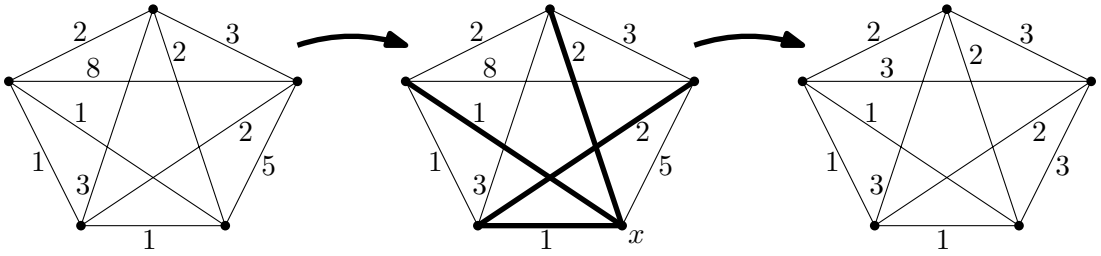


Figure 3.6: Illustration of the metric closure. On the left a distance function on five points represented as a graph. In the middle the shortest path tree rooted at x inducing all values of the metric closure of the distance function from some element to x. On the right the metric closure.

Observation 3.4.8. *Let X be a finite set with distance function ϕ. Let $Y \subset X$. Then for any $\sigma, \tau \in Y$ it holds that $\overline{\phi}(\sigma, \tau) \leq \overline{\phi|_Y}(\sigma, \tau) \leq \phi(\sigma, \tau)$.*

By Lemma 3.4.6 and Observation 3.4.8, dtw_p on any finite set of curves in $\mathbb{X}^{d,m}$ is approximated by its metric closure, with approximation constant depending on m.

Lemma 3.4.9. *For any set of curves X and two curves $\sigma, \tau \in X$ of complexity at most m it holds that $\mathrm{dtw}_p(\sigma, \tau) \leq (2m)^{1/p}\overline{\mathrm{dtw}_p|_X}(\sigma, \tau) \leq (2m)^{1/p}\,\mathrm{dtw}_p(\sigma, \tau)$.*

3.4.3 Cubic Time Algorithm

We now give a bicriteria approximation algorithm for (k, ℓ)-median under dtw_p with cubic (in n) running time. This algorithm will later serve as a basis for a bicriteria approximation algorithm with linear dependence in n.

Lemma 3.4.10. *Let $X \subset \mathbb{X}^{d,m}$ be a set of n curves and k and ℓ be given. Let $X^* = \{\tau^* \mid \tau \in X\}$, where τ^* is a $(1+\varepsilon)$-approximate ℓ-simplification of τ. Let $C \subset X^*$ be an (α, β)-approximation of the k-median problem of X^* in the metric space $(X^*, \overline{\mathrm{dtw}_p|_{X^*}})$. Then C is a $\left((4m\ell)^{1/p}\left((4+2\varepsilon)\alpha + 1 + \varepsilon\right), \beta\right)$-approximation of the (k, ℓ)-median problem on X.*

Proof. For any curve $\tau^* \in X^*$ let $\overline{c}(\tau^*)$ be the closest element among C under the metric $\overline{\mathrm{dtw}_p|_{X^*}}$ and for any curve τ let $c(\tau)$ be the closest element among C under dtw_p, and let $\Delta = \sum_{\tau \in X} \mathrm{dtw}_p(\tau, c(\tau))$ be the cost of C. Let $C^{\mathrm{opt}} = \{c_1^{\mathrm{opt}}, \ldots, c_k^{\mathrm{opt}}\}$ be an optimal solution to the (k, ℓ)-median problem on X with cost Δ^*.

Let $V_i = \{\tau \in X \mid \forall j : \mathrm{dtw}_p(\tau, c_i^{\mathrm{opt}}) \leq \mathrm{dtw}_p(\tau, c_j^{\mathrm{opt}})\}$ be the Voronoi cell of c_i^{opt}, which we assume partitions X by breaking ties arbitrarily. We can assume that all sets V_i are non-empty by removing the elements of C^{opt} with empty V_i. For any i, fix the closest $\pi_i \in V_i$ to c_i^{opt} under dtw_p. Letting $\Delta_i^* = \sum_{\sigma \in V_i} \mathrm{dtw}(c_i^{\mathrm{opt}}, \sigma)$, we have $\Delta^* = \sum_{i \leq k} \Delta_i^*$. Let $\mathfrak{X} = X \cup X^* \cup C^{\mathrm{opt}}$. By Observation 3.4.8, for any $i \leq k$ and $\tau \in V_i$, it holds that

$$\begin{aligned}
\overline{\mathrm{dtw}_p|_{\mathfrak{X}}}(c_i^{\mathrm{opt}}, \pi_i^*) &\leq \overline{\mathrm{dtw}_p|_{\mathfrak{X}}}(c_i^{\mathrm{opt}}, \pi_i) + \overline{\mathrm{dtw}_p|_{\mathfrak{X}}}(\pi_i, \pi_i^*) \\
&\leq \mathrm{dtw}_p(c_i^{\mathrm{opt}}, \pi_i) + \mathrm{dtw}_p(\pi_i, \pi_i^*) \\
&\leq \mathrm{dtw}_p(c_i^{\mathrm{opt}}, \pi_i) + (1+\varepsilon)\,\mathrm{dtw}_p(\pi_i, c_i^{\mathrm{opt}}) \\
&\leq (2+\varepsilon)\,\mathrm{dtw}_p(c_i^{\mathrm{opt}}, \tau),
\end{aligned}$$

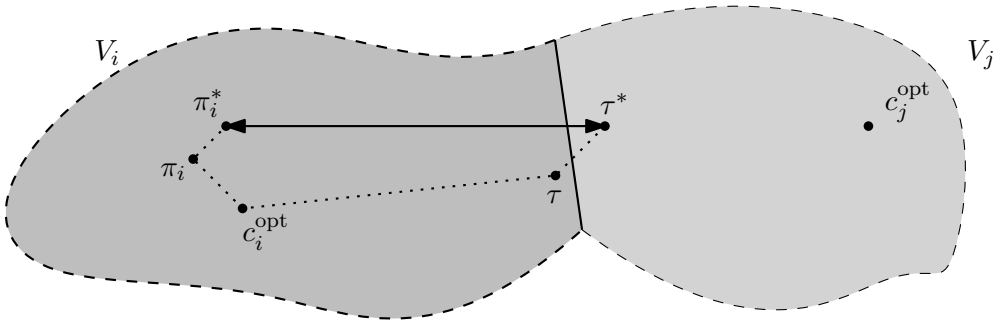


Figure 3.7: Illustration to Proof of Lemma 3.4.10: Assigning τ^* (the $(1+\varepsilon)$-simplification of τ which lies inside the Voronoi cell V_i of c_i^{opt}) to π_i^* (the $(1+\varepsilon)$-simplification of the closest element π_i in V_i to c_i^{opt}) under $\overline{\text{dtw}_p\,|_{X^*}}$ is at most $4+2\varepsilon$ times as bad as assigning τ to c_i^{opt} under dtw_p.

and further observe that for

$$\begin{aligned}\overline{\text{dtw}_p\,|_{\mathfrak{X}}}(\tau^*, c_i^{\text{opt}}) &\leq \overline{\text{dtw}_p\,|_{\mathfrak{X}}}(\tau^*, \tau) + \overline{\text{dtw}_p\,|_{\mathfrak{X}}}(\tau, c_i^{\text{opt}}) \\ &\leq \text{dtw}_p(\tau^*, \tau) + \text{dtw}_p(\tau, c_i^{\text{opt}}) \\ &\leq (1+\varepsilon)\,\text{dtw}_p(c_i^{\text{opt}}, \tau) + \text{dtw}_p(\tau, c_i^{\text{opt}}) \\ &\leq (2+\varepsilon)\,\text{dtw}_p(c_i^{\text{opt}}, \tau).\end{aligned}$$

And thus it holds that $\sum_i \sum_{\tau \in V_i} \overline{\text{dtw}_p\,|_X}(\tau^*, \pi_i^*) \leq (4+2\varepsilon)\Delta^*$ (refer to Figure 3.7). In conjunction with Lemma 3.4.6, Observation 3.4.8, and Lemma 3.4.9 this yields

$$\begin{aligned}\Delta &= \sum_{\tau \in X} \text{dtw}_p(\tau, c(\tau)) \leq \sum_{\tau \in X} \text{dtw}_p(\tau, \overline{c}(\tau^*)) \\ &\leq (2m)^{1/p} \sum_{\tau \in X} \overline{\text{dtw}_p\,|_{\mathfrak{X}}}(\tau, \overline{c}(\tau^*)) \\ &\leq (2m)^{1/p} \sum_{\tau \in X} \left(\overline{\text{dtw}_p\,|_{\mathfrak{X}}}(\tau, \tau^*) + \overline{\text{dtw}_p\,|_{\mathfrak{X}}}(\tau^*, \overline{c}(\tau^*)) \right) \\ &\leq (2m)^{1/p} \left(\sum_{\tau \in X} \text{dtw}_p(\tau, \tau^*) + \sum_{\tau \in X} \overline{\text{dtw}_p\,|_{X^*}}(\tau^*, \overline{c}(\tau^*)) \right) \\ &\leq (2m)^{1/p} \left(\sum_i \sum_{\tau \in V_i} \text{dtw}_p(\tau, \tau^*) + \alpha \sum_i \sum_{\tau \in V_i} \overline{\text{dtw}_p\,|_{X^*}}(\tau^*, \pi_i^*) \right) \\ &\leq (2m)^{1/p} \left(\sum_i \sum_{\tau \in V_i} \text{dtw}_p(\tau, \tau^*) + \alpha \sum_i \sum_{\tau \in V_i} \text{dtw}_p(\tau^*, \pi_i^*) \right) \\ &\leq (2m)^{1/p} \left(\sum_i \sum_{\tau \in V_i} (1+\varepsilon)\,\text{dtw}_p(\tau, c_i^{\text{opt}}) + \alpha(2\ell)^{1/p} \sum_i \sum_{\tau \in V_i} \overline{\text{dtw}_p\,|_{\mathfrak{X}}}(\tau^*, \pi_i^*) \right) \\ &\leq (2m)^{1/p} \left((1+\varepsilon)\Delta^* + \alpha(2\ell)^{1/p}(4+2\varepsilon)\Delta^* \right) \leq (4m\ell)^{1/p}((4+2\varepsilon)\alpha + 1 + \varepsilon)\Delta^*. \square\end{aligned}$$

Lemma 3.4.11. *Let $X \subset \mathbb{X}^{d,\ell}$ be a set of n curves. The metric closure $\overline{\mathrm{dtw}_p|_X}$ for all pairs of curves in X can be computed in $\mathcal{O}(n^2\ell^2 d + n^3)$ time.*

Proof. First compute the value $\mathrm{dtw}_p(\sigma, \tau)$ for all pairs of curves $\sigma, \tau \in X$. This takes $\mathcal{O}(n^2\ell^2)$ time. Using these values, we define the complete graph $\mathcal{G}(X) = (X, \binom{X}{2})$ on X, where edge weights correspond to the computed distances. The metric closure of dtw_p corresponds to the weights of a shortest path in $\mathcal{G}(X)$. All these $\binom{n}{2}$ values can be computed in $\mathcal{O}(n^3)$ time by n applications of Dijkstra's algorithm. □

Theorem 3.4.12 ([Che09])**.** *Given a set P of n points in a metric space, for $\varepsilon \in (0, 1]$, one can compute a $(10+\varepsilon)$-approximate k-median clustering of P in $\mathcal{O}(nk+k^7\varepsilon^{-5}\log^5 n)$ time, with constant probability of success.*

Theorem 3.4.13. *Let X be a set of curves of complexity at most m. Let k and ℓ be given. Let $X^* = \{\tau^* \mid \tau \in X\}$ be a set of $(1+\varepsilon)$-approximate optimal ℓ-simplifications. There is an algorithm with input X^* which computes a $(10+\varepsilon, 1)$-approximation to the k-median problem of X^* in $(X^*, \overline{\mathrm{dtw}_p|_{X^*}})$ in $\mathcal{O}(n^2\ell^2 d + n^3 + nk + k^7\varepsilon^{-5}\log^5 n)$ time.*

Proof. This is a direct consequence of Lemma 3.4.11 and Theorem 3.4.12. □

Corollary 3.4.14. *Let X be a set of curves of complexity at most m. Let k and ℓ be given. There is an algorithm which computes a $((4m\ell)^{1/p}(62 + \mathcal{O}(\varepsilon)), 1)$-approximation to the (k, ℓ)-median problem on X under dtw_p in $\mathcal{O}(nm^3 + n^2\ell^2 + n^3 + nk + k^7\varepsilon^{-5}\log^5 n)$.*

Proof. This is a direct consequence of Theorem 3.4.13, Lemma 3.4.10 and Lemma 3.4.5. □

We next show how to combine our ideas with a sampling technique for bicriteria k-median approximations [Ind99] to achieve linear dependence on n.

3.4.4 Linear Time Algorithm

With Theorem 3.4.13 (and by extension also Corollary 3.4.14), we have ran into the following predicament: We would like to apply linear-time algorithms to the metric closure of dtw_p. However, constructing the metric closure takes cubic time. We circumvent this by applying a technique from [Ind99], which reduces a k-median instance with n points to two k-median instances with $\mathcal{O}(\sqrt{n})$ points via sampling. More precisely, we will apply this technique twice, so that we will compute the metric closure only on sampled subsets of size $\mathcal{O}(n^{1/4})$.

In this section we want to analyze the problem of computing a k-median of a set X in the metric space $(X, \overline{\phi})$, where ϕ is a distance function on X with the guarantee that there is a constant ζ such that for any $x, y \in X$ it holds that $\phi(x, y) \leq \zeta\overline{\phi}(x, y)$, with a linear running time, and more precisely, only a linear number of calls to the distance function ϕ, and no calls to $\overline{\phi}$. By Lemma 3.4.9, the results in this section translate directly to $\phi = \mathrm{dtw}_p|_X$ with $\zeta = (m + \ell)^{1/p}$.

Observe that similar to Theorem 3.4.13, the following lemma holds.

Lemma 3.4.15. *Let X be a set of n points, equipped with a distance function ϕ that can be computed in time T_ϕ. There is a $(10+\varepsilon, 1)$-approximate algorithm for k-median of X in $(X, \overline{\phi})$ that has constant probability of success and has running time $\mathcal{O}(n^2 T_\phi + n^3 + nk + k^7\varepsilon^{-5}\log^5 n)$.*

Algorithm 1 k-median framework

procedure k-ROUTINE$((X,\phi),\varepsilon,\mathcal{A})$
  $a \leftarrow \Theta(\varepsilon^{-1}\sqrt{\log(\varepsilon^{-1})})$, $b \leftarrow \Theta(a^2)$
  $s \leftarrow a\sqrt{kn\log k}$
  Choose a set S of s points sampled without replacement from X
  $C' \leftarrow \mathcal{A}((S,\phi))$
  Select the set M of points x with the $b\frac{kn\log k}{s}$ largest values of $\min_{c'\in C'}\overline{\phi}(x,c')$
  return $C = C' \cup \mathcal{A}((M,\phi))$

procedure k-MEDIAN$((X,\phi),\varepsilon,\mathcal{A})$
  $a \leftarrow \Theta(\varepsilon^{-1}\sqrt{\log(\varepsilon^{-1})})$, $b \leftarrow \Theta(a^2)$
  $s \leftarrow a\sqrt{kn\log k}$
  Choose a set S of s points sampled without replacement from X
  $C' \leftarrow k$-ROUTINE$((S,\phi|_S),\varepsilon,\mathcal{A})$
  Select the set M of points x with the $b\frac{kn\log k}{s}$ largest values of $\min_{c'\in C'}\phi(x,c')$
  return $C = C' \cup k$-ROUTINE$((M,\phi|_M),\varepsilon,\mathcal{A})$

Lemma 3.4.16. *Let X be a set of n points, equipped with a distance function ϕ, such that $\phi \le \zeta\overline{\phi}$ for some $\zeta > 0$, and $Y \subset X$. An (α,β)-approximation for the k-median problem for Y in $(Y,\overline{\phi|_Y})$ is an $(\alpha\zeta,\beta)$-approximation for the k-median problem for Y in $(Y,\overline{\phi}|_Y)$.*

Proof. Let $C \subset Y$ be an (α,β)-approximation for the k-median problem for Y in $(Y,\overline{\phi|_Y})$, and let $C^{\text{opt}} = \{c_1^{\text{opt}},\dots,c_k^{\text{opt}}\} \subset Y$ be an optimal solution for the k-median problem for Y in $(Y,\overline{\phi}|_Y)$ with cost Δ^{opt}. For any $\tau \in Y$ let $c_Y(\tau)$ be the closest element among C under $\overline{\phi|_Y}$, and let $c_X^{\text{opt}}(\tau)$ be the closest element among C^{opt} under $\overline{\phi} = \overline{\phi}|_Y$. Then $\Delta^{\text{opt}} = \sum_{\tau\in Y}\overline{\phi}(\tau, c_X^{\text{opt}}(\tau))$. Overall by Observation 3.4.8 we see that

$$\begin{aligned}\sum_{\tau\in Y}\overline{\phi}(\tau,c_Y(\tau)) &\le \sum_{\tau\in Y}\overline{\phi|_Y}(\tau,c_Y(\tau)) \le \alpha\sum_{\tau\in Y}\overline{\phi|_Y}(\tau,c_X(\tau))\\ &\le \alpha\sum_{\tau\in Y}\phi(\tau,c_X(\tau)) \le \alpha\sum_{\tau\in Y}\zeta\overline{\phi}(\tau,c_X(\tau)) = \zeta\alpha\Delta^{\text{opt}}.\end{aligned}$$

□

Theorem 3.4.17 ([Ind99, Theorem 1])**.** *Let X be a set of n points, and let ϕ be a distance function on X. Let $\mathcal{A}$ be an algorithm that provided with S and ϕ computes an (α,β)-approximation for k-median of S w.r.t. $\overline{\phi}|_S$. Then for any $\varepsilon > 0$ the k-ROUTINE in Algorithm 1 provided with $\mathcal{A}$ computes a $(3(1+\varepsilon)(2+\alpha),2\beta)$-approximation for k-median in the metric space $(X,\overline{\phi})$ with constant success probability.*

Lemma 3.4.18. *Let X be a set of n points, and let ϕ be a distance function that can be computed in T_ϕ time for any $x,y \in X$. Let $T_{\mathcal{A}}(n)$ be the algorithm $\mathcal{A}$ that provided with S and ϕ computes an (α,β)-approximation for k-median of S w.r.t. $\overline{\phi}|_S$. Then k-ROUTINE has a running time of $\mathcal{O}(n^2T_\phi + T_{\mathcal{A}}(\min(n,\varepsilon^{-1}\sqrt{kn\log(k)\log(\varepsilon^{-1})})))$.*

Proof. The only steps that take time outside the two calls to $\mathcal{A}$ are sampling S and constructing M. Computing the values $\min_{c'\in C'}\overline{\phi}(x,c')$ for all $x \in X$ can be done in a single execution of Dijkstra's algorithm, starting with the points of $C' \subset X$ at distance 0, by adding a temporary point with distance 0 to all points in C' and starting Dijkstra's

algorithm on this temporary point. This takes $\mathcal{O}(n^2 T_\phi)$ time, which also dominates the time it takes to sample S as well as constructing M from these computed values. □

Lemma 3.4.19. *Let X be a set of n points, and let ϕ be a distance function on X, which can be computed in time T_ϕ, and further there is a constant ζ such that $\phi \leq \zeta\overline{\phi}$. Let $Y \subset X$. Let $\varepsilon > 0$ and let $\mathcal{A}$ be the $(10+\varepsilon, 1)$-approximation for metric k-median of Lemma 3.4.15 which provided with S and ϕ computes a $(10+\varepsilon, 1)$-approximation for metric k-median w.r.t $\overline{\phi|_S}$. Then k-ROUTINE returns a $(3(1+\varepsilon)\zeta(12+\varepsilon), 2)$-approximation of k-median in the metric space $(Y, \overline{\phi}|_Y)$ in time $\mathcal{O}(|Y|^2 T_\phi + |Y|^2 k \log(k)\varepsilon^{-2}\log(\varepsilon^{-1}) + k^7\varepsilon^{-5}\log^5(|Y|))$.*

Proof. The running time bound follows by Lemma 3.4.18 and Lemma 3.4.15, together with the fact that $\min(|Y|, \varepsilon^{-1}\sqrt{k|Y|\log(k)\log(\varepsilon^{-1})})^3 \leq |Y|^2 k\log(k)\varepsilon^{-2}\log(\varepsilon^{-1})$. The approximation guarantee follows by Theorem 3.4.17, Lemma 3.4.16, and Lemma 3.4.15. □

We obtain our two main results of the section. The first is Theorem 3.4.21, which provides a linear-time approximation algorithm for k-median in metric closures, assuming the underlying distance is reasonably well approximated by its metric closure. The second is Corollary 3.4.22, combining Theorem 3.4.21 with Lemma 3.4.10 to yield an approximation algorithm for p-DTW with an unoptimized approximation guarantee.

Lemma 3.4.20 ([Ind99, Proof of Theorem 1])**.** *Assume that C', computed in the algorithm k-MEDIAN, is an (α, β)-approximation of k-median of S in the metric space $(S, \overline{\phi}|_S)$. With constant probability depending only on a and b, there is a subset of X of size at least $n - b\frac{kn}{s}\log k$, whose cost under $\overline{\phi}$ with C' as medians is at most $(1+\varepsilon)(2+\alpha)\Delta^{\mathrm{opt}}$, where Δ^{opt} is the cost of an optimal k-median under $\overline{\phi}$ of X.*

Theorem 3.4.21. *Let X be a set of points and let ϕ be a distance function on X with $\phi \leq \zeta\overline{\phi}$. Let $\varepsilon > 0$ and let $\mathcal{A}$ be the $(10+\varepsilon, 1)$-approximation for metric k-median of Theorem 3.4.13. Then k-MEDIAN returns a $(11\zeta^2(1+\varepsilon)^2(12+\varepsilon), 4)$-approximation of k-median of X in the metric space $(X, \overline{\phi})$ in time $\mathcal{O}(nk\log(k)T_\phi + nk^2\log^2 k + k^7\varepsilon^{-5}\log^5(n))$.*

Proof. Let Δ^{opt} be the cost of an optimal solution to the k-median problem on X under ϕ. By Lemma 3.4.19, the set C' is a $(3(1+\varepsilon)\zeta(12+\varepsilon), 2)$-approximation of k median of S in $(S, \overline{\phi}|_S)$. This set can be computed in time $\mathcal{O}(nk\log(k)T_\phi + nk^2\log^2(k)\varepsilon^{-4}\log^2(\varepsilon^{-1}) + k^7\varepsilon^{-5}\log^5(n))$. The set M can be computed in $\mathcal{O}(knT_\phi)$ time. By Lemma 3.4.19, the set C'' is a $(3(1+\varepsilon)\zeta(12+\varepsilon), 2)$-approximation of k median of M in $(M, \overline{\phi}|_M)$ and can be computed in time $\mathcal{O}(nk\log(k)T_\phi + nk^2\log^2(k)\varepsilon^{-4}\log^2(\varepsilon^{-1}) + k^7\varepsilon^{-5}\log^5(nk))$.

Now observe that by the choice of M it holds that $\sum_{x\in X\setminus M}\phi(x, C') \leq \sum_{x\in O}\phi(x, C')$ for the set O of Lemma 3.4.20, and hence

$$\begin{aligned}\sum_{x\in X\setminus M}\overline{\phi}(x, C') &\leq \sum_{x\in X\setminus M}\phi(x, C') \leq \sum_{x\in O}\phi(x, C') \\ &\leq \sum_{x\in O}\zeta\overline{\phi}(x, C') \leq \zeta(1+\varepsilon)(2+3(1+\varepsilon)\zeta(12+\varepsilon))\Delta^{\mathrm{opt}}.\end{aligned}$$

Further observe that by replacing every point of the optimum on X with its closest point in M there exists a set of points $C_M \subset M$ of size k with $\sum_{x\in M}\overline{\phi}|_M(x, C_M) \leq 2\Delta^{\mathrm{opt}}$.

Thus,

$$\sum_{x\in M}\overline{\phi}(x,C'') = \sum_{x\in M}\overline{\phi}|_M(x,C'') \leq 6(1+\varepsilon)\zeta(12+\varepsilon)\Delta^{\mathrm{opt}}.$$

Overall we get a $(11\zeta^2(1+\varepsilon)^2(12+\varepsilon),4)$-approximation (as $\zeta \geq 1$ and $\varepsilon > 0$) in time $\mathcal{O}(nk\log(k)T_\phi + nk^2\log^2(k)\varepsilon^{-4}\log^2(\varepsilon^{-1}) + k^7\varepsilon^{-5}\log^5(n))$. □

Corollary 3.4.22. *For any $\varepsilon > 0$ the procedure k-MEDIAN from Algorithm 1 can be used to compute a $(72(1+\varepsilon)^2(12+\varepsilon)(16m\ell^3)^{1/p},4)$-approximation for (k,ℓ)-median for an input set X of n curves of complexity m under dtw_p in time*

$$\mathcal{O}(nm^3d + nk\log(k)\ell^2d + nk^2\log^2(k)\varepsilon^{-4}\log^2(\varepsilon^{-1}) + k^7\varepsilon^{-5}\log^5(n)).$$

Proof. Let $X^* = \{\tau^* \mid \tau \in X\}$ be a set of 2-approximate optimal ℓ-simplifications of X. By Lemma 3.4.5, X^* can be computed in $\mathcal{O}(nm^3d)$ time. We now apply Theorem 3.4.21 and Lemma 3.4.9 to obtain a $(12(2\ell)^{2/p}(1+\varepsilon)^2(12+\varepsilon),4)$-approximation of k-median of X^* in $(X^*,\overline{\mathrm{dtw}_p|_{X^*}})$ in $\mathcal{O}(nk\log(k)\ell^2d + nk^2\log^2(k)\varepsilon^{-4}\log^2(\varepsilon^{-1}) + k^7\varepsilon^{-5}\log^5(n))$ time. By Lemma 3.4.10, the computed set is a $(6(4m\ell)^{1/p}12(2\ell)^{2/p}(1+\varepsilon)^2(12+\varepsilon),4)$-approximation for (k,ℓ)-median for X under dtw_p. □

3.5 Putting It All Together

The theoretical derivations of the previous sections culminate in an approximation algorithm (Theorem 3.5.2) to (k,ℓ)-median that is particularly useful in the big data setting, where $n \gg m$. Our strategy is to first compute an efficient but not very accurate approximation (Corollary 3.4.22) of (k,ℓ)-median. Subsequently, we use the approximation to construct a coreset. The metric closure of the coreset is then used as the proxy set for metric approximation algorithms, where by virtue of the size reduction we can greatly reduce the running time of slower more accurate approximation algorithms, yielding a better approximation.

Theorem 3.5.1 ([AGK+04, Che09])**.** *Given a set X of n points in a metric space, one can compute a $(5+\varepsilon)$-approximate k-median clustering of X in $\mathcal{O}(\varepsilon^{-1}n^2k^3\log n)$ time. If P is a weighted point set, with total weight W, then the time required is in $\mathcal{O}(\varepsilon^{-1}n^2k^3\log W)$.*

Theorem 3.5.2. *Let $\varepsilon \in (0,1]$. The algorithm (k,ℓ)-MEDIAN in Algorithm 2 is a $((32+\varepsilon)(4m\ell)^{1/p},1)$-approximate algorithm of constant success probability for (k,ℓ)-median on curves under dtw_p with a running time of $\widetilde{\mathcal{O}}\Big(n(m^3d+k^2+k\ell^2d) + \varepsilon^{-6}d^3\ell^3k^7\sqrt[p]{m^6\ell^{12}}\Big)$, where $\widetilde{\mathcal{O}}$ hides polylogarithmic factors in n, m, ℓ, k and ε^{-1}.*

Proof. By Corollary 3.4.22, computing a $(\mathcal{O}(m^{1/p}\ell^{3/p}),4)$-approximation takes time in $\mathcal{O}(nm^3d + nk\log(k)\ell^2 + nk^2\log^2(k) + k^7\log^5(n))$ and has constant success probability. From this we can compute a ε'-coreset S by Theorem 3.3.7 of size

$$\mathcal{O}(\varepsilon^{-2}d\ell k^2(m^2\ell^4)^{1/p}\log^3(m\ell)\log^2(k)\log(\varepsilon^{-1})\log(n))$$

with constant success probability, and in time $\mathcal{O}(kn)$. Computing S^* takes $\mathcal{O}(nm^3)$ time. Computing the metric closure of S^* takes $\mathcal{O}(|S|^3)$ time and computing a $(5+\varepsilon')$-approximate solution to the k-median solution of S^* takes $\mathcal{O}(\varepsilon^{-1}|S|^2k^3\log|S|)$ time by

Algorithm 2 $((32+\varepsilon)(4m\ell)^{1/p},1)$-approximate (k,ℓ)-median

procedure (k,ℓ)-MEDIAN$(X \subset \mathbb{X}^{d,m}, p, \varepsilon)$
- $\varepsilon' \leftarrow \varepsilon/46$
- Compute $(\mathcal{O}((16m\ell^3)^{1/p}),4)$-approximation C' (Corollary 3.4.22)
- Compute bound of sensitivity for each curve $x \in X$ from C' (Lemma 3.3.3)
- Compute sample size $s \leftarrow \mathcal{O}(\varepsilon^{-2}d\ell k^2(m^2\ell^4)^{1/p}\log^3(m\ell)\log^2(k)\log(\varepsilon^{-1})\log(n))$
- Sample and weigh ε'-coreset S of X of size s (Theorem 3.3.7)
- Compute a 2-simplification for every $s \in S$ resulting in the set S^* (Lemma 3.4.5)
- Compute metric closure values $\overline{\phi} = \overline{\mathrm{dtw}_p\,|_{S^*}}$ (Lemma 3.4.11)
- **return** $(5+\varepsilon',1)$-approximation of weighted k-median in $(S^*,\overline{\phi})$ (Theorem 3.5.1)

Theorem 3.5.1. This is a $(4m\ell)^{1/p}(32+13\varepsilon')(1+\varepsilon')$-approximation to (k,ℓ)-median of X by Lemma 3.4.10 and Theorem 3.3.7. Overall the approximation factor is $(4m\ell)^{1/p}(32+\varepsilon)$ and the running time is in

$$\mathcal{O}\Big(n\big(m^3d + k\log k\ell^2 + (k\log k)^2\big) + \varepsilon^{-6}d^3\ell^3k^7\sqrt[p]{m^6\ell^{12}}L(n,m,\ell,k,\varepsilon^{-1})\Big),$$

where $L(n,m,\ell,k,\varepsilon^{-1}) = \log^9(m\ell)\log^6(k)\log^5(n)\log^3(\varepsilon^{-1})$. □

Combining the computed ε-coreset with the (k,ℓ)-median algorithm from [BDvG+22, Theorem 35] instead, we achieve a matching approximation guarantee and improve the dependency on n. The improved approximation guarantee from Corollary 3.5.3 compared to Theorem 3.5.2 comes at the cost of an exponential dependency in k, as is also present in their results.

Corollary 3.5.3. *Let $\varepsilon,\delta \in (0,1]$ be given. There is an $((8+\varepsilon)(m\ell)^{1/p},1)$-approximation for (k,ℓ)-median with $\Theta(1-\delta)$ success probability and running time in*

$$\widetilde{\mathcal{O}}\left(n(m^3d + k^2 + k\ell^2) + k^7 + \big(32k^2\varepsilon^{-1}\log(1/\delta)\big)^{k+2}md\left(m^3 + \varepsilon^{-2}d\ell k^2\sqrt[p]{m^2\ell^4}\right)\right),$$

where $\widetilde{\mathcal{O}}$ hides polylogarithmic factors in n, m, ℓ, k and ε^{-1}.

Finally, combining Theorem 3.5.2 with Theorem 3.3.7 yields the following result.

Corollary 3.5.4. *The algorithm (k,ℓ)-MEDIAN in Algorithm 2 can be used to construct an ε-coreset for (k,ℓ)-median in time $\widetilde{\mathcal{O}}\Big(n(m^3d + k^2 + k\ell^2d) + \varepsilon^{-6}d^3\ell^3k^7\sqrt[p]{m^6\ell^{12}}\Big)$ with constant success probability of size*

$$\mathcal{O}\left(\varepsilon^{-2}d\ell k^2(m^2\ell^2)^{1/p}\log^3(m\ell)\log^2(k)\log(\varepsilon^{-1})\log(n)\right).$$

Chapter 4

Clustering Subtrajectories

In this chapter, we study the Subtrajectory Covering and Subtrajectory Coverage Maximization problems and whether they permit efficient approximation algorithms from both a theoretical and experimental perspective.

The main content of this chapter previously appeared in the following three papers. The theoretical groundwork was laid in the paper *Faster Approximate Covering of Subcurves under the Fréchet Distance* [BCD22a] by Frederik Brüning, Jacobus Conradi, and Anne Driemel which was published in the *Proceedings of the* 30^{th} *Annual European Symposium on Algorithms (ESA 2022).* A full version of the paper is available on arXiv [BCD22b]. The extension to non-constant complexity center curves and the experimental validation appeared as the paper *Finding Complex Patterns in Trajectory Data via Geometric Set Cover* [CD23] by Jacobus Conradi and Anne Driemel on arXiv. An initial version of the work has also been presented at the 41^{st} *European Workshop on Computational Geometry (EuroCG 2025)* based on an extended abstract without formal publication. The theoretical improvements to subcubic dependency in the input complexity appeared in the paper *Subtrajectory Clustering and Subtrajectory Coverage Maximization in Cubic Time, or Better* [CD25a] by Jacobus Conradi and Anne Driemel which was published in the *Proceedings of the* 33^{rd} *Annual European Symposium on Algorithms (ESA 2025).* A full version of the paper is available on arXiv [CD25b].

4.1 Problem Definition

We begin by formally defining the problem as posed in [ABCD23].

Definition 4.1.1 (Δ-Coverage)**.** Let P be a polygonal curve in $\mathbb{R}^d$, and let $\Delta \in \mathbb{R}_{>0}$ and $\ell \in \mathbb{N}_{\geq 2}$ be given. For any $C \subset \mathbb{X}^{d,\ell}$ define the Δ-coverage of C on P as

$$\mathrm{Cov}_P(C, \Delta) = \bigcup_{Q \in C} \left(\bigcup_{0 \leq s \leq t \leq 1,\, \mathrm{d}_{\mathcal{F}}(P[s,t], Q) \leq \Delta} [s, t] \right) \subset [0, 1].$$

For a curve $Q \in \mathbb{X}^{d,\ell}$ we may denote the Δ-coverage $\mathrm{Cov}_P(\{Q\}, \Delta)$ by $\mathrm{Cov}_P(Q, \Delta)$ instead (refer to Figure 4.1).

Problem 2 (**Subtrajectory Covering (SC)**)**.** Let P be a polygonal curve in $\mathbb{R}^d$, and let $\Delta \in \mathbb{R}_{>0}$ and $\ell \in \mathbb{N}_{\geq 2}$ be given. Compute a set C of curves in $\mathbb{X}^{d,\ell}$ minimizing $|C|$ such that $\mathrm{Cov}_P(C, \Delta) = [0, 1]$.

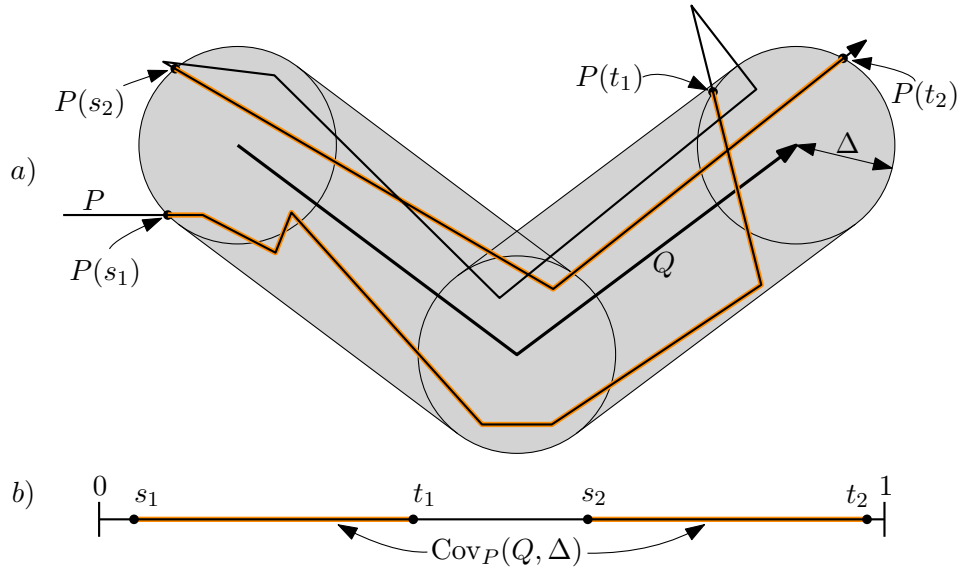


Figure 4.1: *a*): Example of all points on P that lie on subcurves of P that have Fréchet distance at most Δ to a curve Q of complexity 3. *b*): The set $\mathrm{Cov}_P(Q, \Delta) \subset [0, 1]$.

Definition 4.1.2 (Bicriteria approximation for SC)**.** Let P be a polygonal curve in $\mathbb{R}^d$, and let $\Delta \in \mathbb{R}_{>0}$ and $\ell \in \mathbb{N}_{\geq 2}$ be given. Let C^* be a set of minimal size k_Δ, such that $\mathrm{Cov}_P(C^*, \Delta) = [0, 1]$. An algorithm that outputs a set $C \subset \mathbb{X}^{d,\ell}$ of size $\alpha|C^*|$ such that $\mathrm{Cov}_P(C, \beta\Delta) = [0, 1]$ is called an (α, β)-approximation for SC.

A natural extension to the SC problem is computing k centers that cover 'as much as possible'.

Problem 3 (**Subtrajectory Coverage Maximization (SCM)**)**.** Let P be a polygonal curve in $\mathbb{R}^d$, and let $\Delta \in \mathbb{R}_{>0}$, $\ell \in \mathbb{N}_{\geq 2}$ and $k \in \mathbb{N}_{\geq 1}$ be given. Compute a set C of k curves in $\mathbb{X}^{d,\ell}$ maximizing the Lebesgue measure $\lambda(\mathrm{Cov}_P(C, \Delta))$.

Definition 4.1.3 (Bicriteria approximation for SCM)**.** Let P be a polygonal curve in $\mathbb{R}^d$, and let $\Delta \in \mathbb{R}_{>0}$ and $\ell \in \mathbb{N}_{\geq 2}$ be given. Let $C^* \subset \mathbb{X}^{d,\ell}$ be a set of size k such that $\lambda(\mathrm{Cov}_P(C^*, \Delta))$ is maximal. An algorithm that outputs a set $C \subset \mathbb{X}^{d,\ell}$ of size k such that $\lambda(\mathrm{Cov}_P(C, \beta\Delta)) \geq \alpha\lambda(\mathrm{Cov}_P(C^*, \Delta))$ is called an (α, β)-approximation for SCM.

4.2 Introduction

Prior to the line of work on the SC and SCM problem, [BBG+11] presented one of the earlier works on clustering subtrajectories for both the discrete and continuous Fréchet distance. Their work focuses on finding the largest cluster of *disjoint* subtrajectories, where different variants of 'largest' are considered. They present hardness results for $(2-\varepsilon)$-approximations for any ε and a matching polynomial time 2-approximation algorithm. In subsequent work, [GW22] presented a cubic lower-bound conditioned on SETH for the same problem. They further showed that this lower bound is tight. We remark that the condition that the subtrajectories are disjoint is essential to their lower bound construction and does not readily extend to either the SC or the SCM problem as subtrajectories defining the coverage of a center may overlap. The formulation in [BBG+11] was also studied in subsequent work [GV14, BBD+17, BBG+20, BKK20].

[AFM+18] formulate the problem of clustering subtrajectories based on facility location, an alternative well-known clustering formulation. In this problem formulation there is an opening cost associated with every center curve, a cost associated to every point on the input that is assigned to a cluster, and a different cost for points that are not assigned. They show conditional NP-hardness results and give an $\mathcal{O}(\log^2 n)$-approximation for well-behaved classes of curves under the discrete Fréchet distance.

The SC problem was first introduced in [ABCD23]. They identified a curve S that approximates the input P, such that any solution to the problem induces an approximate solution of similar size consisting of only subcurves of edges of S. This set of edges defines a set system which turns out to have low VC dimension. This enables randomized set cover algorithms resulting in an $(\mathcal{O}(d^2\ell^2 \log^2(k_\Delta d\ell)), \mathcal{O}(1))$-approximation algorithm for SC with expected running time in $\widetilde{\mathcal{O}}(k_\Delta \ell^3 \Lambda^2 \Delta^{-2} + \Lambda n \Delta^{-1})$ where Λ corresponds to the arc length of the curve P. This approach, based on the VC dimension of the set system, was improved upon in [BCD22a] resulting in an $(\mathcal{O}(\ell \log(k_\Delta \ell))), \mathcal{O}(1))$-approximation algorithm for SC with expected running time in $\mathcal{O}(n(k_\Delta)^3(\log^4(\Lambda/\Delta) + \log^3(n/k_\Delta)) + n\log^2 n)$. In this chapter we instead focus our efforts on *deterministic* algorithms for the SC and SCM problems based on the well-known greedy set cover algorithm. Subsequent to our work [BCD22a] on a purely combinatorial discretization of the set system and the public release of our arXiv manuscript [CD23], in which we extend the set system to candidates with non-constant complexity and provide an implementation of our approach, [vdHvdHO24] made an arXiv manuscript publicly available, which has recently been published [vdHvdHO25], improving upon the theoretical work in [CD23], giving additional insights into the structure of the set system. They reduce the size of the set system by a factor of roughly $n\ell$, resulting in an algorithm with running time in $\widetilde{\mathcal{O}}(k_\Delta n^3)$.

4.2.1 Results

We first give a brief overview of the results and techniques used in this chapter.

Subtrajectory Covering in Quartic Time via a Discretized Set System

We start in Section 4.3 by describing general algorithmic techniques central to both the SC and SCM problem. In particular, we describe how to obtain a *discrete* set of candidate center curves $\mathcal{C}$ consisting of few subcurves of a simplification S of the input curve P which contains an approximately optimal solution to both the SC and SCM problem. The cardinality of $\mathcal{C}$ is in $\widetilde{\mathcal{O}}(n^2)$[1], where n is the complexity of the input curve P. This results in a discrete range space, to which we apply standard greedy set cover arguments obtaining an $(\mathcal{O}(\log n), 4)$-approximation algorithm for the SC problem (Section 4.3). Its running time is in $\widetilde{\mathcal{O}}(k_\Delta n^3)$. For the special case that P is c-packed, we show that $\mathcal{C}$ has cardinality $\widetilde{\mathcal{O}}(c^2 n)$ and hence the algorithm has a running time of $\widetilde{\mathcal{O}}(k_\Delta c n^2)$.

Experiments

In Section 4.5 we present an implementation of the greedy algorithm described in Section 4.4. We evaluate our approach on GPS data of ocean drifters, with a total complexity

[1] In the original arXiv manuscript [CD23], we showed a bound of $\mathcal{O}(n^3\ell)$. The cardinality can be reduced to $\widetilde{\mathcal{O}}(n^2)$ via a critical insight into the set system presented in [vdHvdHO25]. Since we build upon the reduced set system, we present the relevant results from [vdHvdHO25] in Sections 4.3 and 4.4 as well, marking them as such.

of $n \geq 10^6$ and high-dimensional full-body motion capture data and compare the output to a state-of-the-art motion segmentation algorithm to argue the merit of our approach. We observe that, in practice, the running time is significantly better than the theoretical bound of $\widetilde{\mathcal{O}}(k_\Delta n^3)$. We further demonstrate that in practice the approximation quality of our solutions is much better than suggested by the theoretical worst-case guarantees by comparing to the size of a greedily computed independent set.

Sweep-Sequences

In Section 4.6, we analyze the discretized range space from Section 4.3, with the goal of improving the running time of the greedy set cover algorithm. In each step of the algorithm, we have to find the element $c \in \mathcal{C}$ that maximizes the coverage that is added to a partial solution. To this end, we identify a new structure, which we call ***sweep-sequences***, in Section 4.6.1. The goal is to reduce the inherently two-dimensional search space of subcurves of S to few one-dimensional search spaces that may be processed via a sequence of sweep algorithms. In Sections 4.6.2 and 4.6.3, we describe how sweep-sequences allow efficient maintenance of the Δ-coverage and with Theorem 4.6.21 describe the main interface to this identified structure enabling efficient algorithms for SC and SCM: for a weighted set $A \subset [0,1]$ of points we are able to efficiently compute the total weight of points that lie inside the Δ-coverage of every element in the sweep-sequence.

Subtrajectory Covering in Cubic Time, or Better

In Section 4.7 and Section 4.8, we present two $(\mathcal{O}(\log n), 4)$-approximation algorithms for Subtrajectory Covering based on Sections 4.3 and 4.6. We obtain a first approximation algorithm for SC with running time in $\widetilde{\mathcal{O}}(|\mathcal{C}|n + k_\Delta|\mathcal{C}|)$. The algorithm first computes the arrangement in $\widetilde{\mathcal{O}}(|\mathcal{C}|n)$ time. Afterwards, each round of the greedy algorithm runs in (roughly) logarithmic time per element in $\mathcal{C}$ by repeatedly querying the subroutine from Theorem 4.6.21. In Section 4.8, we describe an improvement to this algorithm. Instead of computing the arrangement explicitly, we first identify a representative subset of size roughly $\mathcal{O}(\sqrt{k_\Delta} n^{3/2})$ of the arrangement which we cover in an initial pass. The solution covering this representative subset already covers almost every element in the arrangement. The remaining roughly $\mathcal{O}(\sqrt{k_\Delta} n^{5/2})$ elements of the arrangement are then explicitly identified and covered in a second pass with the algorithm from Section 4.7 resulting in the following theorem.

Theorem 4.2.1. *There is a* $(96\ln(n) + 128, 4)$*-approximation for SC. Given a curve* P *of complexity* n*, together with values* $\Delta > 0$ *and* $\ell \leq n$*, its running time is in* $\mathcal{O}\left(\left(n^2\ell + \sqrt{k_\Delta} n^{\frac{5}{2}}\right)\log^2 n\right)$*, where* k_Δ *is the size of the smallest subset* $C^* \subset \mathbb{X}_\ell^d$ *such that* $\mathrm{Cov}_P(C^*, \Delta) = [0,1]$*.*

As $k_\Delta \leq \lceil \frac{n}{\ell} \rceil$, Theorem 4.2.1 yields an algorithm with (near-)cubic running time. Further, in the case that $k_\Delta \in \mathcal{O}(n^{1-\varepsilon})$, it yields an algorithm with subcubic running time.

Subtrajectory Coverage Maximization in Quadratic Time

In Section 4.9, we show how the Lebesgue measure of the coverage of the elements in $\mathcal{C}$ can be approximated efficiently by few piecewise linear functions. This approximation can also be maintained efficiently by using the identified sweep-sequences from Section 4.6,

allowing the evaluation of the aforementioned Lebesgue measure of the coverage of every element in $\mathcal{C}$ in total time $\widetilde{\mathcal{O}}(|C|)$. This results in the following theorem, which compares favorably to the best known algorithm with running time $\widetilde{\mathcal{O}}(kn^3)$ [vdHvdHO25].

Theorem 4.2.2. *Let $\varepsilon \in (0,1]$. There is an $(\frac{e-1}{16e}, 4+\varepsilon)$-approximation algorithm for SCM, where e is the base of the natural logarithm. Given a polygonal curve P of complexity n, $\Delta > 0$, $\ell \leq n$, and $k > 0$, its running time is in $\mathcal{O}((k+\ell)n^2\varepsilon^{-2}\log^2 n \log^2(\varepsilon^{-1}))$.*

4.3 Range Space Discretization

The problem can be framed as a set cover/coverage maximization problem of the range space

$$\left([0,1], \left\{ \mathrm{Cov}_P(\pi, \Delta) \middle| \pi \in \mathbb{X}^{d,\ell} \right\}\right).$$

Unfortunately, both the ground set as well as the set system of this range space are uncountably large. In this section we will work towards a discrete approximation of this range space in the sense that computing an approximate set cover solution in the approximate range space amounts to computing an approximate set cover solution in the original range space.

4.3.1 Maximal Curve Simplifications

We begin by defining the notion of curve simplification central to this chapter. Our notion of curve simplification is inspired by [dBCG13] and [DHW12].

Definition 4.3.1 (Maximal Simplification)**.** Let P be a polygonal curve in $\mathbb{R}^d$. Let $(t_1, \ldots, t_n)$ be the vertex parameters of P, and $p_i = P(t_i)$ the vertices of P. Consider indices $1 \leq i_1 < \ldots < i_k \leq n$ defining vertices p_{i_j}. We call a curve S defined by such an ordered set of vertices $(p_{i_1}, \ldots, p_{i_k}) \in (\mathbb{R}^d)^k$ a ***simplification*** of P. We say a simplification S of P is (Δ, ε)-***maximal***, if the following properties hold:

(i) $\|p_{i_j} - p_{i_{j+1}}\| \geq \varepsilon$ for $j \in [k-1]$,
(ii) $\mathrm{d}_{\mathcal{F}}(P[t_{i_j}, t_{i_{j+1}}], \overline{p_{i_j}\, p_{i_{j+1}}}) \leq \Delta + \varepsilon$ for all $j \in [k-1]$,
(iii) $\mathrm{d}_{\mathcal{F}}(P[t_1, t_{i_1}], \overline{p_{i_1}\, p_{i_1}}) \leq \Delta + \varepsilon$ and $\mathrm{d}_{\mathcal{F}}(P[t_{i_k}, t_n], \overline{p_{i_k}\, p_{i_k}}) \leq \Delta + \varepsilon$, and
(iv) $\mathrm{d}_{\mathcal{F}}(P[t_{i_j}, t_{i_{j+2}}], \overline{p_{i_j}\, p_{i_{j+2}}}) > \Delta$ for all $j \in [k-2]$.

Note that a (Δ, ε)-maximal simplification S of a curve P has small Fréchet distance to P, i.e., $\mathrm{d}_{\mathcal{F}}(S, P) \leq \Delta + \varepsilon$ by property (ii) and (iii). Outside the context of c-packed curves, we usually only talk about $(\Delta, 0)$-maximal simplifications, which we may simply call Δ-maximal. In fact, the notion of a $(\Delta, 0)$-maximal simplification coincides with the simplification introduced in [dBCG13], and the notion of a $(0, \varepsilon)$-maximal simplification coincides with the simplification introduced in [DHW12].

We turn to proving that any solution to the SC and SCM problem can be restricted to subcurves of a maximal simplification of P.

Definition 4.3.2 (Container [dBCG13])**.** Let P be a polygonal curve, let $\pi = P[s,t]$ be a subcurve of P, and let $(t_1, \ldots, t_n)$ be the vertex parameters of P. For a simplification S of P defined by index set $I = (i_1, \ldots, i_k)$, define the ***container*** $c_S(\pi)$ of π on S as $S[t_a, t_b]$, with $a = \max(\{i_1\} \cup \{i \in I \mid t_i \leq s\})$ and $b = \min(\{i \in I \mid t_i \geq t\} \cup \{i_k\})$.

[dBCG13] proved the following lemma for 2Δ-maximal simplifications. We restate and reprove it here with respect to our notion of simplification.

Lemma 4.3.3 ([dBCG13])**.** *Let P be a polygonal curve in $\mathbb{R}^d$, and let S be a $(2\Delta, \varepsilon)$-maximal simplification of P. Let $Q \in \mathbb{X}^{d,2}$ be an edge in $\mathbb{R}^d$ and let π be a subcurve of P with $\mathrm{d}_{\mathcal{F}}(Q, \pi) \leq \Delta$. Then $c_S(\pi)$ consists of at most 3 edges.*

Proof. Assume for the sake of contradiction that $c_S(\pi)$ contains at least 4 edges, that is, it has three internal vertices s_1, s_2, s_3. By Definition 4.3.2, these three vertices are also interior vertices of π. As $\mathrm{d}_{\mathcal{F}}(Q, \pi) \leq \Delta$, there are points q_1, q_2, and q_3 along Q that get matched to s_1, s_2, and s_3 respectively during a traversal with associated cost Δ, and hence $\|s_i - q_i\| \leq \Delta$. This implies $\mathrm{d}_{\mathcal{F}}(\pi[s_1, s_3], \overline{q_1\, q_3}) \leq \Delta$. It also implies that $\mathrm{d}_{\mathcal{F}}(\overline{s_1\, s_3}, \overline{q_1\, q_3}) \leq \Delta$. But then

$$\mathrm{d}_{\mathcal{F}}(\overline{s_1\, s_3}, P[s_1, s_3]) = \mathrm{d}_{\mathcal{F}}(\overline{s_1, s_3}, \pi[s_1, s_3]) \leq \mathrm{d}_{\mathcal{F}}(\overline{s_1\, s_3}, \overline{q_1\, q_3}) + \mathrm{d}_{\mathcal{F}}(\pi[s_1, s_3], \overline{q_1\, q_3}) \leq 2\Delta,$$

contradicting the assumption that S is a $(2\Delta, \varepsilon)$-maximal simplification. □

Via the pigeon hole principle, we extend this lemma to curves of higher complexity.

Lemma 4.3.4. *Let P be a polygonal curve in $\mathbb{R}^d$ and let S be a $(2\Delta, \varepsilon)$-maximal simplification of P. Let $Q \in \mathbb{X}^{d,\ell}$ and let π be a subcurve of P with $\mathrm{d}_{\mathcal{F}}(Q, \pi) \leq \Delta$. Then $c_S(\pi)$ consists of at most $2\ell + 1$ edges.*

Proof. Assume for the sake of contradiction that $c_S(\pi)$ contains $2\ell + 2$ edges, that is, it has $2\ell+1$ internal vertices. By the pigeon hole principle and the fact that $\mathrm{d}_{\mathcal{F}}(Q, \pi) \leq \Delta$, there is an edge Q_i of Q and three points q_1, q_2, and q_3 along Q_i that get matched with three such internal vertices during the traversal of Q and π. But then there is a subcurve π' of π such that $\mathrm{d}_{\mathcal{F}}(Q_i, \pi') \leq \Delta$ and $c_S(\pi')$ has at least 4 edges contradicting Lemma 4.3.3. □

Corollary 4.3.5. *Let P be a polygonal curve and let S be a $(2\Delta, \varepsilon)$-maximal simplification of P. For any curve $\pi \in \mathbb{X}^{d,\ell}$ there is a subcurve $\hat{\pi}$ of S of complexity at most $2\ell + 1$—that is $\hat{\pi} \in \mathbb{X}^{d,2\ell+1}$—such that $\mathrm{Cov}_P(\pi, \Delta) \subset \mathrm{Cov}_P(\hat{\pi}, 4\Delta + \varepsilon)$.*

Proof. This is an immediate consequence of Lemma 4.3.4, the definition of a $(2\Delta, \varepsilon)$-simplification and the triangle inequality. □

Lemma 4.3.6. *Let π be a curve. For any $0 \leq t \leq t' \leq 1$ it holds that*

$$\mathrm{Cov}_P(\pi, \Delta) \subset \mathrm{Cov}_P(\{\pi[0, t'], \pi[t, 1]\}, \Delta).$$

Proof. Let $P[a, b]$ be such that $\mathrm{d}_{\mathcal{F}}(\pi, P[a, b]) \leq \Delta$. Then there are values $a \leq s \leq s' \leq b$ such that $\mathrm{d}_{\mathcal{F}}(\pi[0, t'], P[a, s']) \leq \Delta$ and $\mathrm{d}_{\mathcal{F}}(\pi[t, 1], P[s, b]) \leq \Delta$. And hence

$$[a, b] = [a, s'] \cup [s, b] \subset \mathrm{Cov}_P(\{\pi[0, t'], \pi[t, 1]\}, \Delta).$$

□

Summarizing, Corollary 4.3.5 and Lemma 4.3.6 imply that for any $(2\Delta, \varepsilon)$-maximal simplification S of P and curve $\pi \in \mathbb{X}^{d,\ell}$ there are at most three subcurves π_1, π_2, and π_3 of S of complexity at most ℓ such that $\mathrm{Cov}_P(\pi, \Delta) \subset \mathrm{Cov}_P(\{\pi_1, \pi_2, \pi_3\}, 4\Delta + \varepsilon)$.

The following lemma suggests that we may cover a $(0, \varepsilon\Delta)$-maximal simplification P' of P instead of P, relaxing Δ by at most a factor of $(1 + \varepsilon)$.

Lemma 4.3.7. *Let P and P' be curves such that $\mathrm{d}_{\mathcal{F}}(P, P') \leq \varepsilon$. Let $C \subset \mathbb{X}^{d,\ell}$ such that $\mathrm{Cov}_P(C, \Delta) = [0, 1]$. Then $\mathrm{Cov}_{P'}(C, \Delta + \varepsilon) = [0, 1]$.*

Proof. Let (f, g) be a traversal of P and P', with associated cost at most ε. Let $\mu_P(x) = \{y \in [0, 1] \mid \exists t \in [0, 1] : f(t) = x,\ g(t) = y\}$, that is, all the (parametrized) points along P' that get matched to $P(x)$ during some traversal with associated distance at most ε. Note that $\mu_P([0, 1]) = [0, 1]$, and more importantly for $[a, b] \subset [c, d]$, it holds that $\mu_P([a, b]) \subset \mu_P([c, d])$, as f and g are monotone.

We claim that

$$[0, 1] = \mu_P([0, 1]) = \mu_P\left(\mathrm{Cov}_P(C, \Delta)\right) \subseteq \mathrm{Cov}_{P'}(C, \Delta + \varepsilon)$$

implying $\mathrm{Cov}_{P'}(C, \Delta + \varepsilon) = [0, 1]$.

Observe that by the triangle inequality it holds for any $0 \leq t \leq t' \leq 1$ and any $Q \in C$ with $\mathrm{d}_{\mathcal{F}}(P[t, t'], Q) \leq \Delta$, and for any $s \in \mu_P(t)$ and $s' \in \mu_P(t')$ that

$$\mathrm{d}_{\mathcal{F}}(P'[s, s'], Q) \leq \mathrm{d}_{\mathcal{F}}(P'[s, s'], P[t, t']) + \mathrm{d}_{\mathcal{F}}(P[t, t'], Q) \leq \Delta + \varepsilon.$$

Hence

$$\begin{aligned}
[0, 1] = \mu_P\left(\mathrm{Cov}_P(C, \Delta)\right) &= \bigcup_{Q \in C} \bigcup_{0 \leq t \leq t' \leq 1} \{x \in \mu_P([t, t']) \mid \mathrm{d}_{\mathcal{F}}(P[t, t'], Q) \leq \Delta\} \\
&\subset \bigcup_{Q \in C} \bigcup_{0 \leq s \leq s' \leq 1} \{x \in [s, s'] \mid \mathrm{d}_{\mathcal{F}}(P'[s, s'], Q) \leq \Delta + \varepsilon\} \\
&= \mathrm{Cov}_{P'}(C, \Delta + \varepsilon).
\end{aligned}$$

The second step follows from the above observation since $\mu_P([t, t']) = [s, s']$ for any $s \in \mu_P(t)$ and $s' \in \mu_P(t')$ with $s \leq s'$ since f and g are monotone. □

4.3.2 Set System Discretization

In this section, we describe how, given a curve P of complexity n, and a 2Δ-maximal simplification S there is a set $\mathcal{C}_S$ of $\mathcal{O}(n^2 \log n)$ subcurves of S such that for any set of curves $C \subset \mathbb{X}^{d,\ell}$ there is a subset $\hat{C} \subset \mathcal{C}_S$ of size $\mathcal{O}(|C|)$ such that $\mathrm{Cov}_P(C, \Delta) \subset \mathrm{Cov}_P(\hat{C}, 4\Delta)$. We will prove this statement in more general forms w.r.t. (α, Δ)-approximate freespaces and $(2\Delta, \varepsilon)$-simplifications instead, as these underlie our results for SC for general and c-packed curves as well as SCM for general curves. We begin by defining $\mathcal{C}_S$.

Definition 4.3.8 (Subcurve types)**.** Let S be a polygonal curve. Let π be a subcurve of S. We say π is a ***vertex-vertex-subcurve*** of S if π starts and ends at vertices of S. We say π is a ***subedge*** of S, if π has complexity 2, i.e., it is a subcurve of a single edge of S. We say π is an ***edge-affix*** of S if it is a subedge of S which starts or ends at a vertex of S.

Definition 4.3.9 (Type (I)-, (II)- and (III)-subcurves)**.** Let P be a polygonal curve and let S be a simplification of P. Let α, Δ and ℓ be given. Let A_S be an (α, Δ)-free space of S and P. For every edge e of S let A_e be the restriction of A_S onto the edge e—that is A_e is an (α, Δ)-free space of e and P—and let $E_e = \{\varepsilon_1, \ldots\}$ be a finite sorted superset of the y-coordinates of extremal points $\mathcal{E}(A_e)$ of A_e (refer to Figure 4.2). Define three types of subcurves of S via the set $E = (E_{e_1}, E_{e_2}, \ldots)$.

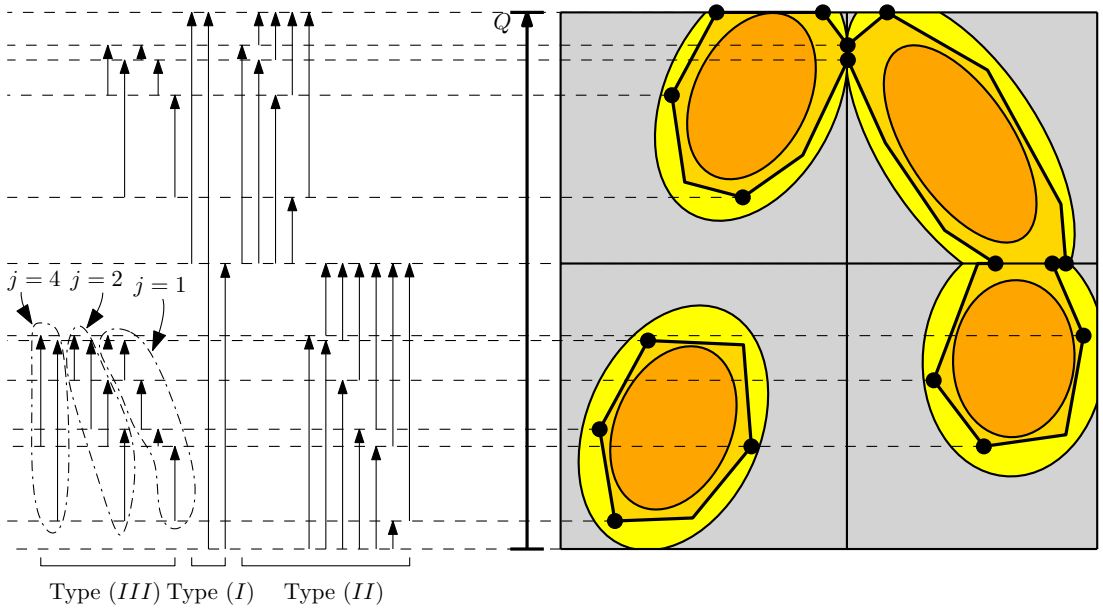


Figure 4.2: Illustration of all Type (I)-, (II)- and (III)-subcurves of Q (as vertical lines) that are not reversals, induced by an approximate free space. Further marked are the values j, which induced the set of Type (III)-subcurves on the first edge of Q.

- (I): A Type (I)-subcurve of S is a vertex-vertex-subcurve of S that starts at some i^{th} vertex of S and ends at the $(i+j)^{\text{th}}$ vertex for $j \in \{2^m \mid 0 \leq m \leq \lfloor \log_2(\ell) \rfloor\}$.
- (II): A Type (II)-subcurve of S induced by E is either an affix-subcurve $e[0, \varepsilon_i]$ or $e[\varepsilon_i, 1]$ of some edge e of S that is defined by a vertex of S and some value $\varepsilon_i \in E_e$ or its reversal $\text{rev}(e[0, \varepsilon_i])$ or $\text{rev}(e[\varepsilon_i, 1])$.
- (III): A Type (III)-subcurve of S induced by E is either a subedge $e[\varepsilon_i, \varepsilon_{i+j}]$ of an edge e of S that is defined by two values $\varepsilon_i, \varepsilon_{i+j} \in E_e$, such that j is a power of 2, i.e., $j \in \{2^m \mid 0 \leq m \leq \lfloor \log_2(|E_e|) \rfloor\}$, or its reversal $\text{rev}(e[\varepsilon_i, \varepsilon_{i+j}])$.

By a slight abuse of notation we may call E a finite sorted superset of the y-coordinates of $\mathcal{E}(A_S)$ and E_e the restriction of E onto the edge e of S. The set of all Type (I)-, (II)-, and (III)-subcurves of S induced by E we denote by $\mathcal{C}_S(E)$.

The set $\mathcal{C}_S(E)$ serves as our discretized set system. A variant of this set of candidates was originally presented in [vdHvdHO25]. We now analyze its central properties w.r.t. the notion of the coverage computed via an approximate free space.

Definition 4.3.10 (Free Space Coverage)**.** Let A be an (α, Δ)-free space of curves S and P. For $0 \leq b \leq d \leq 1$ define the free space coverage $\text{Cov}_A(S[b,d])$ of $S[b,d]$ to be the union of all intervals $[a,c] \subset [0,1]$ such that there is a monotone path from (a,b) to (c,d) inside A.

Trivially, for an (α, Δ)-free space of S and P and values $0 \leq b \leq d \leq 1$ it holds that

$$\text{Cov}_P(S[b,d], \Delta) \subset \text{Cov}_A(S[b,d]) \subset \text{Cov}_P(S[b,d], \alpha\Delta).$$

We may extend Lemma 4.3.6 to the free space coverage.

Observation 4.3.11. *Let S and P be curves. Let A be an approximate free space of S and P. Then for any $0 \leq a \leq b \leq c \leq d \leq 1$ it holds that*

$$\mathrm{Cov}_A(S[a,d]) \subset \mathrm{Cov}_A(\{S[a,c], S[b,d]\}).$$

Recall (Section 2.3) that any (α, Δ)-free space of curves S and P is also an (α, Δ)-free space A' of $\mathrm{rev}(S)$ and P and as such we extend the free space coverage to subcurves of reversals of S, with $\mathrm{Cov}_A(\mathrm{rev}(S[a,b])) := \mathrm{Cov}_{A'}(\mathrm{rev}(S)[1-b, 1-a])$. Observe that if A is the $(1, \Delta)$-free space then $\mathrm{Cov}_A(\cdot) = \mathrm{Cov}_P(\cdot, \Delta)$.

Definition 4.3.12. Let e be an edge and let P be a polygonal curve. Let A_e be an (α, Δ)-free space of e and P. Then define $l_i(\cdot)$ as the function mapping any y to the x-coordinate of the leftmost point of the i^{th} cell in A_e at height y. If this point does not exist, $l_i(y) = \infty$. Similarly define $r_j(y)$ to be the x-coordinate of the rightmost point at height y, and ∞ otherwise.

The two types of free spaces we will discuss in this section are the ones discussed in Section 2.3. For a $(1, \Delta)$-free space both $l_i(\cdot)$ and $r_j(\cdot)$ can be computed in $\mathcal{O}(1)$ time, and for a $(1+\varepsilon, \Delta)$-free space which, inside any cell, consists of a convex polygon of complexity at most $\mathcal{O}(\varepsilon^{-2})$, $l_i(\cdot)$ and $r_j(\cdot)$ may be computed in $\mathcal{O}(\log(\varepsilon^{-1}))$ time via a binary search over the $\mathcal{O}(\varepsilon^{-2})$ edges defining the polygon representing the free space in each cell.

Lemma 4.3.13. *Let P be a polygonal curve and let S be a simplification of P. Let Δ be given. Let π be a subcurve of S of complexity at most $2\ell+1$. There is a set S'_π consisting of either one subedge of S or two vertex-vertex-subcurve, of complexity at most ℓ and two edge-affixes of S such that*

$$\mathrm{Cov}_P(\pi, \Delta) \subset \bigcup_{s \in S'_\pi} \mathrm{Cov}_P(s, \Delta).$$

Proof. This is an immediate consequence of Lemma 4.3.6 splitting the curve π at its second, $(\ell+1)^{\text{th}}$, and penultimate vertex parameter. □

Lemma 4.3.14. *Let P be a polygonal curve and let S be a simplification of P. Let A_S be an (α, Δ)-free space of S and P. Let E be a finite sorted superset of the y-coordinates of $\mathcal{E}(A_S)$. Let further e be an edge of S and let π be a subcurve of the edge e. Then there is a set $S''_\pi = \{e[s_1,t_1], \ldots, e[s_4,t_4]\}$ consisting of at most four subedges of the edge e of S starting and ending at values $s_1, t_1, \ldots, s_4, t_4 \in E_e$ and*

$$\mathrm{Cov}_P(\hat{\pi}, \Delta) \subset \bigcup_{s \in S''_\pi} \mathrm{Cov}_{A_S}(s).$$

If $\hat{\pi}$ is an edge-affix, then S''_π consists of only two such subcurves which are edge-affixes of e starting and ending at values in E_e.

Proof. Let $0 \leq s \leq t \leq 1$ such that $\pi = e[s,t]$. Let ε_1 be the largest value among E_e such that $\varepsilon_1 \leq s$ and let ε_2 be the smallest value among E_e such that $\varepsilon_2 \geq s$. Let similarly ε_3 be the largest value among E_e such that $\varepsilon_3 \leq t$ and ε_4 be the smallest value among E_e such that $\varepsilon_4 \geq t$. In particular, both ε_1 and ε_2 as well as ε_3 and ε_4 are consecutive in E_e.

Assume first that $\varepsilon_2 \leq \varepsilon_3$. Observe that there is no extremal point of A_e between s and ε_1, and between s and ε_2, and there is no extremal point of A_e between t and ε_3,

and between t and ε_4. Hence, if there is a monotone path from some cell i at height s to cell j at height t, then there are also monotone paths from cell i at height ε_1 (resp. ε_2) to cell j at height ε_3 (resp. ε_4). By the convexity of the free space in every cell and the fact that the y-coordinates of the leftmost points are also in E_e it holds further that $\min(l_i(\varepsilon_1), l_i(\varepsilon_2)) \leq l_i(s)$ and $r_i(t) \leq \max(r_j(\varepsilon_3), r_j(\varepsilon_4))$. And thus

$$\mathrm{Cov}_P(\pi, \Delta) \subset \mathrm{Cov}_{A_S}(\{e[\varepsilon_1, \varepsilon_3], e[\varepsilon_1, \varepsilon_4], e[\varepsilon_2, \varepsilon_3], e[\varepsilon_2, \varepsilon_4]\}).$$

If s itself is an extremal coordinate, then $\varepsilon_1 = \varepsilon_2$. Similarly if t itself is an extremal coordinate, then $\varepsilon_3 = \varepsilon_4$. This is the case if π is an edge-affix. In this case, $\mathrm{Cov}_P(\pi, \Delta)$ is a subset of either $\mathrm{Cov}_{A_S}(\{e[\varepsilon_1, \varepsilon_3], e[\varepsilon_1, \varepsilon_4]\})$ or $\mathrm{Cov}_{A_S}(\{e[\varepsilon_1, \varepsilon_3], e[\varepsilon_2, \varepsilon_3]\})$.

If instead $\varepsilon_2 > \varepsilon_3$ then in particular $\varepsilon_1 = \varepsilon_3 < \varepsilon_2 = \varepsilon_4$. Similarly as before, observe that there is no extremal point of A_e between s and ε_1, and between s and ε_2, and there is no extremal point of A_e between t and ε_3, and between t and ε_4. Hence, if there is a monotone path from some cell i at height s to cell j at height t, then there are also monotone paths from cell i at height ε_1 to cell j at height ε_3. There are further monotone paths from cell i at height ε_1 to cell j at height ε_4 and from cell i at height ε_2 to cell j at height ε_4. Lastly there is also a path from cell i at height ε_3 to cell j at height ε_2 that is monotonously increasing in the x-coordinate and monotonously *decreasing* in the y-coordinate. But this implies that

$$\mathrm{Cov}_P(\pi, \Delta) \subset \mathrm{Cov}_{A_S}(\{e[\varepsilon_1, \varepsilon_3], e[\varepsilon_1, \varepsilon_4], \mathrm{rev}(e[\varepsilon_3, \varepsilon_2]), e[\varepsilon_2, \varepsilon_4]\}).$$

If s itself is an extremal coordinate, then $\varepsilon_1 = \varepsilon_2$. Similarly if t itself is an extremal coordinate, then $\varepsilon_3 = \varepsilon_4$. This is the case if π is an edge-affix. In this case $\mathrm{Cov}_P(\pi, \Delta)$ is a subset of either $\mathrm{Cov}_{A_S}(\{e[\varepsilon_1, \varepsilon_3], e[\varepsilon_1, \varepsilon_4]\})$ or $\mathrm{Cov}_{A_S}(\{e[\varepsilon_1, \varepsilon_3], \mathrm{rev}(e[\varepsilon_3, \varepsilon_2])\})$, which concludes the proof.

□

The following two theorems are morally the same with slightly modified proofs and statements geared towards SC and SCM. The underlying technique is the 2^j*-trick*: Given a subedge π defined by the a^{th} and b^{th} y-coordinate ε_a and ε_b in E_e defined by an approximate free space A, that is, $\pi = e[\varepsilon_a, \varepsilon_b]$, let $j = \lfloor \log_2(b-a) \rfloor$. Then $e[\varepsilon_a, \varepsilon_{a+2^j}]$ and $e[\varepsilon_{b-2^j}, \varepsilon_b]$ are both Type (III)-subcurves, and as $a + 2^j \geq b - 2^j$ we have by Observation 4.3.11 that $\mathrm{Cov}_A(\pi) \subset \mathrm{Cov}_A(\{e[\varepsilon_a, \varepsilon_{a+2^j}], e[\varepsilon_{b-2^j}, \varepsilon_b]\})$. This similarly applies for vertex-vertex-subcurves resulting in two Type (I)-subcurves.

Theorem 4.3.15 (adapted from [vdHvdHO25])**.** *Let P be a polygonal curve and let $\varepsilon \in (0, 1]$, $\Delta \geq 0$, and $\ell \in \mathbb{N}$ be given. Let further P' be a $(0, \varepsilon\Delta)$- and S a $(2\Delta, \varepsilon\Delta)$-maximal simplification of P. Let E be the y-coordinates of all extremal points $\mathcal{E}(\mathcal{D}_{(4+2\varepsilon)\Delta}(S, P'))$. Let $C \subset \mathbb{X}^{d,\ell}$ be a set such that $\mathrm{Cov}_P(C, \Delta) = [0, 1]$. Then there is a set $C' \subset \mathcal{C}_S(E)$ of size $8|C|$ such that*

$$[0, 1] = \mathrm{Cov}_A(C') \subset \mathrm{Cov}_P(C', (4 + 3\varepsilon)\Delta).$$

Proof. Let $C \subset \mathbb{X}^{d,\ell}$ be such that $\mathrm{Cov}_P(C, \Delta) = [0, 1]$. Corollary 4.3.5 together with Lemma 4.3.14 imply that for every $c \in C$ there is a set S_c consisting of either one subedge of S or two vertex-vertex-subcurves of complexity at most ℓ and two edge-affixes of S such that $\mathrm{Cov}_P(\bigcup_{c \in C} S_c, (4 + \varepsilon)\Delta) = [0, 1]$. Hence by Lemma 4.3.7 we have that $\mathrm{Cov}_P(\bigcup_{c \in C} S_c, (4 + 2\varepsilon)\Delta) = [0, 1]$ and thus $\mathrm{Cov}_A(\bigcup_{c \in C} S_c) = [0, 1]$. Now it suffices

to prove that for any such set S_c Lemma 4.3.14 and Observation 4.3.11 imply that for any S_c there is a set S'_c consisting of at most 8 Type (I)-, (II)- or (III)-subcurves induced by E such that $\mathrm{Cov}_A(S_c) \subset \mathrm{Cov}_A(S'_c)$.

If S_c consists of one subedge, Lemma 4.3.14 implies that there are four subedges π_1, π_2, π_3, and π_4 starting and ending at values in E such that the coverage $\mathrm{Cov}_A(S_c)$ is a subset of $\mathrm{Cov}_A(\{\pi_1, \pi_2, \pi_3, \pi_4\})$, and for any such π_i the 2^j-trick implies that there are two Type (III)-subcurves $\pi_{i,1}$ and $\pi_{i,2}$ such that $\mathrm{Cov}_A(\pi_i) \subset \mathrm{Cov}_A(\{\pi_{i,1}, \pi_{i,2}\})$ implying the claim.

If instead S_c consists of two edge-affixes and two vertex-vertex-subcurves of complexity at most ℓ, then similarly via the 2^j-trick for any vertex-vertex-subcurve π of complexity at most ℓ there are Type (I)-subcurves π_1 and π_2 with $\mathrm{Cov}_A(\pi) \subset \mathrm{Cov}_A(\{\pi_1, \pi_2\})$, and by Lemma 4.3.14 for any edge-affix π there are two Type (II)-subcurves π_1 and π_2 such that $\mathrm{Cov}_A(\pi) \subset \mathrm{Cov}_A(\{\pi_1, \pi_2\})$ implying the claim.

Hence overall we identified a set C' of Type (I)-, (II)-, and (III)-subcurves induced by E of size at most $8|C|$ such that $\mathrm{Cov}_A(C') = [0, 1]$. Lastly, Lemma 4.3.7 implies that $[0, 1] \subset \mathrm{Cov}_P(C', (4 + 3\varepsilon)\Delta)$. □

Theorem 4.3.16. *Let P be a polygonal curve and let $\Delta \geq 0$, $\alpha \geq 1$ and $\ell \in \mathbb{N}$ be given. Let further S be a 2Δ-maximal simplification of P. Let A be an $(\alpha, 4\Delta)$-free space of S and P. Let E be a finite sorted superset of the y-coordinates of all extremal points $\mathcal{E}(A)$. Let $C \subset \mathbb{X}^{d,\ell}$. Then there is a set $C' \subset \mathcal{C}_S(E)$ of size $8|C|$ such that*

$$\mathrm{Cov}_P(C, \Delta) \subset \mathrm{Cov}_A(C') \subset \mathrm{Cov}_P(C', 4\alpha\Delta).$$

Proof. Let $C \subset \mathbb{X}^{d,\ell}$ be given. By Corollary 4.3.5 and Lemma 4.3.14 for every $c \in C$ there is a set S_c consisting of either one subedge of S or two vertex-vertex-subcurves of complexity at most ℓ and two edge-affixes of S such that $\mathrm{Cov}_P(C, \Delta) \subset \mathrm{Cov}_P(\bigcup_{c \in C} S_c, 4\Delta)$. Hence we have that $\mathrm{Cov}_P(C, \Delta) \subset \mathrm{Cov}_A(\bigcup_{c \in C} S_c)$. Now it suffices to prove that for any such set S_c, Lemma 4.3.14 and Observation 4.3.11 imply that for any S_c there is a set S'_c consisting of at most 8 Type (I)-, (II)- or (III)-subcurves induced by E such that $\mathrm{Cov}_A(S_c) \subset \mathrm{Cov}_A(S'_c)$.

If S_c consists of one subedge, Lemma 4.3.14 implies that there are four subedges π_1, π_2, π_3, and π_4 starting and ending at values in E such that the coverage $\mathrm{Cov}_A(S_c)$ is a subset of $\mathrm{Cov}_A(\{\pi_1, \pi_2, \pi_3, \pi_4\})$, and for any such π_i the 2^j-trick implies that there are two Type (III)-subcurves $\pi_{i,1}$, and $\pi_{i,2}$ such that $\mathrm{Cov}_A(\pi_i) \subset \mathrm{Cov}_A(\{\pi_{i,1}, \pi_{i,2}\})$ implying the claim.

If instead S_c consists of two edge-affixes and two vertex-vertex-subcurves of complexity at most ℓ, then similarly via the 2^j-trick for any vertex-vertex-subcurve π of complexity at most ℓ there are Type (I)-subcurves π_1 and π_2 with $\mathrm{Cov}_A(\pi) \subset \mathrm{Cov}_A(\{\pi_1, \pi_2\})$, and by Lemma 4.3.14 for any edge-affix π there are two Type (II)-subcurves π_1 and π_2 such that $\mathrm{Cov}_A(\pi) \subset \mathrm{Cov}_A(\{\pi_1, \pi_2\})$ implying the claim.

Hence overall we identified a set C' of Type (I)-, (II)-, and (III)-subcurves induced by E of size at most $8|C|$ such that $\mathrm{Cov}_P(C) \subset \mathrm{Cov}_A(C')$. Lastly, as A is an (α, Δ)-approximate free space, $\mathrm{Cov}_A(C') \subset \mathrm{Cov}_P(C', 4\alpha\Delta)$. □

4.4 Subtrajectory Covering via Greedy Set Cover

In this section we focus exclusively on the SC problem. In particular, the set E inducing $\mathcal{C}_S(E)$ will be exactly the set of y-coordinates of extremal points of the given free space,

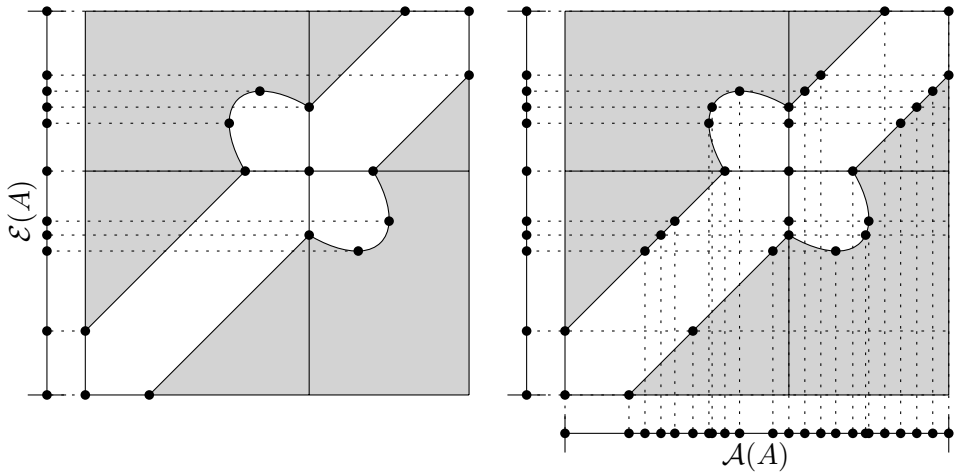


Figure 4.3: Construction of atomic intervals based on the extremal points $\mathcal{E}(A)$ of a free space A.

and not an arbitrary super set. We begin by discretizing the ground set of the range space which in turn makes the use of classical set cover algorithms feasible.

Definition 4.4.1 (Atomic Intervals)**.** Let A be an approximate Δ-free space. Let G be the set of all intersection points of horizontal lines rooted at y for $y \in \mathcal{E}(A)$ with the boundary of the free space A. From this, define the set of atomic intervals $\mathcal{A}(A)$ as the set of intervals describing the arrangement of $[0,1]$ defined by the set $\{x \in [0,1] \mid \exists y \in [0,1] : (x,y) \in G\}$ (Figure 4.3).

Observe that for any atomic interval $a \in \mathcal{A}(A)$ and any Type (I)-, (II)- or (III)-subcurve π induced by the y-coordinates of $\mathcal{E}(A)$ either $a \subset \mathrm{Cov}_A(\pi)$, or $a \cap \mathrm{Cov}_A(\pi) = \emptyset$. Hence the following observation holds.

Observation 4.4.2. *Let S and P be polygonal curves of complexity at most n and let $\Delta > 0$ be given. Then $|\mathcal{E}(\mathcal{D}_\Delta(S,P))| \leq 8n^2$. As each horizontal line intersects at most n cells, and the free space in every cell is convex it follows that the set of all midpoints of atomic intervals $\mathcal{A}(\mathcal{D}_\Delta(S,P))$ is a set of points $A \subset [0,1]$ of size $16n^3$ such that for any set C of Type (I)-, (II)-, and (III)-subcurves induced by y-coordinates of points in $\mathcal{E}(\mathcal{D}_\Delta(S,P))$ it holds that*

$$\mathrm{Cov}_P(C,\Delta) = [0,1] \iff A \subset \mathrm{Cov}_P(C,\Delta).$$

The Subtrajectory Covering algorithm we present relies upon the following theorem.

Theorem 4.4.3. *Let P be a curve of complexity n, let Δ, ε and ℓ be given. Let S be a $(2\Delta, \varepsilon\Delta)$-maximal simplification of P and let P' be a $(0, \varepsilon\Delta)$-maximal simplification of P. Let $A \subset [0,1]$ be the set of midpoints of atomic intervals of the $(4\Delta + 2\varepsilon\Delta)$-free space of S and P'. Let E be the set of y-coordinates of $\mathcal{E}(\mathcal{D}_{(4+2\varepsilon)\Delta}(S,P))$. Let k_Δ be the size of a smallest set $\mathcal{C}^* \subset \mathbb{X}^{d,\ell}$ such that $\bigcup_{c \in \mathcal{C}^*} \mathrm{Cov}_P(c,\Delta) = [0,1]$. Any algorithm that iteratively adds the curve c among $\mathcal{C}_S(E)$ to R maximizing*

$$\left| \left\{ a \in A \middle| a \in \left(\mathrm{Cov}_P(c, 4\Delta + 2\varepsilon\Delta) \setminus \left(\bigcup_{r \in R} \mathrm{Cov}_P(r, 4\Delta + 2\varepsilon\Delta) \right) \right) \right\} \right|$$

terminates after $8(3\ln(n) + \ln(16) + 1)k_\Delta$ iterations.

Algorithm 3 Curve simplification

```
1: procedure SimplifyCurve(curve P in R^d, ε, Δ)
2:     Let Ŝ be an empty stack.
3:     Ŝ.push(1)
4:     for 2 ≤ i ≤ n do
5:         j ← Ŝ.next_to_top()
6:         while j is defined and d_F(P[t_j, t_i], p_j p_i) ≤ Δ do
7:             Ŝ.pop()
8:             j ← Ŝ.next_to_top()
9:         j ← Ŝ.top()
10:        if ‖P(t_j) − P(t_i)‖ ≥ ε then
11:            Ŝ.push(i)
12:    return simplification S defined by the indices in Ŝ
```

Proof. By Theorem 4.9.1 there is a set C of size $8k_\Delta$ in $\mathcal{C}_S(E)$ such that $\mathrm{Cov}_{A_S}(C) = [0,1] \supset A$. By standard arguments for the greedy set cover algorithm [Joh73, Lov75, Ste74], the algorithm thus terminates after $(\ln|A|+1)8k_\Delta$ iterations. □

4.4.1 Simplification Algorithm

In this section we describe an algorithm to construct a (Δ,ε)-maximal simplification S for a given polygonal curve P.

Theorem 4.4.4. *Let P be a polygonal curve in $\mathbb{R}^d$. Let $(t_1,\ldots,t_n)$ be the vertex parameters of P and $p_i = P(t_i)$ its vertices. Let $\Delta \geq 0$ and $\varepsilon \geq 0$ be given. There exists an algorithm that outputs an index set defining a (Δ,ε)-maximal simplification of P. Furthermore, it does so in $\mathcal{O}(n^2)$ time and $\mathcal{O}(n)$ space.*

Proof. Consider Algorithm 3. We want to show that the simplification S of P defined by the index set $\hat{S}$ is (Δ,ε)-maximal. For this we have to show that S fulfills properties (i)–(iv) of Definition 4.3.1. Note that property (i) follows immediately, as otherwise any index considered in line 11 would not have been added to $\hat{S}$.

Denote by s the last item of $\hat{S}$, which is updated whenever $\hat{S}$ changes. We show the following invariance: Whenever we start some generic iteration of the loop in line 4, where we try to add i to the index set, then $P[t_s,t_{i-1}] \subset \mathrm{D}_{\Delta+\varepsilon}(p_s)$.

At the start of the first iteration, $\hat{S} = (1)$ and $i = 2$. As $P[t_1,t_{i-1}] = P[t_1,t_1] = p_1$, the invariance holds.

Now assume we are at the start of some iteration, where we try to add i to $\hat{S}$, and assume the invariance holds. Assume for now that during the current iteration we do not enter line 7. In particular, when arriving at line 9, the invariance still holds. We either enter line 11, in which case at the start of the next iteration $i+1$ we have that $s = i$ and the invariance holds, or we do not enter line 11, in which case both $P[t_s,t_{i-1}]$ and $P(t_i)$ and hence also $P[t_s,t_i]$ are in $\mathrm{D}_{\Delta+\varepsilon}(t_s)$ and thus the invariance also holds at the start of the next iteration.

Hence let us instead assume that we do enter line 7. Thus, when arriving at line 9 we have that $\mathrm{d}_\mathcal{F}(P[t_s,t_i],\overline{p_s\,p_i}) \leq \Delta$. We again either enter line 11, in which case at the start of the next iteration $i+1$ we have that $s = i$ and the invariance holds, or we do

not enter line 11. In the latter case we have that

$$\mathrm{d}_{\mathcal{F}}(P[t_s, t_i], \overline{p_s\, p_s}) \leq \mathrm{d}_{\mathcal{F}}(P[t_s, t_i], \overline{p_s\, p_i}) + \mathrm{d}_{\mathcal{F}}(\overline{p_s\, p_i}, \overline{p_s\, p_s}) \leq \Delta + \varepsilon,$$

and hence the invariance holds at the start of the next iteration.

We observe that whenever i is added to $\hat{S}$ in iteration i, we have that either the invariance holds, or $\hat{S}$ was modified between the start of the iteration and the arrival at line 9. The latter implies that $\mathrm{d}_{\mathcal{F}}(P[t_s, t_i], \overline{p_s\, p_i}) \leq \Delta$ and thus in either case $\mathrm{d}_{\mathcal{F}}(P[t_s, t_i], \overline{p_s\, p_i}) \leq \Delta + \varepsilon$, implying that the resulting simplification has property (ii).

Property (iii) similarly is an immediate consequence of the invariance. First observe that $P[t_1, t_{i_1}] = P[t_1, t_1]$ and thus $\mathrm{d}_{\mathcal{F}}(P[t_1, t_{i_1}], \overline{p_{i_1}\, p_{i_1}}) = 0 \leq \Delta + \varepsilon$. Second observe that at the end of iteration n the invariance holds, and hence $\mathrm{d}_{\mathcal{F}}(P[t_s, t_n], \overline{p_s\, p_n}) \leq \Delta + \varepsilon$.

Lastly, for property (iv) observe that, if $\mathrm{d}_{\mathcal{F}}(P[t_{i_j}, t_{i_{j+2}}], \overline{p_{i_j}\, p_{i_{j+2}}}) < \Delta$ for some i_j and i_{j+2} in the resulting index set I, the algorithm would have removed i_{j+1} from I in Line 6, as when we add i_{j+2}, $\hat{S}.\mathrm{next\_to\_top}() = i_j$.

The space requirement follows as we only store S, $\hat{S}$, and possibly the Δ-free space of a subcurve of P with a single edge. For analyzing the running time, note that each vertex of P is inserted to and removed from the index set at most once. Therefore, the total running time is bounded by the at most $\mathcal{O}(n)$ checks of whether the Fréchet distance between a polygonal curve of complexity n and an edge is at most Δ. This can be computed in $\mathcal{O}(n)$ time concluding the proof. □

If instead $\Delta = 0$, the algorithm has linear running time. In this case, the computed simplification coincides exactly with the simplification introduced in [DHW12].

Theorem 4.4.5. *Let P be a polygonal curve in $\mathbb{R}^d$. Let $\varepsilon \geq 0$ be given. There exists an algorithm that outputs an index set defining a $(0, \varepsilon)$-maximal simplification of P. Furthermore, it does so in $\mathcal{O}(n)$ time.*

Proof. Consider Algorithm 3 and Theorem 4.4.4. If $\Delta = 0$, each loop takes $\mathcal{O}(1)$ time concluding the proof. □

4.4.2 The Algorithm for General Curves

We now present a bicriteria approximation algorithm for the SC problem.

Lemma 4.4.6 ([vdHvdHO25])**.** *Let $\pi \in \mathbb{X}^{d,\ell}$ and $P \in \mathbb{X}^{d,n}$ be two curves and let Δ be given. Then $\mathrm{Cov}_P(\pi, \Delta)$ consists of at most $\mathcal{O}(n)$ continuous closed disjoint intervals. These intervals can be computed in $\mathcal{O}(n\ell \log^2 n)$ time.*

Theorem 4.4.7. *Algorithm 4 is a $(24\ln(n)+32, 4)$-approximation for SC. Given a polygonal curve P of complexity n, $\Delta \geq 0$ and $\ell \leq n$, its running time is in $\mathcal{O}\left(k_\Delta n^3 \log^3 n\right)$, where k_Δ is the size of the smallest subset $C^* \subset \mathbb{X}^d_\ell$ such that $\mathrm{Cov}_P(C^*, \Delta) = [0, 1]$.*

Proof. Theorem 4.3.15, Theorem 4.4.3, and Observation 4.4.2 imply that Algorithm 4 is a $(24\ln(n) + 32, 4)$-approximation for SC. The running time is clearly dominated by computing $\mathrm{Cov}_P(c, 4\Delta)$, A and computing $|A \cap \mathrm{Cov}_P(c, 4\Delta)|$ for every c in $\mathcal{C}_S(E)$ in every round. By Lemma 4.4.6, computing the $\mathcal{O}(n)$ disjoint intervals representing $\mathrm{Cov}_P(c, 4\Delta)$ takes $\mathcal{O}(n \log^2 n)$ time for Type (II)- and (III)-subcurves and $\mathcal{O}(n\ell \log^2 n)$ for Type (I)-subcurves. Since there are $\mathcal{O}(n \log n)$ Type (I)-subcurves and $\mathcal{O}(n^2 \log n)$ Type (II)- and (III)-subcurves, computing and storing $\mathrm{Cov}_P(c, 4\Delta)$ for every c in $\mathcal{C}_S(E)$ takes

Algorithm 4 Subtrajectory Covering

procedure ComputeSubtrajectoryCovering(P,Δ)
  Compute $(2\Delta, 0)$-maximal simplification S of P
  Compute $\mathcal{D}_{4\Delta}(S, P)$ and with it $\mathcal{E}(\mathcal{D}_{4\Delta}(S, P))$ and $\mathcal{A}(\mathcal{D}_{4\Delta}(S, P))$
  Compute the set E of y-coordinates of $\mathcal{E}(\mathcal{D}_{4\Delta}(S, P))$
  Compute the set $\mathcal{C}_S(E)$ of Type (I)-, $(II)-$ and (III)-subcurves of S
  Compute and store $\mathrm{Cov}_P(c, 4\Delta)$ for every $c \in \mathcal{C}_S(E)$
  Store the midpoints A of atomic intervals $\mathcal{A}(\mathcal{D}_{4\Delta}(S, P))$ in a balanced binary tree
  $R \leftarrow \emptyset$
  while $A \neq \emptyset$ **do**
    Identify $c^* \in \mathcal{C}_S(E)$ such that $|A \cap \mathrm{Cov}_P(c^*, 4\Delta)|$ is maximum
    $R \leftarrow R \cup \{c^*\}$, $A \leftarrow A \setminus \mathrm{Cov}_P(c^*, 4\Delta)$
  return R

$\mathcal{O}(n^3 \log^3 n)$ time. The set A is computed via the $\mathcal{O}(n^3)$ intersection points of $\mathcal{O}(n^2)$ horizontal lines with the free space and thus computed in $\mathcal{O}(n^3 \log n)$ time. Computing $|A \cap \mathrm{Cov}_P(c, 4\Delta)|$ for c in $\mathcal{C}_S(E)$ takes $\mathcal{O}(n \log n)$ time by storing A in a binary tree, where each inner node is augmented with the minimal interval containing all elements in A stored in the subtree rooted at that node as well as the number of leaves in that subtree. By Theorem 4.4.3, the algorithm terminates after a total of $\mathcal{O}(k_\Delta \log n)$ iterations and hence the total running time is bounded by $\mathcal{O}(k_\Delta n^3 \log^3 n)$. □

4.4.3 Improvements for c-Packed Curves

If the input curve is a c-packed polygonal curve, we can obtain results that depend on c. The first two lemmas and proofs of this section are reminiscent of Lemma 4.2 and Lemma 4.3 in [DHW12]. The main difference is the definition of the simplifications used. The third lemma uses the second lemma to show that given a c-packed curve P, the number of extremal points in the $(4 + 2\varepsilon)\Delta$-free space of a $(2\Delta, \varepsilon\Delta)$-maximal and $(0, \varepsilon\Delta)$-maximal simplification of P is in $\mathcal{O}(nc\varepsilon^{-2})$ for c-packed curves improving upon the naïve bound of $\mathcal{O}(n^2)$ in the worst-case.

Lemma 4.4.8. *Let P be a curve in $\mathbb{R}^d$. Let S be a simplification of P with $\mathrm{d}_{\mathcal{F}}(P, S) \leq \Delta$. Then for any ball $\mathrm{D}_r(p)$ it holds that the total arc length $\|S \cap \mathrm{D}_r(p)\|$ of S in $\mathrm{D}_r(p)$ is at most $\|P \cap \mathrm{D}_{r+\Delta}(p)\|$.*

Proof. Consider any segment u of the simplification S that intersects $\mathrm{D}_r(p)$, defined by vertices p_i and p_j of P, with $P(t_i) = p_i$ and $P(t_j) = p_j$, and let $v = u \cap \mathrm{D}_r(p)$. Observe that $P[t_i, t_j]$ lies inside a capsule of radius Δ around u. Now erect two hyperplanes passing through the endpoints of v. $P[t_i, t_j]$ must intersect both, hence the length of $P[t_i, t_j]$ inside a capsule of radius Δ around v is at least $||v||$. As this capsule lies completely inside $\mathrm{D}_{r+\Delta}(p)$, the claim follows. □

Lemma 4.4.9. *Let P be a c-packed curve, and let $\varepsilon \in (0, 1]$, $\Delta > 0$ and S a simplification of P such that any edge of S has length at least Δ/ε and $\mathrm{d}_{\mathcal{F}}(S, P) \leq (2 + \varepsilon)\Delta$ be given. Then S is $18c/\varepsilon$-packed.*

Proof. Let $\mu = \mathrm{d}_{\mathcal{F}}(P, S)$ and observe that $\mu \leq 3\Delta$. Assume for the sake of contradiction that the arc length $\|S \cap \mathrm{D}_r(p)\|$ of S inside some disk $\mathrm{D}_r(p)$ is larger than $18cr/\varepsilon$. If $r \geq \mu$ let $r' = 2r$. Then by Lemma 4.4.8 together with our assumption

$$||P \cap \mathrm{D}_{r'}(p)|| \geq ||P \cap \mathrm{D}_{r+\mu}(p)|| \geq ||S \cap \mathrm{D}_r(p)|| > 18cr/\varepsilon > 9cr'/\varepsilon > 9cr'.$$

This contradicts the fact that P is c-packed. If $r < \mu$ let U denote the segments of S intersecting $\mathrm{D}_r(p)$ and let $k = |U|$. Observe that $k > (18cr/\varepsilon)/2r = 9c/\varepsilon$ as any segment can contribute at most $2r$ to the length of S inside $\mathrm{D}_r(p)$.

$$\begin{aligned} ||S \cap \mathrm{D}_{2\mu}(p)|| &\geq ||S \cap \mathrm{D}_{r+\mu}(p)|| \geq ||U \cap \mathrm{D}_{r+\mu}(p)|| \geq k \min(\mu, \varepsilon\Delta) \\ &> 9c/\varepsilon \min(\varepsilon\Delta, \mu) = 3c \min(3\Delta, 3\mu/\varepsilon) \geq 3c \min(\mu, \mu) = 3c\mu, \end{aligned}$$

since every segment of B has minimal length $\varepsilon\Delta$. Hence by Lemma 4.4.8

$$||P \cap \mathrm{D}_{3\mu}(p)|| \geq ||S \cap \mathrm{D}_{2\mu}(p)|| > 3c\mu,$$

again contradicting the fact that P is c-packed. □

Lemma 4.4.10. *Let P be a c-packed polygonal curve. Let Δ and ε be given. Let S be a $(2\Delta, \varepsilon\Delta)$-maximal simplification of P and let P' be a $(0, \varepsilon\Delta)$-maximal simplification of P. Then $|\mathcal{E}(\mathcal{D}_{4\Delta+2\varepsilon}(S, P'))| = \mathcal{O}(cn/\varepsilon^2)$.*

Proof. This follow the proof of [DHW12, Lemma 4.4]. There are at most 8 extremal points for every non-empty cell in $\mathcal{D}_{4\Delta+2\varepsilon}(S, P')$. A cell defined by edges u of S and v of P' is non-empty if and only if there is a point p on u and q on v such that $\|p - q\| \leq (4 + 2\varepsilon)\Delta$. Charge any such pair of edges to the shorter of the two edges.

Now consider an edge u of S and in particular the ball of radius $r = 3/2\|u\| + (4+2\varepsilon)\Delta$ centered at the midpoint m of u. Every segment v of P' participating in a pair with u which is charged to u must intersect the ball $\mathrm{D}_{\|u\|/2+(4+2\varepsilon)\Delta}(m)$. Hence, as $\|v\| > \|u\|$, the arc length of v in the ball $\mathrm{D}_r(m)$ must be at least $\|u\|$. By Lemma 4.4.9, both P' and S are $18c/\varepsilon$-packed. Hence we have that the number of charges to u is at most

$$\frac{\|P' \cap \mathrm{D}_r(m)\|}{\|u\|} \leq \frac{18c/\varepsilon(3/2\|u\| + (4 + 2\varepsilon)\Delta)}{\|u\|} \leq 27c/\varepsilon + \frac{18c/\varepsilon(4 + 2\varepsilon)\Delta}{\varepsilon\Delta} = \mathcal{O}\left(c/\varepsilon^2\right).$$

Similarly, if u is an edge of P' instead. Thus the total charges to all edges is at most $\mathcal{O}(nc/\varepsilon^2)$. □

Theorem 4.4.11. *Let $\varepsilon \in (0, 1]$ be given. There is a $(24\ln(n)+32, 4+3\varepsilon)$-approximation for SC. Given a c-packed polygonal curve P of complexity n in $\mathbb{R}^d$, $\Delta > 0$ and $\ell \leq n$, its running time is in $\mathcal{O}\left(n^2\ell \log^2 n + k_\Delta cn^2\varepsilon^{-2}\log^3 n\right)$, where k_Δ is the size of the smallest subset $C^* \subset \mathbb{X}_\ell^d$ such that $\mathrm{Cov}_P(C^*, \Delta) = [0, 1]$.*

Proof. The proof follows the proof of Theorem 4.4.7. The algorithm starts by computing a $(2\Delta, \varepsilon\Delta)$-maximal simplification S of P and a $(0, \varepsilon\Delta)$-maximal simplification P' of P. Then it executes line 3 to line 12 of Algorithm 4 with $S \leftarrow S$, $P \leftarrow P'$ and $\Delta \leftarrow (1 + \varepsilon/2)\Delta$. The correctness follows from Theorem 4.3.15, Theorem 4.4.3, and Observation 4.4.2. The running time is clearly dominated by computing $\mathrm{Cov}_{P'}(c, (4+2\varepsilon)\Delta)$, A and computing $|A \cap \mathrm{Cov}_{P'}(c, (4 + 2\varepsilon)\Delta)|$ for every c in $\mathcal{C}_S(E)$ in every round. By Lemma 4.4.6, computing the $\mathcal{O}(n)$ disjoint intervals representing $\mathrm{Cov}_{P'}(c, (4 + 2\varepsilon)\Delta)$ takes $\mathcal{O}(n \log^2 n)$ time for Type (II)- and (III)-subcurves and $\mathcal{O}(n\ell \log^2 n)$ for Type

(I)-subcurves. Since there are $\mathcal{O}(n \log n)$ Type (I)-subcurves and $\mathcal{O}(nc\varepsilon^{-2} \log n)$ Type (II)- and (III)-subcurves by Lemma 4.4.10, computing and storing $\mathrm{Cov}_{P'}(c, (4+2\varepsilon)\Delta)$ for every c in $\mathcal{C}_S(E)$ takes $\mathcal{O}(n^2 c\varepsilon^{-2} \log^3 n)$ time. Observe that by Lemma 4.4.10, the cardinality $|A|$ is at most $\mathcal{O}(\min(n^3, n^2 c\varepsilon^{-2}))$. Hence the set A can be computed in $\mathcal{O}(n^2 c\varepsilon^{-2} \log n)$ time. Computing $|A \cap \mathrm{Cov}_P(c, 4\Delta)|$ for c in $\mathcal{C}_S(E)$ takes $\mathcal{O}(n \log n)$ time. By Theorem 4.4.3 the algorithm terminates after a total of $\mathcal{O}(k_\Delta \log n)$ iterations and hence the total running time is bounded by $\mathcal{O}(n^2 \ell \log^2 n + k_\Delta c n^2 \varepsilon^{-2} \log^3 n)$. □

4.5 Experimental Evaluation

So far, we have established a bicriteria approximation algorithm for the SC problem. We have observed that in theory this algorithm scales well with realistic (i.e., c-packed for small/constant c) input curves. The approximation guarantees and constants hidden in the asymptotic notation may, however, make the algorithm not practically viable. In this section we give empirical evidence that this is *not* the case, that is, the algorithm performs well in practice. All experiments were conducted on a Linux system with 16GB of memory with an Intel `i5-9600` CPU.

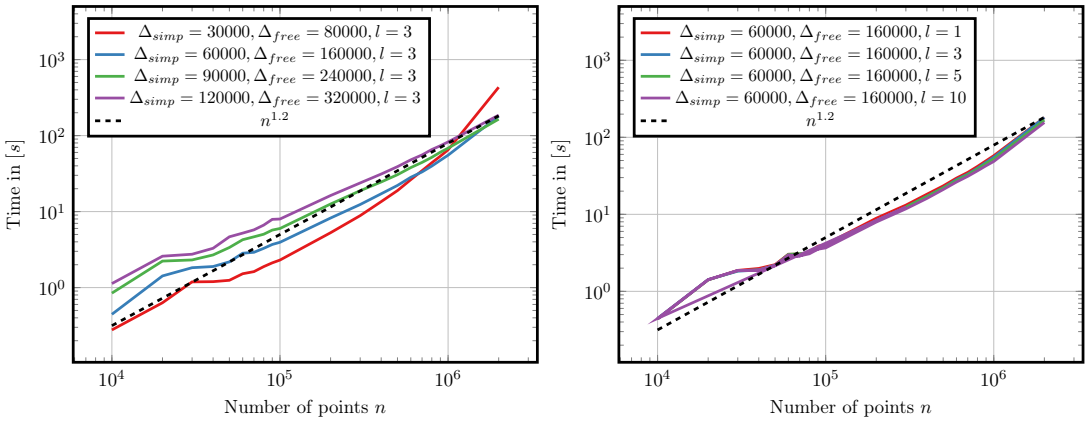
4.5.1 Implementation Details

Our `C++`-implementation is available online[2]. The input of the algorithm consists of a set $\{P_1, \ldots, P_m\}$ of curves and three parameters Δ_{simp}, Δ_{free}, and ℓ. It first computes $(2/3\Delta_{simp}, 1/3\Delta_{simp})$-maximal simplifications S_i of P_i for all i with parameter Δ_{simp}, and then computes the Δ_{free}-free space of S_i and S_j as well as their extremal points for all pairs $(i, j) \in [m]^2$. Next it computes all pairs of extremal coordinates that (i) lie on the same curve, (ii) the resulting subcurve has complexity at most ℓ and (iii) there is a power of 2 of other extremal coordinates in the interval between them (c.f. Definition 4.3.9 and [vdHvdHO25]). For every such pair of extremal coordinates (a, b) on curve S_i we compute the Δ_{free}-coverage of $S_i[a, b]$ via the computed Δ_{free}-spaces of S_i with any other S_j. This Δ_{free}-coverage is a subset of $[m] \times [0, 1]$ and defines a set cover instance. This set cover instance we solve via a greedy set cover algorithm that iteratively picks and adds the candidate which maximizes the arc length (computed on each S_i) of the additional coverage. This step is repeated until $[m] \times [0, 1]$ is covered. This resembles solving the SCM problem rather than the SC problem without a cardinality constraint k. However, it allows us to stop the greedy algorithm after a small number of rounds and still have a partial solution that covers a large fraction of the input. We introduced the two parameters Δ_{simp} and Δ_{free} to test the stability of the threshold parameter Δ in both the simplification and free-space computation step. Given some Δ, setting $\Delta_{simp} = 3\Delta$ and $\Delta_{free} = 8\Delta$ yields bounds on the running time comparable to Theorem 4.4.11.

4.5.2 Ocean Drifters

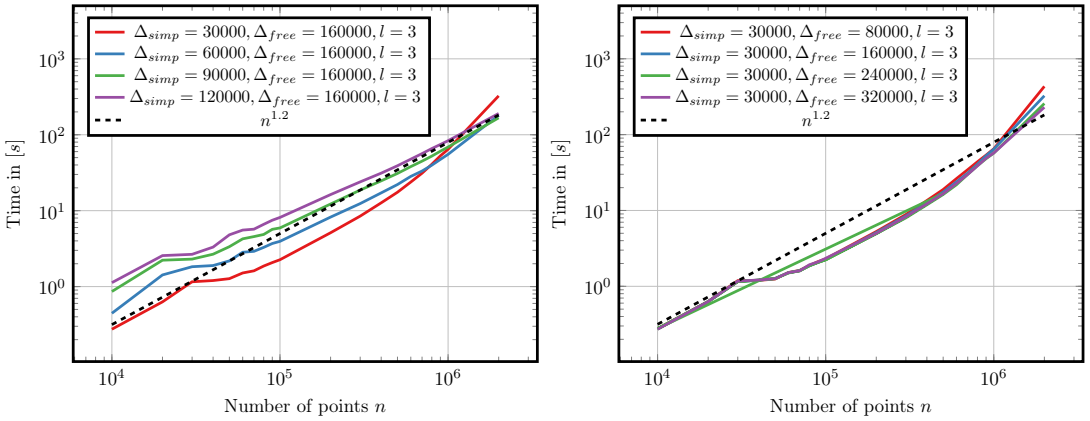
We evaluate our implementation to trajectories from the NOAA Global Drifter Program [LC19]. This is a comprehensive data set consisting of almost 20 000 ocean surface drifters that have been released across the ocean as far back as 1979. For the evaluation, we focus on the subset of trajectories consisting of all drifters recorded in the years

[2] `archive.softwareheritage.org/swh:1:dir:a5910d607f46de40d9b7f271f9156e5a048cc351`



(a) varying Δ_{simp} and Δ_{free} with $3:8$ ratio.

(b) varying l.



(c) varying Δ_{simp}.

(d) varying Δ_{free}.

Figure 4.4: Influence of different combinations of parameters on the running time evaluated on the data set from the NOAA Global Drifter Program [LC19].

2022–2024. This subset consist of 5500 different trajectories which consist on average of 500 points resulting in a total input complexity of $n \geq 2 \cdot 10^6$. Refer to Figure 1.1 on page 10, in which the data set and the computed set of center curves with $\Delta_{simp} = 20\text{km}$, $\Delta_{free} = 400\text{km}$ and $\ell = 20$ is depicted.

Evaluation

We apply our techniques with a range of radii with Δ_{simp} between 5km and 120km, Δ_{free} between 10km and 320km, as well as a range of complexity bounds with ℓ between 1 and 10. Figure 4.4 shows the running times. We observe that the running time appears to be mostly independent of the exact values of Δ_{simp} and Δ_{free}, and scales favorably in n compared to the theoretical results. In the $n \leq 10^5$ regime, the running time of the algorithm appears to scale near-linearly, with an observed running time of approximately $\mathcal{O}(n^{1.2})$. As n increases, the observed running time approaches $\mathcal{O}(n^2)$, compared to the theoretical running time of $\mathcal{O}(k_\Delta n^3 \log^3 n)$ and $\mathcal{O}(k_\Delta c n^2 \log^3 n)$. In addition, we empirically evaluate the approximation ratio of our set cover algorithm using the size of a greedily computed independent set as a lower bound. We observe that for all tested instances the approximation ratio is less than 3 compared to $24 \ln n + 32 \approx 350$.

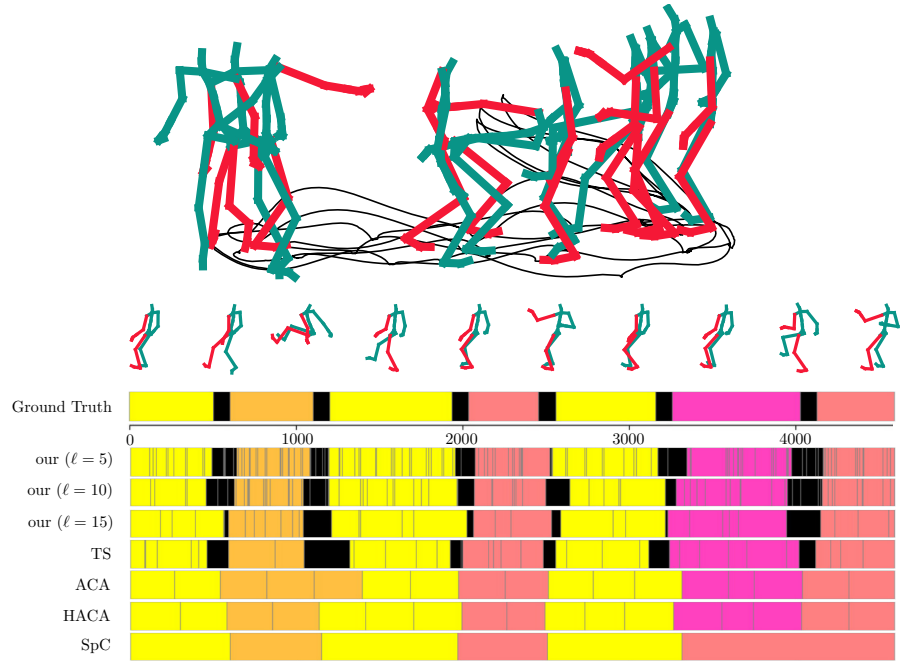


Figure 4.5: Trial `01` of subject `86` with its ground truth labels. Colors correspond to the labels `walk` (yellow), `jump` (orange), `punch` (light red), `kick` (magenta) and `transition` (black). Underneath, the resulting labeling using our techniques with different values for ℓ, temporal segmentation (TS), (hierarchical) aligned cluster algorithm (HACA/ACA) and spectral clustering (SC) [KVW+17, ZDlTH12, ZDlTH08], with gray lines corresponding to the start/end of the coverage of some computed center.

4.5.3 Full-Body Motion Tracking Data

Motion segmentation finds applications in many different fields, such as robotics, sports analysis or traffic monitoring [MGAM20]. We apply our techniques to this problem on the CMU data set [MoC07]. This data set consists of motion tracking data of 31 trackers attached to the joints of subjects doing physical activities (refer to Figure 4.5). Each frame of such a motion capture sequence is labeled with one of 15 semantic labels, such as `walk`, `jump`, or `punch` [KVW+17]. We treat such a sequence of frames as a curve in 93-dimensional Euclidean space by concatenating the three-dimensional coordinates of all joints back to back to form a pose. Each trial consists of up to 10^4 poses. We apply our bicriteria approximation algorithm for the SC problem with $\Delta_{simp} = 0.8$ and $\Delta_{free} \approx 1.35$ and a complexity bound ℓ between 5 and 15, where the exact parameters have been identified via an exhaustive search to yield the best accuracy for the given complexity bound ℓ. The output consists of a set of curves that act as cluster centers. For each center, we identify the ground truth label that best corresponds to it and label all points in its Δ_{free}-coverage with the identified label. Whenever different labels are assigned to a point along the curve we mark it as a transition between motion labels.

Evaluation

Figure 4.5 shows the resulting labeling. Observe in particular that an increase in ℓ decreases the number of patterns identified with the total number of labeled segments ap-

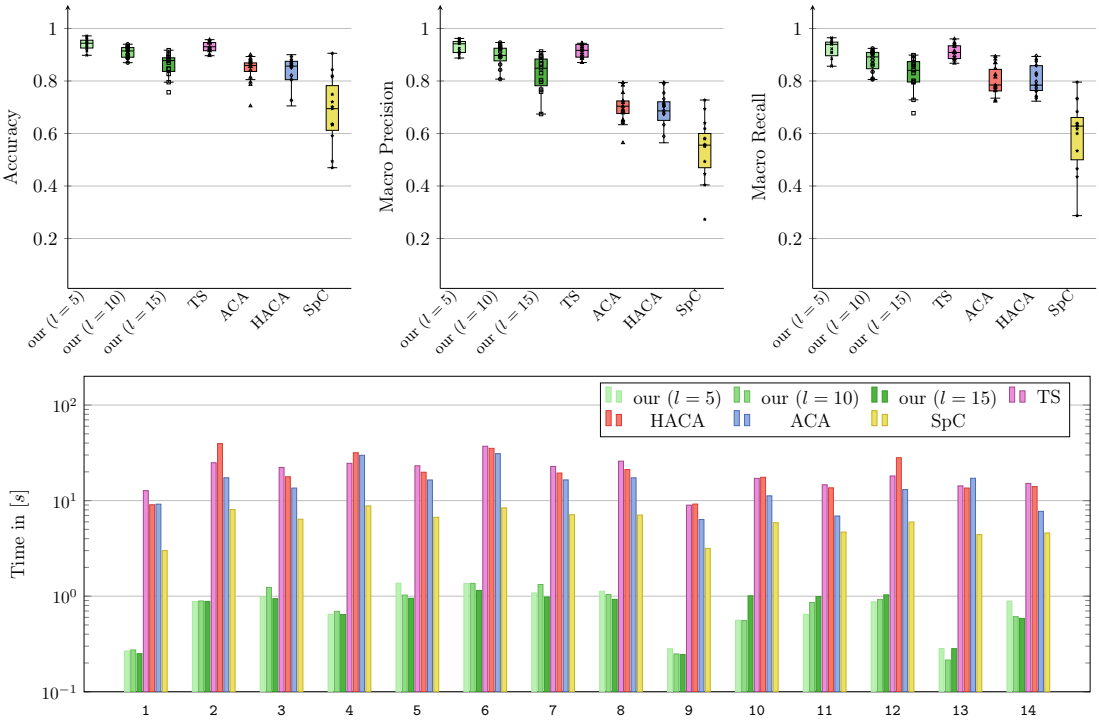


Figure 4.6: Quantative analysis on trial `1` to `14` of subject `86` from [MoC07] and comparison of our techniques to temporal segmentation (TS), (hierarchical) aligned cluster algorithm (HACA/ACA), and spectral clustering (SpC) from [KVW+17, ZDlTH12, ZDlTH08].

proaching that of the state-of-the-art, while the accuracy decreases only slightly. We compute the accuracy of the resulting segmentations on ground truth data from [KVW+17] and compare this accuracy with the accuracy of the temporal segmentation approach (TS) discussed in [KVW+17] as well as the aligned cluster algorithm (ACA), hierarchical aligned cluster algorithm (HACA), and spectral clustering (SpC) discussed in [ZDlTH12, ZDlTH08]. Figure 4.6 shows the resulting accuracies. The quantitative accuracy of our techniques compares well to the state-of-the-art techniques, with a (roughly) tenfold improvement in the running time.

In addition to the comparable accuracy and the improved running time, we also want to highlight the small number of parameters—namely ℓ, Δ_{free} and Δ_{simp}—as compared to some of the other approaches, such as HACA, with up to twelve tunable parameters [ZDlTH12], or TS with a variety of parameters ranging from sub-sampling frequencies, to different radii and importance cut-off values for feature selection [KVW+17].

4.6 Structuring the Solution Space

We now return to the theoretical analysis of the range space

$$([0,1], \{\mathrm{Cov}_A(c) \subset [0,1] \mid c \in \mathcal{C}_S(E)\}) .$$

The goal of this section, as well as Section 4.7, and Section 4.8 is the improvement of the running time of the greedy set cover algorithm by improving the running time of the coverage computation in each round.

4.6.1 Sweep-Sequences

In this section we identify an ordering of the Type (II)- and Type (III)-subcurves from Theorem 4.3.15 which allows maintaining a symbolic representation of (an approximation of) the coverage, as well as construction of a data structure that allows efficient point-queries returning the set of all Type (II)- and Type (III)-subcurves whose coverage includes the query point.

Throughout this section we are given a polygonal curve $\hat{P}$ of complexity n, values $\hat{\Delta}$, ε, $\alpha \geq 1$, and ℓ. Let S be a $(2\hat{\Delta}, \varepsilon\hat{\Delta})$-maximal simplification and P a $(0, \varepsilon\hat{\Delta})$-maximal simplification of $\hat{P}$. Let $\Delta = (4 + 2\varepsilon\hat{\Delta})$. Let $\mathcal{D}$ be an (α, Δ)-free space of S and P and let $\mathcal{D}'$ be the (α, Δ)-free space of $\text{rev}(S)$ and P resulting from $\mathcal{D}$ by mirroring it along the x-axis. Let E be a finite sorted superset of the set of y-coordinates of the extremal points $\mathcal{E}(\mathcal{D})$ of $\mathcal{D}$. Recall that E_e denotes the restriction of E onto the edge e of S. For ease of exposition we assume that for every edge e the set E_e has cardinality in $\mathcal{O}(n)$ and thus $|E| = \mathcal{O}(n^2)$ and $|\mathcal{C}_S(E)| = \mathcal{O}(n^2 \log n)$. We will additionally assume that for every edge e, the functions $l_i(\cdot)$ and $r_i(\cdot)$ of the free space $\mathcal{D}$ can be computed in $\mathcal{O}(1)$ time.

Definition 4.6.1 (Sweep-sequence)**.** Let $E = (e_1, \ldots)$ be a sorted list of values in $\mathbb{R}$. We say $\mathfrak{s}$ is a sweep-sequence of E if $\mathfrak{s}$ is a list of tuples of E such that either (i) for all consecutive tuples (e_a, e_b) and (e_c, e_d) it holds that $a \leq b$, $c \leq d$, $0 \leq a - c \leq 1$ and $0 \leq b - d \leq 1$ or (ii) for all consecutive tuples it holds that $a \geq b$, $c \geq d$, $0 \leq c - a \leq 1$ and $0 \leq d - b \leq 1$.

Lemma 4.6.2. *Let e be an edge of S. There is a set $\mathfrak{S}_e$ of $\mathcal{O}(\log n)$ sweep-sequences of E_e, each of length $\mathcal{O}(n)$, such that for any Type (II)- or Type (III)-subcurve π on the edge e there is a tuple $(\varepsilon_i, \varepsilon_j)$ in one of the sweep-sequences in $\mathfrak{S}_e$ such that $\pi = e[\varepsilon_i, \varepsilon_j]$ if $\varepsilon_i \leq \varepsilon_j$, and $\pi = \text{rev}(e[\varepsilon_j, \varepsilon_i])$ if $\varepsilon_i > \varepsilon_j$. Further, for the first and last pair (e_a, e_b) in every sweep-sequence it holds that $a = b$.*

Proof. Let $E_e = \{\varepsilon_1, \ldots, \varepsilon_m\}$. We construct one sweep-sequence for the Type (II)-subcurves, and $\mathcal{O}(\log n)$ sweep-sequences for the Type (III)-subcurves.

For the Type (II)-subcurves, the sought-after sweep-sequence is precisely the sequence $\{(\varepsilon_1, \varepsilon_1), (\varepsilon_1, \varepsilon_2), \ldots, (\varepsilon_1, \varepsilon_{m-1}), (\varepsilon_1, \varepsilon_m), (\varepsilon_2, \varepsilon_m), \ldots, (\varepsilon_m, \varepsilon_m)\}$.

For the Type (III)-subcurves we construct two sweep-sequences for any $j \in \{2^m \mid 0 \leq m \leq \lfloor \log_2(n) \rfloor\}$, namely the sequences $\{(\varepsilon_1, \varepsilon_{1+2^j}), (\varepsilon_2, \varepsilon_{2+2^j}), \ldots, (\varepsilon_{m-2^j}, \varepsilon_m)\}$ and $\{(\varepsilon_{1+2^j}, \varepsilon_1), (\varepsilon_{2+2^j}, \varepsilon_2), \ldots, (\varepsilon_m, \varepsilon_{m-2^j})\}$. To satisfy the requirement that the first and last pair consists of two copies of the same value, we append and prepend the sequence $\{(\varepsilon_1, \varepsilon_1), (\varepsilon_1, \varepsilon_2), \ldots, (\varepsilon_1, \varepsilon_{2^j})\}$ (resp. $\{(\varepsilon_1, \varepsilon_1), (\varepsilon_2, \varepsilon_1), \ldots, (\varepsilon_{2^j}, \varepsilon_1)\}$) and append the sequence $\{(\varepsilon_{m-2^j+1}, \varepsilon_m), \ldots, (\varepsilon_m, \varepsilon_m)\}$ (resp. $\{(\varepsilon_m, \varepsilon_{m-2^j+1}), \ldots, (\varepsilon_m, \varepsilon_m)\}$) (refer to Figure 4.7).

The claim then follows by definition of Type (II)- and (III)-subcurves as well as the fact that $m = |E_e| = \mathcal{O}(n)$. □

Observe that for any edge e of S these sweep-sequences $\mathfrak{S}_e$ can be constructed in $\mathcal{O}(n \log n)$ time.

In the remainder of this section we focus our analysis on sweep-sequences where for every tuple $(\varepsilon_a, \varepsilon_b)$ it holds that $a \leq b$. Any analysis of such sweep-sequences carries over to sweep-sequences where for every tuple $(\varepsilon_a, \varepsilon_b)$ it holds that $a > b$, by setting $S \leftarrow \text{rev}(S)$, and thus to all sweep-sequences constructed in Lemma 4.6.2.

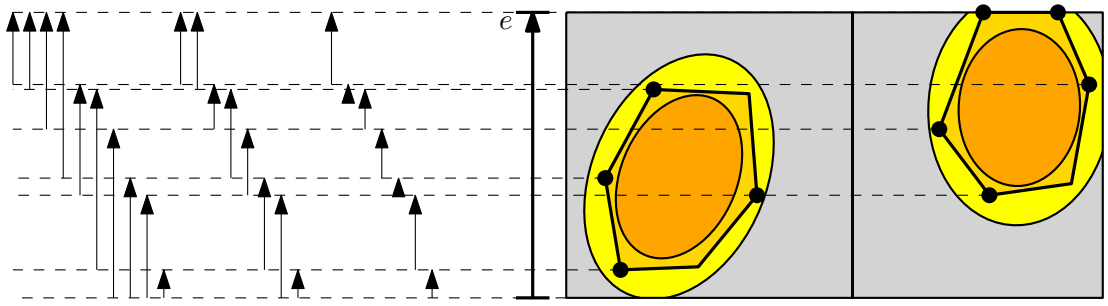


Figure 4.7: Illustration of three of the eight sweep-sequences in $\mathfrak{S}_e$ of the edge e that are constructed for Type (III)-subcurves. One for $j = 1$, one for $j = 2$, and one for $j = 4$.

4.6.2 Combinatorial Representation and the Proxy Coverage

In this section we define an approximation to the notion of free space coverage as well as an underlying structure, the *reduced combinatorial representation*, which we later on show how to maintain efficiently. An important consequence is the definition of the *proxy coverage* (Definition 4.6.7) which allows efficient maintenance along a sweep-sequence.

Definition 4.6.3 (Combinatorial Representation)**.** Let e be an edge of S and let $0 \leq s \leq t \leq 1$ be given. The combinatorial representation $\mathcal{R}(e[s,t])$ of the coverage $\mathrm{Cov}_{\mathcal{D}}(e[s,t])$ of $e[s,t]$ is the set of all inclusionwise maximal pairs of indices (i,j), such that there are points s' and t' on the i^{th} and j^{th} edge of P respectively and a monotone path from (s',s) to (t',t) in $\mathcal{D}$. A pair of indices (i,j) includes another pair (i',j') if $i \leq i'$ and $j' \leq j$.

The combinatorial representation separates into two sets, the global group $\mathcal{G}(e[s,t])$ consisting of all index pairs $(i,j) \in \mathcal{R}(e[s,t])$ such that $i < j$ and the local group $\mathcal{L}(e[s,t])$ consisting of all index pairs $(i,j) \in \mathcal{R}(e[s,t])$ with $i = j$.

Observation 4.6.4. *Let $e[s,t]$ be a subedge of an edge of S and let P be given. Then*

$$\begin{aligned}\mathrm{Cov}_{\mathcal{D}}(e[s,t]) &= \bigcup_{(i,j)\in\mathcal{R}(e[s,t])} [l_i(s), r_j(t)] \\ &= \left(\bigcup_{(i,j)\in\mathcal{G}(e[s,t])} [l_i(s), r_j(t)] \right) \cup \left(\bigcup_{(i,i)\in\mathcal{L}(e[s,t])} [l_i(s), r_i(t)] \right).\end{aligned}$$

The overall goal is to find a representation of $\mathrm{Cov}_{\mathcal{D}}(e[s,t])$ that is computationally easy to maintain. More precisely, we want to represent $\mathrm{Cov}_{\mathcal{D}}(e[s,t])$ as a *disjoint* union of intervals.

Let an edge e of S together with $0 \leq s \leq t \leq 1$ be given. We say an index i is bad for $e[s,t]$, if all topmost points in cell i of the free space of e and P are to the left of both $l_i(s)$ and $r_i(t)$, and both $l_i(s)$ and $r_i(t)$ are to the left of all bottom most points in cell i. Call this set of bad indices $\mathcal{B}(e[s,t])$. If $i \notin \mathcal{B}(e[s,t])$, i is said to be good for $e[s,t]$. Intuitively, an index is bad if the free space inside the cell corresponding to the index is a 'diagonal' from the top left to the bottom right. For later structural lemmas (Lemma 4.6.9 and Lemma 4.6.10) to hold, the definition of a bad index is slightly stronger and depends on s and t.

Observation 4.6.5. *If i is good for $e[s,t]$ and $l_i(s) \neq \infty \neq r_i(t)$, then $l_i(s) \leq r_i(t)$.*

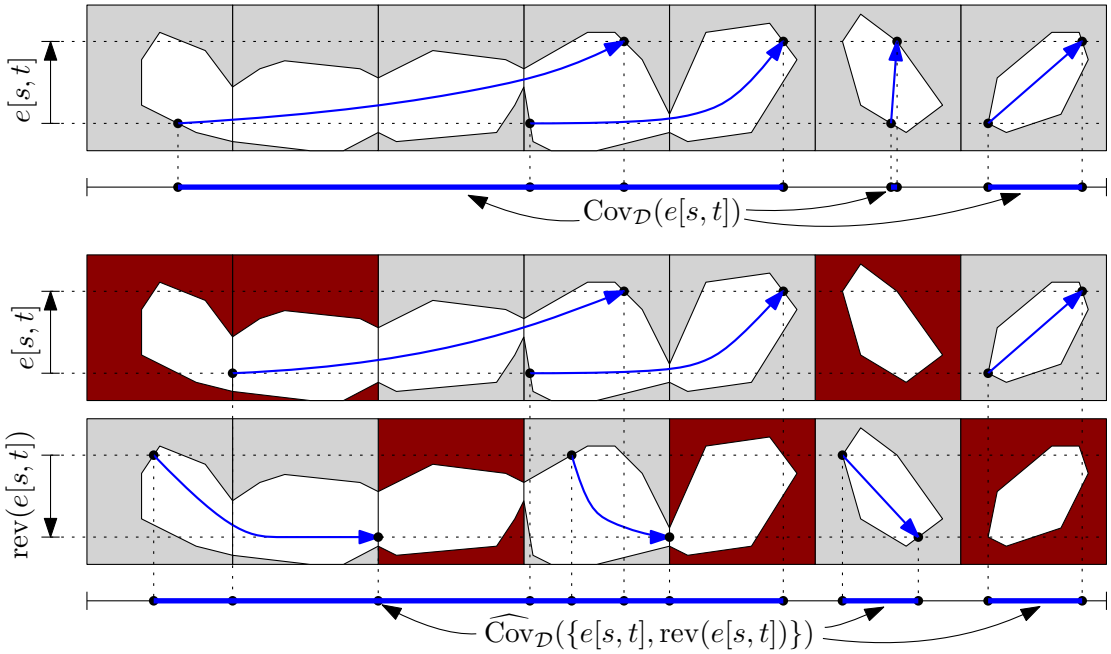


Figure 4.8: Illustration of the proxy coverage of $e[s,t]$ and $\mathrm{rev}(e[s,t])$ compared to the coverage of $e[s,t]$. Cells with bad index are marked red. The global group of $e[s,t]$ is $\{(1,4),(4,5)\}$, and the reduced global group of $e[s,t]$ is $\{(1,5)\}$.

Definition 4.6.6 (Reduced Global Group)**.** Let e be an edge of S and let $0 \leq s \leq t \leq 1$ be given. Based on the global group $\mathcal{G}(e[s,t])$ we define the reduced global group $\widetilde{\mathcal{G}}(e[s,t])$ which results from $\mathcal{G}(e[s,t])$ after merging all pairs of index pairs (a,b) and (c,d) if either $a < c < b < d$, or $b = c$ and $b = c$ is good for $e[s,t]$.

Definition 4.6.7 (Proxy Coverage)**.** For edge e of S and $0 \leq s \leq t \leq 1$ define

$$\hat{l}_{i,e[s,t]}(s) = \begin{cases} l_i(s), & \text{if } i \text{ is good for } e[s,t] \\ r_i(s), & \text{if } i \text{ is bad for } e[s,t] \end{cases} \qquad \hat{r}_{j,e[s,t]}(t) = \begin{cases} l_j(t), & \text{if } j \text{ is good for } e[s,t] \\ r_j(t), & \text{if } j \text{ is bad for } e[s,t]. \end{cases}$$

With these at hand, define the ***proxy coverage*** of subedges of S as the union

$$\widehat{\mathrm{Cov}}_{\mathcal{D}}(e[s,t]) = \left(\bigcup_{i \in \mathcal{L}(e[s,t]) \setminus \mathcal{B}(e[s,t])} [l_i(s), r_i(t)] \right) \cup \left(\bigcup_{(i,j) \in \widetilde{\mathcal{G}}(e[s,t])} [\hat{l}_{i,e[s,t]}(s), \hat{r}_{j,e[s,t]}(t)] \right),$$

where by a slight abuse of notation let $\mathcal{L}(e[s,t]) \setminus \mathcal{B}(e[s,t])$ be all index pairs (i,i) in $\mathcal{L}(e[s,t])$ such that i is not in $\mathcal{B}(e[s,t])$ (refer to Figure 4.8).

Intuitively, the proxy coverage is almost exactly the free space coverage, except for in cells with a bad index b, which the proxy coverage avoids, unless there is a path from some cell $i < b$ to some cell $j > b$. For this refer to Figure 4.8: the index of the second cell of $\mathcal{D}$ in Figure 4.8 is bad for $e[s,t]$ and yet $\widehat{\mathrm{Cov}}_{\mathcal{D}}(e[s,t])$ covers the entirety of the second cell as there is a path from the first to the forth cell. Observe that in certain cases the combinatorial description of the free space coverage of neighboring elements from a

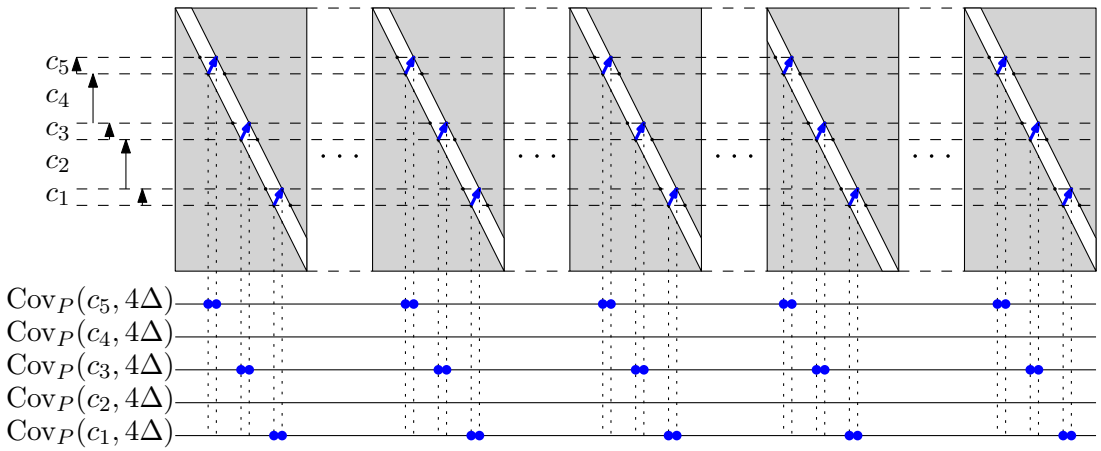


Figure 4.9: Illustration of how the combinatorial description of the free space coverage of two neighboring elements in a sweep-sequence may differ by up to n index pairs. Instead, $\widehat{\mathrm{Cov}}(c_i) = \emptyset$ for all i.

sweep-sequence can greatly differ, while the proxy coverage of neighboring elements in the sweep-sequence does not differ at all (Figure 4.9).

We observe that the proxy coverage can in fact be expressed as a disjoint union via the reduced global group:

Lemma 4.6.8. *Let e be an edge and let $0 \leq s \leq t \leq 1$. Then $\widehat{\mathrm{Cov}}_{\mathcal{D}}(e[s,t])$ coincides with the following disjoint union*

$$\widehat{\mathrm{Cov}}_{\mathcal{D}}(e[s,t]) = \left(\bigsqcup_{(i,i) \in \mathcal{L}(e[s,t]) \setminus \mathcal{B}(e[s,t])} [l_i(s), r_i(t)] \right) \sqcup \left(\bigsqcup_{(i,j) \in \widetilde{\mathcal{G}}(e[s,t])} [\hat{l}_{i,e[s,t]}(s), \hat{r}_{j,e[s,t]}(t)] \right)$$
$$\subset \mathrm{Cov}_{\mathcal{D}}(e[s,t]).$$

Proof. The only way for two index pairs (a,b) and (c,d) among the index pairs in $\mathcal{L}(e[s,t]) \setminus \mathcal{B}(e[s,t])$ and $\widetilde{\mathcal{G}}(e[s,t])$ to have their corresponding intervals overlap is if $b = c$ is bad for $e[s,t]$ and (a,b) and (c,d) are in $\widetilde{\mathcal{G}}(e[s,t])$. But then by definition $[\hat{l}_{a,e[s,t]}(s), \hat{r}_{b,e[s,t]}(t)] = [\hat{l}_{a,e[s,t]}(s), l_b(t)]$ and $[\hat{l}_{c,e[s,t]}(s), \hat{r}_{d,e[s,t]}(t)] = [r_b(s), \hat{r}_{d,e[s,t]}(t)]$. As (a,b) is in the reduced global group, there is a path which starts to the left of cell b and ends in cell b, which implies that the approximate free space in cell b intersects the left boundary and $l_b(s)$ lies to the left of all topmost points in cell b. Similarly $r_b(t)$ lies to the right of all bottommost points. As b is bad, this implies that $l_b(t) < r_b(s)$ and hence $[\hat{l}_{a,e[s,t]}(s), \hat{r}_{b,e[s,t]}(t)]$ and $[\hat{l}_{c,e[s,t]}(s), \hat{r}_{d,e[s,t]}(t)]$ do not intersect, implying the claim that $\widehat{\mathrm{Cov}}_{\mathcal{D}}(\cdot)$ is indeed the described *disjoint* union.

Finally, as $l_i(s) \leq \hat{l}_{i,e[s,t]}(s)$ and $\hat{r}_{i,e[s,t]}(t) \leq r_i(t)$ and hence $[\hat{l}_{i,e[s,t]}(s), \hat{r}_{j,e[s,t]}(t)]$ is a subset of $[l_i(s), r_j(t)]$, it follows that

$$\widehat{\mathrm{Cov}}_{\mathcal{D}}(e[s,t]) \subset \bigcup_{(i,j) \in (\widetilde{\mathcal{G}}(e[s,t])) \cup (\mathcal{L}(e[s,t]) \setminus \mathcal{B}(e[s,t]))} [l_i(s), r_j(t)] \subset \mathrm{Cov}_{\mathcal{D}}(e[s,t]). \qquad \square$$

Next, we show that $\widehat{\mathrm{Cov}}_{\mathcal{D}}$ approximates $\mathrm{Cov}_{\mathcal{D}}$ in the sense that for every element $\pi \in \mathcal{C}_S(E)$ there are two elements $\pi_1, \pi_2 \in \mathcal{C}_S(E)$ such that

$$\mathrm{Cov}_P(\pi, 4\Delta) = \mathrm{Cov}_{\mathcal{D}}(\pi) \subset \widehat{\mathrm{Cov}}_{\mathcal{D}}(\pi_1) \cup \widehat{\mathrm{Cov}}_{\mathcal{D}}(\pi_2).$$

In order to do so, we start by extending the proxy coverage $\widehat{\text{Cov}}_{\mathcal{D}}$ to also be defined for vertex-vertex-subcurves of S, where we simply set $\widehat{\text{Cov}}_{\mathcal{D}}(S[s,t]) = \text{Cov}_{\mathcal{D}}(S[s,t])$, and reversals of subedges of S, where we define $\widehat{\text{Cov}}_{\mathcal{D}}(\text{rev}(e[s,t])) = \widehat{\text{Cov}}_{\mathcal{D}'}(\text{rev}(e[s,t]))$. For reversals of subedges of S the notion of good and bad indices is defined relative to $\mathcal{D}'$. In the remainder of this section we will analyze the proxy coverage for subedges only, as the above statement is trivial for vertex-vertex-subcurves by setting $\pi_1 = \pi_2 = \pi$.

Lemma 4.6.9. *If i is bad for $e[s,t]$, then i is good for* $\text{rev}(e[s,t])$. *Furthermore, $l_i(s)$ and $r_i(t)$ lie in $[l_i(t), r_i(s)]$.*

Proof. Let i be bad for $e[s,t]$. This implies that all bottommost points in cell i of $\mathcal{D}_e$ are to the right of all topmost points. As $\mathcal{D}'_e$ results from $\mathcal{D}_e$ by mirroring it along the y-axis, all bottommost points of cell i in $\mathcal{D}'_e$ lie to the left of all topmost points. Hence i can not be bad for $\text{rev}(e[s,t])$.

For the second claim first observe that $l_i(s) \neq \infty \neq r_j(t)$ and thus by Observation 4.6.5 $l_i(t) < r_i(s)$. Further observe that t lies between s and the y-coordinate of the topmost point of cell i. Thus by convexity of the free space and the fact that $l_i(s)$ lies to the right of the topmost point there is a point in the free space at height t that is left of $l_i(s)$ and thus in particular $l_i(t) < l_i(s)$. Similarly it follows that $r_i(t) < r_i(s)$ and thus the claim follows. □

Lemma 4.6.10. *The proxy coverage approximates the coverage (refer to Figure 4.8), i.e.,*

$$\text{Cov}_{\mathcal{D}}(e[s,t]) \subset \widehat{\text{Cov}}_{\mathcal{D}}(e[s,t]) \cup \widehat{\text{Cov}}_{\mathcal{D}}(\text{rev}(e[s,t])).$$

Proof. If i is in $\mathcal{L}(e[s,t]) \setminus \mathcal{B}(e[s,t])$, then $[l_i(s), r_i(t)] \subset \widehat{\text{Cov}}_{\mathcal{D}}(e[s,t])$. If instead i is in $\mathcal{L}(e[s,t]) \cap \mathcal{B}(e[s,t])$ then by Lemma 4.6.9 it follows that $[l_i(s), r_i(t)] \subset [l_i(t), r_i(s)] \subset \widehat{\text{Cov}}_{\mathcal{D}}(\text{rev}(e[s,t]))$.

Let now instead $(i,j) \in \widetilde{\mathcal{G}}(e[s,t])$. We show that

$$[l_i(s), r_i(s)] \cup [r_i(s), l_j(t)] \cup [l_j(t), r_j(t)] \subset \widehat{\text{Cov}}_{\mathcal{D}}(e[s,t]) \cup \widehat{\text{Cov}}_{\mathcal{D}}(\text{rev}(e[s,t])).$$

First observe that $[r_i(s), l_j(t)] \subset \widehat{\text{Cov}}_{\mathcal{D}}(e[s,t])$, as $(i,j) \in \widetilde{\mathcal{G}}(e[s,t])$. Assume i is bad for $e[s,t]$, as otherwise $[l_i(s), r_i(s)] \subset [l_i(s), l_j(t)] \subset \widehat{\text{Cov}}_{\mathcal{D}}(e[s,t])$. But then by Lemma 4.6.9 $[l_i(s), r_i(s)] \subset [l_i(t), r_i(s)] \subset \widehat{\text{Cov}}_{\mathcal{D}}(\text{rev}(e[s,t]))$ (refer to Figure 4.10). Conversely, regardless of if j is bad or not for $e[s,t]$, $[l_j(t), r_j(t)] \subset \widehat{\text{Cov}}_{\mathcal{D}}(e[s,t]) \cup \widehat{\text{Cov}}_{\mathcal{D}}(\text{rev}(e[s,t]))$ implying the claim. □

Theorem 4.6.11. *For any curve $\pi \in \mathbb{X}^{d,\ell}$ there is a set $S_\pi \subset \mathcal{C}_S(E)$ of size at most* 16 *such that*

$$\text{Cov}_P(\pi, \Delta) \subset \bigcup_{s \in S_\pi} \widehat{\text{Cov}}_{\mathcal{D}}(s).$$

Proof. This follows from the definition of the proxy coverage, Lemma 4.6.10, and Theorem 4.3.16. □

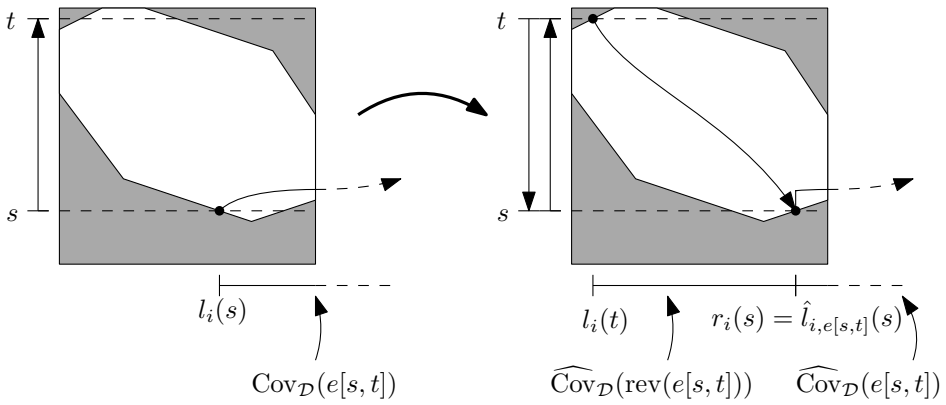


Figure 4.10: Illustration of how the path starting in cell i from the global group of the coverage of $e[s,t]$ is dominated by the proxy coverage of $e[s,t]$ and $\mathrm{rev}(e[s,t])$ if i is bad for $e[s,t]$ and hence good for $\mathrm{rev}(e[s,t])$. Importantly $[l_i(s), r_j(t)] \subset [l_i(t), r_i(s)] \cup [\hat{l}_i(s), r_j(t)]$.

4.6.3 Maintaining the Proxy Coverage Along a Sweep-Sequence

In this section we show that during a linear scan of a sweep-sequence one can correctly maintain a symbolic representation of $\widehat{\mathrm{Cov}}_{\mathcal{D}}(e[s,t])$ in (roughly) logarithmic time per element in the sweep-sequence. To this end we show that one can correctly maintain $\widetilde{\mathcal{G}}(e[s,t])$, $\mathcal{L}(e[s,t]) \setminus \mathcal{B}(e[s,t])$, and $\mathcal{B}(e[s,t])$. This involves maintaining $\mathcal{G}(e[s,t])$ which in turn involves maintaining the set $\mathcal{U}(e[s,t])$ consisting of all inclusionwise maximal index pairs (i,j) such that there is a monotone path from cell i at height s into cell j at some height $t' \leq t$. By inclusionwise maximality observe that for any index i there is at most one index j with $(i,j) \in \mathcal{U}(e[s,t])$. Similarly for any j there is at most one i with $(i,j) \in \mathcal{U}(e[s,t])$.

Maintenance of $\mathcal{U}(\cdot)$

For an index pair (i,j) and value s such that $l_s(i) \neq \infty$ we define its left extension $\overleftarrow{(i,j)_s}$ to be the index pair (i^*,j) where $i^* \leq i$ is the biggest index such that the leftmost point in cell i^* at height s does not lie in the left free space interval, or $i^* = 1$. In particular, if $(i,j) \in \mathcal{U}(e[s,t])$, then $\overleftarrow{(i,j)_s} \in \mathcal{U}(e[s,t])$ and $\overleftarrow{(i,j)_s}$ includes (i,j). Similarly, for an index pair (i,j) define its right restriction $\overrightarrow{(i,j)_s}$ to be the index pair (i^*,j) where $i < i^* \leq j$ is the smallest index such that there is a leftmost point in cell i^* at height s. Unlike the left extension, the right restriction does not necessarily exist. We will later observe that, given the right data structure, both the left extension and the right restriction can be computed efficiently. Conceptually, both the left extension and the right restriction for the index pair (i,j) are the index pair (i^*,j), where i^* is the index of the cell in which a horizontal ray starting in cell i at height s oriented to the left (resp. right) intersects the boundary of the free space.

The following lemmas give precise instructions on how to update $\mathcal{U}(e[s,t])$ to result in $\mathcal{U}(e[s',t'])$ with only $\mathcal{O}(1)$ changes to the set, or more precisely, how to update $\mathcal{U}(e[s,t])$ to result in $\mathcal{U}(e[s',t])$, and how to update $\mathcal{U}(e[s',t])$ to result in $\mathcal{U}(e[s',t'])$ in $\mathcal{O}(1)$ changes, given $s < s'$ and $t < t'$ are consecutive elements in E_e.

Lemma 4.6.12. *Let $s' < s \leq t \in E_e$ be given, with s' and s consecutive. Let I_s and $I_{s'}$ be all index pairs (i,j) such that there is a monotone path from cell i at height s (resp.*

s') to a point in cell j at height at most t. Denote for any set I of index pairs the set of inclusionwise maximal index pairs by $M(I)$. Then

(i) $\mathcal{U}(e[s,t]) = M(I_s)$ *and* $\mathcal{U}(e[s',t]) = M(I_{s'})$,

(ii) if s does not correspond to some bottommost point then $M(I_s) = M(I_s \cap I_{s'})$

(iii) if s corresponds to the bottommost point in cell i and $(i,j) \in \mathcal{U}(e[s,t])$ for some j, then let (i^*,j) be the right extension $\overrightarrow{(i,j)}_s$. If it exists, let j^* be such that $(i^*,j^*) \in M(I_s)$ and $j^* > j$. If j^* exists, then

$$M(I_s \cap I_{s'}) = M(I_s) \setminus \{(i,j)\},$$

otherwise

$$M(I_s \cap I_{s'}) = M(I_s) \setminus \{(i,j)\} \cup \{(i^*,j)\},$$

(iv) if s' corresponds to neither a topmost point of some cell, or the topmost point in the left free space interval of some cell, then $M(I_{s'}) = M(I_s \cap I_{s'})$.

(v) if s' corresponds to the topmost point in cell i, then $M(I_{s'}) = M(I_s \cap I_{s'}) \cup \{(i,i)\}$,

(vi) if s' corresponds to the topmost point in the left free space interval of cell i, then $(i,j) \in M(I_s \cap I_{s'})$ and

$$M(I_{s'}) = M(I_s \cap I_{s'}) \setminus \left\{\overleftarrow{(i-1,i-1)_{s'}}, (i,j)\right\} \cup \left\{\overleftarrow{(i,j)_{s'}}\right\},$$

Proof. Observe that *(i)* holds by definition of $\mathcal{U}(e[s,t])$ and $\mathcal{U}(e[s',t])$.

For *(ii)*, let us first assume that s does not correspond to the bottommost point of some cell. Then any path from some cell i at height s to some cell j at height $t' \leq t$ induces a path from cell i at height s' to cell j at height $t'' \leq t$, where $t'' = t'$ unless $i = j$, in which case $t'' = s' \leq t$. As such $I_s \subset I_{s'}$ and hence $I_s \cap I_{s'} = I_s$.

For *(iii)*, let s correspond to the bottommost point in cell i and let I be all index pairs (i,b) such that there is a monotone path from cell i at height s to cell b at some height $t' \leq t$. In particular, $I_s \cap I_{s'} = I_s \setminus I$, as any path from some cell a at height s to some cell b at height $t' \leq t$ induces a path from cell a at height s' to some cell b at some height $t'' \leq t$ unless $a = i$. Let (i,j) be the inclusionwise maximal element in I. If $(i,j) \notin M(I_s)$, then in particular $M(I_s \cap I_{s'}) = M(I_s \setminus I) = M(I_s)$. Let instead $(i,j) \in M(I_s)$. Let J be the subset of $I_s \cap I_{s'}$ included in (i,j). Then, if $(i^*,j) = \overrightarrow{(i,j)}_{s'}$ exists, it is the inclusionwise maximal element in J, and $J = \emptyset$ otherwise. Hence, any index pair in $I_s \cap I_{s'}$ is included in some index pair in $M(I_s) \setminus \{(i,j)\} \cup \{(i^*,j)\}$, and as such $M(I_s \cap I_{s'}) \subset M(I_s) \setminus \{(i,j)\} \cup \{(i^*,j)\}$. The only element in $M(I_s) \setminus \{(i,j)\} \cup \{(i^*,j)\}$ that may not be inclusionwise maximal is (i^*,j), which is not inclusionwise maximal if and only if some $j^* > j$ exists such that $(i^*,j^*) \in M(I_s)$.

For *(iv)*, observe that a monotone path from cell a at height s' to some cell b at some height $t' \leq t$ induces a path from cell a at height s to cell b at some height $t'' \leq t$, unless the path passes through a free space interval that is not possible by a path starting at height s, or it starts in a cell, where there is no point in $\mathcal{D}$ at height s. This corresponds exactly to topmost free space intervals and topmost points of cells.

For *(v)*, let s' correspond to the topmost point of cell i. As no point in cell i of $\mathcal{D}$ is at height s, and by the total order of y-coordinates of extremal points (discussed in Section 2.3) no path starting at height s' may enter or exit via the free space intervals of cell i. Thus, $I_{s'} = I_s \cap I'_s \cup \{(i,i)\}$. Further, as no path starting at height s can pass cell i, (i,i) is not included in any index pair in $I_s \cap I'_s$. As such, $M(I_{s'}) = M(I_s \cap I_{s'}) \cup \{(i,i)\}$.

Lastly, for *(vi)*, let s' correspond to the topmost point in the left free space interval of cell i. Observe that $I_{s'} \setminus I_s$ corresponds exactly to all index pairs including $(i-1,i)$,

i.e., index pairs corresponding to paths passing the left free space interval of cell i. As any such path corresponding to the index pair (a, b) must pass the left free space interval at height exactly s', any such path splits into first a horizontal line segment from cell a to cell $i-1$ at height s', and secondly a path starting in cell i at height s' to cell b at some height $t' \leq t$. Let I and J be the sets of index pairs corresponding to all first and all second such paths respectively. Clearly, every index pair in I is dominated by $\overleftarrow{(i-1, i-1)_{s'}}$. For any $(i, b) \in J$, observe that any corresponding path induces a path starting at height s in cell i to cell b at some height $t'' \leq t$. As no path starting at height s may enter cell i, there is an element $(i, j) \in M(I_s \cap I_{s'})$ including all index pairs in J, and hence the inclusionwise maximal element (i, j) of J must be in $M(I_s \cap I_{s'})$. Finally, $\overleftarrow{(i, j)_{s'}}$ includes all element of I and J and is in $I_{s'} \setminus I_s$, and hence $M(I_{s'})$ is precisely the set $M(I_s \cap I_{s'}) \setminus \left\{\overleftarrow{(i-1, i-1)_{s'}}, (i, j)\right\} \cup \left\{\overleftarrow{(i, j)_{s'}}\right\}$, concluding the proof. □

Lemma 4.6.13. *Let $s \leq t' < t \in E_e$ be given, with t' and t consecutive. If t corresponds to the bottommost point in the left free space interval of some cell m then there is at most one index pair $(i, j) \in \mathcal{U}(e[s, t])$ with $i < m \leq j$. If there is no such i and j, then $\mathcal{U}(e[s, t']) = \mathcal{U}(e[s, t])$. Otherwise, let (i^*, j) be the right extension $\overrightarrow{(m, j)_s}$. If it exists, let j^* be such that $(i^*, j^*) \in \mathcal{U}(e[s, t])$ and $j^* > j$. If j^* exists, then*

$$\mathcal{U}(e[s, t']) = \mathcal{U}(e[s, t]) \setminus \{(i, j)\} \cup \{(i, m-1)\},$$

otherwise

$$\mathcal{U}(e[s, t']) = \mathcal{U}(e[s, t]) \setminus \{(i, j)\} \cup \{(i, m-1), (i^*, j)\}.$$

Proof. Let I_t and $I_{t'}$ be all index pairs (a, b) such that there is a monotone path from cell a at height s to cell b at height t and t' respectively. First observe that $I_{t'} \subset I_t$, as any monotone path from some cell a at height s to some cell b at height $t'' \leq t'$ is also a path to cell b at height $t'' \leq t$. Next, observe that the only index pairs in $I_t \setminus I_{t'}$ are those corresponding to index pairs that can pass through a free space interval that can no longer be reached with a maximal height of t'. As such, unless t corresponds to the bottommost point in a free space interval, $I_t = I_{t'}$ and hence $\mathcal{U}(e[s, t']) = \mathcal{U}(e[s, t])$.

Instead, let t correspond to the bottommost point in the left free space interval of some cell m. Any index pair in $I_{t'} \subset I_t$ must hence include $(m-1, m)$. Further, any path corresponding to such an index pair (a, b) splits into first a path from cell a at height s to cell $m-1$ at height exactly t, and secondly a path from cell m starting at height t and ending in cell b at height t. Let I and J be the set of index pairs corresponding to all such first and second paths respectively. Let $(i, m-1)$ be the inclusionwise maximal element in I, and (m, j) the inclusionwise maximal element in J. Clearly (i, j) is the inclusionwise maximal element in $I_{t'} \subset I_t$ and must also be in $\mathcal{U}(e[s, t])$. Next, observe that the set of index pairs in I_t not contained in $\mathcal{U}(e[s, t]) \setminus \{(i, j)\}$ are either contained in $(i, m-1)$, or are contained in (m, j), and hence are included in either $(i, m-1)$ or $\overrightarrow{(m, j)_s}$, which both are in $I_{t'}$. Lastly $(i^*, j) = \overrightarrow{(m, j)_s}$ is not in $\mathcal{U}(e[s, t'])$ if and only if there is some path from cell i^* at height s to some cell $j^* > j$ at some height $t'' \leq t$. But then (i^*, j^*) is also in $\mathcal{U}(e[s, t])$. Hence if j^* exists then

$$\mathcal{U}(e[s, t']) = \mathcal{U}(e[s, t]) \setminus \{(i, j)\} \cup \{(i, m-1)\},$$

otherwise

$$\mathcal{U}(e[s, t']) = \mathcal{U}(e[s, t]) \setminus \{(i, j)\} \cup \{(i, m-1), (i^*, j)\}.$$ □

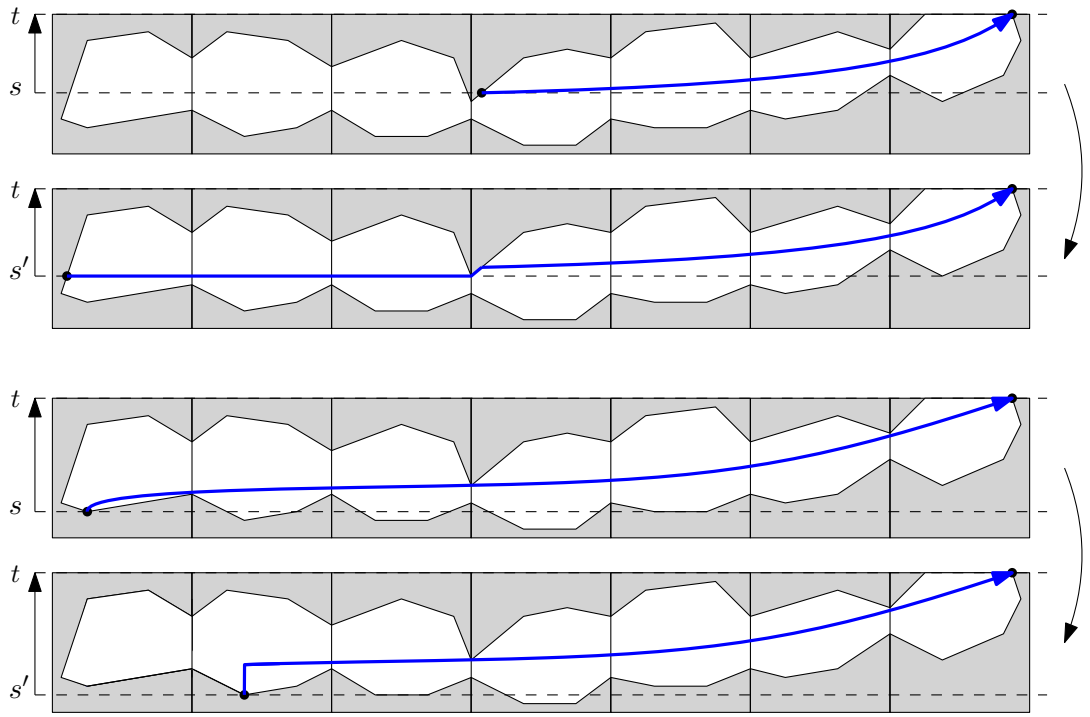


Figure 4.11: Illustration of the left extension and right restriction updates to a path inducing an index pair in $\mathcal{U}(e[s,t])$ to a path inducing an index pair in $\mathcal{U}(e[s',t])$.

We now provide data structures that allow performing the described updates to $\mathcal{U}(\cdot)$ in logarithmic time, which involves computing left extensions and right restrictions (refer to Figure 4.11).

Lemma 4.6.14 (Shoot-left data structure)**.** *Let $I = \{I_1, \ldots, I_n\}$ be a list of intervals in $\mathbb{R}$. One can build in $\mathcal{O}(n \log n)$ time a data structure that allows queries with input $i \leq n$ and $x \in \mathbb{R}$ correctly outputting the smallest index $j \leq i$ such that $x \in \bigcap_{j \leq s \leq i} I_s$ or determining that there is no such j.*

Proof. Store the intervals in a balanced binary tree, where every inner node stores the intersection of the intervals stored in its children nodes. This construction takes $\mathcal{O}(n \log n)$ time. Thus if the query point x lies in the interval stored at some node r of the tree, then it also lies in the intersection of all intervals stored in the leaves of the tree rooted at r.

Now, let i and x be given. If $x \notin I_i$, return that there is no such j. Otherwise we temporarily modify the tree as follows: Traverse the tree upwards starting at the node of I_i removing all children of nodes that represent trees of intervals whose index is strictly bigger than i, updating the intervals along the way. This modification takes $\mathcal{O}(\log n)$ time. It results in a tree of height $\log n$ whose leaves are all intervals $\{I_1, \ldots, I_i\}$ such that every node stores the intersection of all intervals of the leaves on its subtree. Next we traverse the tree starting at the root, checking whether x lies in the stored interval at the root. If it does, we can output the index 1. Otherwise assume there is an interval I_j which is the interval with the largest index that does not contain x. We now maintain the value $\hat{j}$ such that x is in all intervals $I_{\hat{j}}, \ldots, I_i$. Initially, $\hat{j} = i$. We recursively

check for the current node the interval stored in its two children. If the interval stored in the right child contains x, then we update $\hat{j}$ as the smallest index of leaves among the subtree rooted at the right child and recurse on the left child. If the interval stored in the right child does not contain x we instead recurse on the right child, not updating $\hat{j}$. We output $\hat{j}$ once we arrived at a leaf. The correctness and running time are an immediate consequence of the traversal of the tree with its stored data. □

Lemma 4.6.15 (Jump-right data structure)**.** *Let $A = \{a_1, \ldots, a_n\}$ be a list of values. One can build in $\mathcal{O}(n \log n)$ time a data structure that allows queries with input i correctly outputting the smallest index $j > i$ such that $a_i > a_j$ or determining that there is no such j in $\mathcal{O}(1)$ time.*

Proof. First sort the indices $\{1, \ldots, n\}$ with the values $\{a_1, \ldots, a_n\}$ as keys in time $\mathcal{O}(n \log n)$. The resulting list of indices is then iteratively inserted into a sorted list (this time by index). Each insertion takes $\mathcal{O}(\log n)$ time. Furthermore, when some index i is inserted at position k, the smallest index $j > i$ such that $a_i > a_j$ (if it exists), is at position $k + 1$. Hence, when inserting this index i, look up the index at position $k + 1$ and store it as the answer to the query with index i. □

Maintenance of $\widetilde{\mathcal{G}}(\cdot)$ and $\mathcal{L}(\cdot) \setminus \mathcal{B}(\cdot)$

Next, we analyze how to obtain $\mathcal{L}(e[s,t]) \setminus \mathcal{B}(e[s,t])$ and $\mathcal{G}(e[s,t])$ from $\mathcal{U}(e[s,t])$. Importantly, we will observe that both $\mathcal{L}(e[s,t]) \setminus \mathcal{B}(e[s,t])$ and $\mathcal{G}(e[s,t])$ can be updated alongside $\mathcal{U}(e[s,t])$.

Lemma 4.6.16. *Let e be an edge of S and let $\mathfrak{s} \in \mathfrak{S}_e$ be a sweep-sequence of E_e. For every i the set $B_i = \{(s,t) \in \mathfrak{s} | i \text{ is bad for } e[s,t]\}$ is a contiguous subset of $\mathfrak{s}$. The boundaries of all B_i can be computed in time $\mathcal{O}(n \log n)$.*

Proof. This is an immediate consequence of the fact that for the rightmost topmost point at x-coordinate x_{top}, the set $\{(s,t) \in \mathfrak{s} | l_i(s) \neq \infty,\ l_i(s) \geq x_{\text{top}}\}$ is contiguous. So is the set $\{(s,t) \in \mathfrak{s} | r_i(t) \neq \infty,\ r_i(t) \leq x_{\text{bottom}}\}$, where x_{bottom} is the x-coordinate of the leftmost bottommost point. B_i is simply the intersection of these two sets. The boundaries of this contiguous set can be computed via binary search over $\mathfrak{s}$. □

Observation 4.6.17. *For any edge e of S and $0 \leq s \leq t \leq 1$ by definition of $\mathcal{L}(e[s,t])$, $\mathcal{U}(e[s,t])$, and $\mathcal{B}(e[s,t])$*

$$\mathcal{L}(e[s,t]) \setminus \mathcal{B}(e[s,t]) = \{(i,j) \in \mathcal{U}(e[s,t]) \mid i = j,\ r_j(t) \neq \infty, \text{ and } i \notin \mathcal{B}(e[s,t])\}.$$

Lemma 4.6.18. *For every $s \leq t \in E_e$ it holds that $\mathcal{G}(e[s,t])$ is the set of index pairs (i,j) such that $r_j(t) \neq \infty$, i is the smallest index such that there is a j' with $(i,j') \in \mathcal{U}(e[s,t])$ and for all $j < j^* \leq j'$ it holds that $r_{j^*}(t) = \infty$.*

Proof. Let $(i,j) \in \mathcal{G}(e[s,t])$. Trivially, $r_j(t) \neq \infty$. The index i is the smallest index such that there is a path from cell i at height s to cell j at some height $t' \leq t$. But this coincides with the smallest index i such that there is an index j' with $(i,j') \in \mathcal{U}(e[s,t])$ and $i \leq j \leq j'$. As $(i,j) \in \mathcal{G}(e[s,t])$, there is no j^* with $j < j^* \leq j'$ with $r_{j^*}(t) \neq \infty$ as otherwise there would be a path from cell i at height s to cell j^* at some height t and hence $(i,j) \notin \mathcal{G}(e[s,t])$. □

Algorithm 5 Maintenance of the reduced global group

1: **procedure** MAINTAIN(e, $\mathcal{D}$, sweep-sequence $\mathfrak{s} = \{(s_1, t_1), \ldots, (s_m, t_m)\}$ of E_e)
2: Construct shoot-left data structure based on left free space intervals of cells in $\mathcal{D}$
3: Construct jump-right data structure based on lowest points of cell in $\mathcal{D}$
4: For every i compute $B_i = \{(s,t) \in \mathfrak{s} \mid i \text{ is bad for } e[s,t]\}$
5: Let $s = s_m$ and $t = t_m$
6: Compute $\mathrm{Cov}_{\mathcal{D}}(e[s,t])$ represented as $\mathcal{O}(n)$ disjoint intervals
7: Compute $R_t = \{j \mid r_j(t) \neq \infty\}$ and $\mathcal{B}(e[s,t])$
8: Compute $\mathcal{U}(e[s,t])$ and with it $\mathcal{L}(e[s,t]) \setminus \mathcal{B}(e[s,t])$, $\mathcal{G}(e[s,t])$ and $\widetilde{\mathcal{G}}(e[s,t])$
9: Sort R_t, $\mathcal{U}(e[s,t])$, $\mathcal{L}(e[s,t]) \setminus \mathcal{B}(e[s,t])$, $\mathcal{G}(e[s,t])$ and $\widetilde{\mathcal{G}}(e[s,t])$
10: For every $(i,j) \in \mathcal{U}(e[s,t])$ compute $j^*_{(i,j)} = \max\{j^* \in R_t \mid i \leq j^* \leq j\}$
11: Compute $G = \{(i, j, j_{(i,j)}) \mid (i,j) = \min\{(i',j') \in \mathcal{U}(e[s,t]) \mid i' \leq j_{(i,j)} \leq j'\}\}$
12: **for** $(s',t') \in \{(s_{m-1}, t_{m-1}), \ldots, (s_1, t_1)\}$ **do**
13: **if** $s' < s$ **then**
14: Update $\mathcal{U}(e[s,t])$ to store $\mathcal{U}(e[s',t])$ instead, according to Lemma 4.6.12
15: **else if** $t' < t$ **then**
16: Update $\mathcal{U}(e[s',t])$ to store $\mathcal{U}(e[s',t'])$ instead, according to Lemma 4.6.13
17: Update R_t to store $R_{t'}$ instead
18: Update $\mathcal{B}(e[s,t])$ via the sets B_i
19: Update G and with it $\mathcal{G}(e[s,t])$ via the updates to $\mathcal{U}(e[s,t])$ and R_t
20: Update $\mathcal{L}(e[s,t]) \setminus \mathcal{B}(e[s,t])$ via the updates of $\mathcal{U}(e[s,t])$ and $\mathcal{B}(e[s,t])$
21: Update $\widetilde{\mathcal{G}}(e[s,t])$ via the updates to $\mathcal{G}(e[s,t])$ and $\mathcal{B}(e[s,t])$
22: $(s,t) \leftarrow (s',t')$

Thus, the strategy for an algorithm maintaining $\mathcal{L}(\cdot) \setminus \mathcal{B}(\cdot)$ and $\mathcal{G}(\cdot)$ during a linear scan of a sweep sequence is as follows (refer to Algorithm 5): We maintain a sorted list R_t consisting of all indices j such that $r_j(t) \neq \infty$ together with a sorted list consisting of index pairs in $\mathcal{U}(e[s,t])$. This list may be sorted by either its first or last index, resulting in the same ordering by the inclusionwise maximality of the elements in $\mathcal{U}(e[s,t])$. Based on the updates performed to $\mathcal{U}(\cdot)$ and R_t we can maintain $\mathcal{L}(\cdot)\setminus\mathcal{B}(\cdot)$ easily, while maintaining $\mathcal{G}(\cdot)$ takes slightly more effort. For it, maintain the set G consisting of all triples of indices $(i,j,\hat{j})$ such that $(i,j) \in \mathcal{U}(\cdot)$, $\hat{j} \in R_t$, $i \leq \hat{j} \leq j$, for the next smallest index pair $(i',j') < (i,j)$ in $\in \mathcal{U}(\cdot)$ it holds that $j' < \hat{j}$ and for the next biggest index $\hat{j}' > \hat{j}$ in R_t it holds that $\hat{j}' > j$. By Lemma 4.6.18, the set $G(\cdot)$ is precisely the set of pairs of the first and last index in any triple in G. Furthermore, G can be maintained under addition and removal of elements from $\mathcal{U}(\cdot)$ and R_t in logarithmic time per addition or removal, using $\mathcal{O}(1)$ searches over the maintained sorted lists. Overall, throughout the linear scan, there are $\mathcal{O}(n)$ updates to $\mathcal{G}(\cdot)$, where at any point no index pair is contained in another index pair. As $\mathcal{B}(\cdot)$ is also correctly maintained, $\widetilde{\mathcal{G}}(\cdot)$ may be maintained. At every update to $\mathcal{G}(\cdot)$ we check whether $\widetilde{\mathcal{G}}(\cdot)$ changes depending only on the updated index pair and its two neighboring index pairs and change $\widetilde{\mathcal{G}}(\cdot)$ accordingly. Similarly, at every update to $\mathcal{B}(\cdot)$ we check whether $\widetilde{\mathcal{G}}(\cdot)$ changes depending only on the at most two elements in $\mathcal{G}(\cdot)$ whose first or second index changed according to the update to $\mathcal{B}(\cdot)$, and change $\widetilde{\mathcal{G}}(\cdot)$ accordingly.

Theorem 4.6.19. *At the end of each iteration of the loop in Line 12 in* MAINTAIN *in Algorithm 5, the sets of index pairs $\widetilde{\mathcal{G}}(e[s,t])$ and $\mathcal{L}(e[s,t]) \setminus \mathcal{B}(e[s,t])$ are correctly updated. Further, executing* MAINTAIN *takes $\mathcal{O}(n \log n)$ time.*

Proof. By definition of $\mathfrak{S}_e$, for the last element (s,t) in any $\mathfrak{s} \in \mathfrak{S}_e$ it holds that $s = t$. Hence, initially $\mathrm{Cov}_{\mathcal{D}}(e[s,t])$ coincides with the intersection of $\mathcal{D}$ with a horizontal line rooted at height $s = t$. The connected components of this intersection induce $\mathcal{G}(e[s,t])$ and $\mathcal{L}(e[s,t])$ and hence $\mathcal{U}(e[s,t])$. Hence initially, by Lemma 4.4.6, it holds that $\mathcal{U}(e[s,t]) = \mathcal{G}(e[s,t]) \cup \mathcal{L}(e[s,t])$ is computed correctly in time $\mathcal{O}(n \log n)$ time. As the boundaries of the sets B_i are also computed correctly by Lemma 4.6.16, so are $\mathcal{B}(e[s,t])$, so are the sets $\mathcal{L}(e[s,t]) \setminus \mathcal{B}(e[s,t])$ and $\mathcal{G}(e[s,t])$ and $\widetilde{\mathcal{G}}(e[s,t])$.

Now consider some iteration of the **for**-loop. By Lemma 4.6.12 and Lemma 4.6.13 together with Lemma 4.6.14 and Lemma 4.6.15 the updates to $\mathcal{U}(e[s,t])$ can be performed in $\mathcal{O}(\log n)$ time. R_t can be updated in $\mathcal{O}(\log n)$ time, as only one index is added or removed depending on whether t corresponds to a top- or bottommost point of $\mathcal{D}$ in some cell.

There are $B_{(s,t)} = |\mathcal{B}(e[s,t]) \setminus \mathcal{B}(e[s',t'])| + |\mathcal{B}(e[s',t']) \setminus \mathcal{B}(e[s,t])|$ updates to $\mathcal{B}(e[s,t])$. By Lemma 4.6.16 each B_i is a contiguous subset of $\mathfrak{s}$, and its boundaries can be computed in $\mathcal{O}(\log n)$ time. Hence $\mathcal{B}(e[s',t'])$ can be updated based on $\mathcal{B}(e[s',t'])$ and all sets B_i whose boundaries are either s, s', t or t' in total time $\mathcal{O}(B_{(s,t)} \log n)$ time.

Next the updates to G can be performed based on the $\mathcal{O}(1)$ updates to R_t and $\mathcal{U}(e[s,t])$ in $\mathcal{O}(\log n)$ time each, as each addition or removal to R_t and $\mathcal{U}(e[s,t])$ results in at most $\mathcal{O}(1)$ triples in G changing, being inserted, or deleted. These can be identified and modified in $\mathcal{O}(\log n)$ time each. The updates to $\mathcal{L}(e[s,t]) \setminus \mathcal{B}(e[s,t])$ can be performed as by Observation 4.6.17 each update to $\mathcal{B}(e[s,t])$ and $\mathcal{U}(e[s,t])$ influence the decision of whether an index is in $\mathcal{L}(e[s,t]) \setminus \mathcal{B}(e[s,t])$ for at most one index, and hence can be performed in $\mathcal{O}(B_{(s,t)} \log n)$ total time. Lastly the updates to $\widetilde{\mathcal{G}}(e[s,t])$ depend only on the $\mathcal{O}(1)$ updates to $\mathcal{G}(e[s,t])$ and the $B_{(s,t)}$ updates to $\mathcal{B}(e[s,t])$, and each addition or removal from the two sets induce a change in $\widetilde{\mathcal{G}}(e[s,t])$ for which, by inclusionwise maximality of the partaking elements, we need to only check the two neighbors of the inserted or removed index or index pair to apply the according change to $\widetilde{\mathcal{G}}(e[s,t])$.

In total the running time of the algorithm is in

$$\mathcal{O}\left(\sum_{(s,t)\in\mathfrak{s}} \left(\log n + B_{(s,t)} \log n\right)\right) = \mathcal{O}\left(n \log n + \sum_{(s,t)\in\mathfrak{s}} \log n\right) = \mathcal{O}(n \log n). \qquad \square$$

Corollary 4.6.20. *Let e be an edge of S and let $\mathfrak{s} \in \mathfrak{S}_e$ be a sweep-sequence. There are $m = \mathcal{O}(n)$ index pairs $p_1 = (i_1, j_1), \ldots, p_m = (i_m, j_m)$ together with m contiguous subsets $I_i = \{(s_{a_i}, t_{a_i}), \ldots, (s_{b_i}, t_{b_i})\} \subset \mathfrak{s}$, such that for every $(s,t) \in \mathfrak{s}$ it holds that*

$$\widetilde{\mathcal{G}}(e[s,t]) \cup (\mathcal{L}(e[s,t]) \setminus \mathcal{B}(e[s,t])) = \bigcup_{\substack{1\le i\le m \\ \text{with } (s,t)\in I_i}} \{p_i\}.$$

Further for $p_k = (i_k, j_k)$ the index i_k is either always good or always bad for $e[s,t]$ for $(s,t) \in I_k$, and the index j_k is either always good or always bad for $e[s,t]$ for $(s,t) \in I_k$. The index pair p_i as well as the values a_i and b_i can be computed for all i in total time $\mathcal{O}(n \log n)$.

Proof. This is an immediate consequence of Theorem 4.6.19, as throughout Algorithm 5 $\widetilde{\mathcal{G}}(e[s,t]) \cup (\mathcal{L}(e[s,t]) \setminus \mathcal{B}(e[s,t]))$ is correctly maintained along $\mathfrak{s}$ in $\mathcal{O}(n)$ updates to $\mathcal{O}(n)$ initial index pairs. We store any index pair in $\widetilde{\mathcal{G}}(e[s,t]) \cup (\mathcal{L}(e[s,t]) \setminus \mathcal{B}(e[s,t]))$ whenever it gets added, removed, modified, or either of its entries changes from good to bad proving the claim. $\square$

Thus, we are able to maintain a symbolic representation of $\widehat{\mathrm{Cov}}_{\mathcal{D}}(\cdot)$ during a linear scan of a sweep-sequence $\mathfrak{s} \in \mathfrak{S}_e$ in total time $\mathcal{O}(n \log n)$.

Putting It All Together

Finally, we present the following theorem which uses the maintained sets $\widetilde{\mathcal{G}}(\cdot)$ and $\mathcal{L}(\cdot) \setminus \mathcal{B}(\cdot)$ for batch point-queries and constitutes the main result of this section. It is one of the central algorithmic insights underlying the cubic time algorithms we explore in Section 4.7 and Section 4.8.

Theorem 4.6.21. *Let $\mathfrak{s} \in \mathfrak{S}_e$, and let $Q \subset [0,1]$ together with $w_Q : Q \to \mathbb{N}$ be a weighted point set. Let for any $Q' \subset Q$ its weight $w_Q(Q')$ be defined as $\sum_{q \in Q'} w_q(q)$. There is an algorithm which computes $w_Q(Q \cap \widehat{\mathrm{Cov}}_{\mathcal{D}}(e[s,t]))$ for every $(s,t) \in \mathfrak{s}$ in $\mathcal{O}(|Q| \log |Q| + n \log n)$ time.*

Proof. Let Q_i be the set of points in Q that lie on the i^{th} edge of P. Define the following values for all i, j along the sweep-sequence $\mathfrak{s}$ where for $(s,t) \in \mathfrak{s}$:

$$L_i((s,t)) = \sum_{\substack{q \in Q_i \\ \hat{l}_{i,e[s,t]}(s) \neq \infty \\ \hat{l}_{i,e[s,t]}(s) \leq q}} w_Q(q), \qquad R_j((s,t)) = \sum_{\substack{q \in Q_j \\ \hat{r}_{j,e[s,t]}(t) \neq \infty \\ q \leq \hat{r}_{j,e[s,t]}(t)}} w_Q(q),$$

$$M_i((s,t)) = \sum_{\substack{q \in Q_i,\ i \notin \mathcal{B}(e[s,t]) \\ l_i(s) \neq \infty,\ r_i(t) \neq \infty \\ l_i(s) \leq q \leq r_i(t)}} w_Q(q), \quad D(i,j) = \sum_{\substack{i \leq m \leq j \\ q \in Q_m}} w_Q(q).$$

The value $L_i(s,t)$ corresponds to the weight of points in Q_i that lie right of $\hat{l}_{i,e[s,t]}(s)$. Similarly $R_i(s,t)$ corresponds to the weight of points in Q_i that lie left of $\hat{r}_{i,e[s,t]}(t)$. The value $M_i(s,t)$ corresponds to the weight of points in Q_i that lie in between $\hat{l}_{i,e[s,t]}(s)$ and $\hat{r}_{i,e[s,t]}(t)$ if i is good for $e[s,t]$. Lastly $D(i,j)$ corresponds to the total weight of points in $\bigcup_{i \leq m \leq j} Q_m$. Then for any $(s,t) \in \mathfrak{s}$ and any $(i,i) \in \mathcal{L}(e[s,t]) \setminus \mathcal{B}(e[s,t])$ it holds that $M_i((s,t)) = w_Q(Q \cap [l_i(s), r_i(t)])$. Similarly, for any $(i,j) \in \widetilde{\mathcal{G}}(e[s,t])$ it holds that $L_i((s,t)) + D(i+1, j-1) + R_j((s,t))$ equals $w_Q(Q \cap [\hat{l}_{i,e[s,t]}(s), r_{i,e[s,t]}(t)])$. Hence

$$\begin{aligned}
&w_Q(Q \cap \widehat{\mathrm{Cov}}_{\mathcal{D}}(e[s,t])) = \\
&= \left(\sum_{i \in \mathcal{L}(e[s,t]) \setminus \mathcal{B}(e[s,t])} w_Q(Q \cap [l_i(s), r_i(t)]) \right) + \left(\sum_{(i,j) \in \widetilde{\mathcal{G}}(e[s,t])} w_Q(Q \cap [\hat{l}_{i,e[s,t]}(s), \hat{r}_{j,e[s,t]}(t)]) \right) \\
&= \left(\sum_{i \in \mathcal{L}(e[s,t]) \setminus \mathcal{B}(e[s,t])} M_i((s,t)) \right) + \left(\sum_{(i,j) \in \widetilde{\mathcal{G}}(e[s,t])} L_i((s,t)) + D(i+1, j-1) + R_j((s,t)) \right).
\end{aligned}$$

Observe that $D(i,j)$ can be computed via a data structure, constructed beforehand, that first computes $d_i = \sum_{q \in Q_i} w_Q(q)$ for every i in total time $\mathcal{O}(|Q| \log n)$ and stores them in a balanced binary tree as leaves, where every inner node stores the sum of the values of its children. For every $i \leq j$ the value $D(i,j) = \sum_{i \leq m \leq j} d_m$ can then be returned in $\mathcal{O}(\log n)$ time by identifying in $\mathcal{O}(\log n)$ time all $\mathcal{O}(\log n)$ maximal subtrees

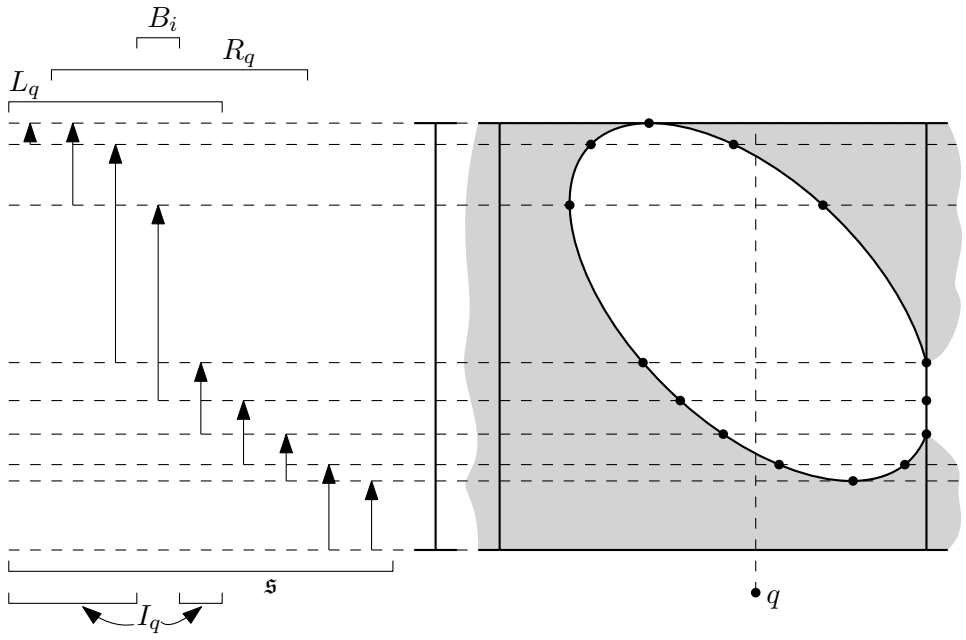


Figure 4.12: Construction of the set I_q encoding when $\hat{l}_{i,e[s,t]}(s) \leq q$ from Proof of Theorem 4.6.21 via the sets L_q and R_q encoding when $l_i(s) \leq q$ and $r_i(s) \geq q$ and the set B_i for all $(s,t) \in \mathfrak{s}$.

whose children lie in the interval $[i,j]$ and then returning the sum of the stored values in the root of each maximal subtree.

Next, we show that $L_i(\cdot)$ (resp. $R_j(\cdot)$ and $M_i(\cdot)$) can correctly be maintained when performing the sweep of $\mathfrak{s}$. To this end let

$$I_q = \{(s,t) \in \mathfrak{s} \mid q \in Q_i \text{ and } \hat{l}_{i,e[s,t]}(s) \neq \infty, \hat{l}_{i,e[s,t]} \leq q\}.$$

Refer to Figure 4.12. Note that the free space in every cell is convex, throughout the sweep-sequence $\mathfrak{s}$ the first indices are monotone, and the y-coordinates of the leftmost and rightmost points are stored in $\mathcal{E}(A_e)$. Hence the boundaries describing the contiguous subsets

$$L_q = \{(s,t) \in \mathfrak{s} \mid q \in Q_i \text{ and } l_i(s) \neq \infty, l_i(s) \leq q\} \text{ and}$$
$$R_q = \{(s,t) \in \mathfrak{s} \mid q \in Q_i \text{ and } l_i(s) \neq \infty, r_i(s) \geq q\}$$

can be computed in $\mathcal{O}(\log n)$ time for every $q \in Q$ via searches over $\mathfrak{s}$ starting at the y-coordinate corresponding to the left- and rightmost points of cell i. Similarly, the set $B_i = \{(s,t) \in \mathfrak{s} \mid i \text{ is bad for } e[s,t]\}$ is a contiguous subset of $\mathfrak{s}$ and all the boundaries of B_i can be computed in $\mathcal{O}(\log n)$ time. Then for any $q \in Q_i$

$$I_q = (L_q \cap (\mathfrak{s} \setminus B_i)) \cup ((\mathfrak{s} \setminus R_q) \cap B_i),$$

and thus I_q consists of $\mathcal{O}(1)$ contiguous disjoint subsets of $\mathfrak{s}$. Thus all sets I_q (represented by the $\mathcal{O}(1)$ boundaries of its contiguous subsets of $\mathfrak{s}$) can be computed in time $\mathcal{O}(|Q| \log n + n \log n)$. Further,

$$L_i((s,t)) = \sum_{q \in Q_i} \mathbb{1}_{q \in I_q} w_Q(q),$$

and hence $L_i(\cdot)$ can be maintained in total time $\mathcal{O}(|Q| + n \log n)$ after one initial computation of all I_q, by adding $w_Q(q)$ for $q \in Q_i$ whenever (s,t) enters the $\mathcal{O}(1)$ contiguous disjoint subsets of I_q, and subtracting $w_Q(q)$ for $q \in Q_i$ whenever (s,t) exits the $\mathcal{O}(1)$ contiguous disjoint subsets of I_q. Sorting the boundaries of all I_q preparing them for the maintenance of $L_i(\cdot)$ takes $\mathcal{O}(|Q| \log |Q|)$ time. Similarly $M_i(\cdot)$ and $R_j(\cdot)$ can be maintained.

Overall, the values $L_i(\cdot)$, $R_j(\cdot)$, $M_i(\cdot)$ and $D(i,j)$ are correctly maintained in total time $\mathcal{O}(|Q| \log |Q|)$ time such that they can be evaluated in $\mathcal{O}(\log n)$. By Corollary 4.6.20 there are only $\mathcal{O}(n)$ total updates to $\widetilde{\mathcal{G}}(\cdot)$ and $\mathcal{L}(\cdot) \setminus \mathcal{B}(\cdot)$. Hence, $w_Q(Q \cap \widehat{\mathrm{Cov}}_{\mathcal{D}}(\cdot))$ can be correctly maintained during a linear scan of $\mathfrak{s}$ by updating it whenever $L_i(\cdot)$, $R_j(\cdot)$, $M_i(\cdot)$, $\widetilde{\mathcal{G}}(\cdot)$, $\mathcal{L}(\cdot) \setminus \mathcal{B}(\cdot)$ or $\mathcal{B}(\cdot)$ change. Thus computing $w_Q(Q \cap \widehat{\mathrm{Cov}}_{\mathcal{D}}(e[s,t]))$ for all $(s,t) \in S$ takes $\mathcal{O}(|Q| \log |Q| + n \log n)$ time. □

4.7 Cubic Subtrajectory Covering

The goal of this section is the following theorem.

Theorem 4.7.1. *Let P be a polygonal curve of complexity n and let $\Delta > 0$ and $\ell \leq n$ be given. Let k_Δ be the size of the smallest set $C^* \subset \mathbb{X}^{d,\ell}$ such that $\bigcup_{c \in C^*} \mathrm{Cov}_P(c, \Delta) = [0,1]$. There is an algorithm that given P, Δ and ℓ computes a set $C \subset \mathbb{X}^{d,\ell}$ of size $\mathcal{O}(k_\Delta \log n)$ such that $\bigcup_{c \in C} \mathrm{Cov}_P(c, 4\Delta) = [0,1]$. Further, it does so in $\mathcal{O}(n^3 \log^2 n + k_\Delta n^2 \log^3 n)$ time using $\mathcal{O}(n^3 \log n)$ space.*

4.7.1 Molecular Intervals

The proxy coverage is defined by intervals starting and ending at boundaries of atomic intervals. Hence, akin to Observation 4.4.2, the set of all midpoints of atomic intervals $\mathcal{A}(\mathcal{D}_{4\Delta}(S,P))$ is a set of points $A \subset [0,1]$ of size $16n^3$ such that for any set C of Type (I)-, (II)-, and (III)-subcurves induced by $\mathcal{E}(\mathcal{D}_{4\Delta}(S,P))$ it holds that

$$\widehat{\mathrm{Cov}}_{\mathcal{D}_{4\Delta}(S,P)}(C) = [0,1] \iff A \subset \widehat{\mathrm{Cov}}_{\mathcal{D}_{4\Delta}(S,P)}(C).$$

Theorem 4.7.2. *Let P be a curve of complexity n, let Δ, and ℓ be given. Let S be a 2Δ-maximal simplification of P. Let $A \subset [0,1]$ be the set of midpoints of atomic intervals of the 4Δ-free space of S and P. Let E be the set of y-coordinates of $\mathcal{E}(\mathcal{D}_{4\Delta}(S,P))$. Let k_Δ be the size of a smallest set $C^* \subset \mathbb{X}^{d,\ell}$ such that $\bigcup_{c \in C^*} \mathrm{Cov}_P(c, \Delta) = [0,1]$. Any algorithm that iteratively adds the curve c among $\mathcal{C}_S(E)$ to R maximizing*

$$\left| \left\{ a \in A \middle| a \in \left(\widehat{\mathrm{Cov}}_{\mathcal{D}_{4\Delta}(S,P)}(c) \setminus \left(\bigcup_{r \in R} \widehat{\mathrm{Cov}}_{\mathcal{D}_{4\Delta}(S,P)}(r) \right) \right) \right\} \right|,$$

terminates after $16(\ln |A| + 1)k_\Delta$ iterations.

Proof. Let $\mathcal{D}$ be the 4Δ-free space of S and P. By Theorem 4.6.11 there is a set C_1 of size $16k_\Delta$ in $\mathcal{C}_S(E)$ such that $\widehat{\mathrm{Cov}}_{\mathcal{D}}(C_1) = [0,1]$. Hence, by standard greedy set cover algorithm arguments, the algorithm terminates after $(\ln |A| + 1)16k_\Delta$ iterations, returning a set $C \subset \mathcal{C}_S(E)$ of size $(\ln |A| + 1)16k_\Delta$ such that $\widehat{\mathrm{Cov}}_{\mathcal{D}}(C) = [0,1]$. □

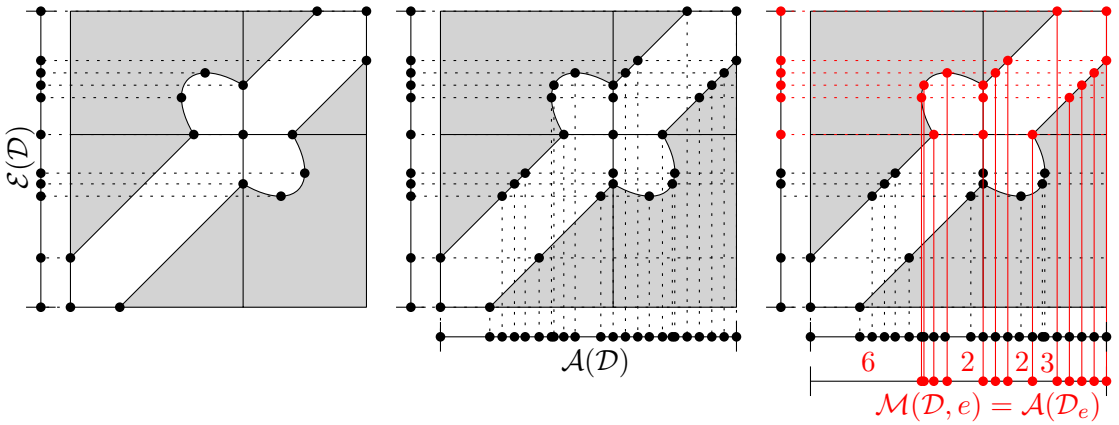


Figure 4.13: Construction of both the atomic and molecular intervals for an edge e based on the free space $\mathcal{D}$. If a molecular interval contains more than one atomic interval, the number of atomic intervals contained in a molecular intervals is shown.

Observation 4.7.3. *Let $\mathcal{D}$ be the 4Δ-free space of two polygonal curves S and P. For any edge e of S it holds that $\mathcal{E}(\mathcal{D}_e) \subset \mathcal{E}(\mathcal{D})$ and as such the atomic intervals of e and P partition the atomic intervals of S and P, where each atomic interval of e and P can be described as a contiguous subset of atomic intervals of S and P.*

To clarify the difference of atomic intervals $\mathcal{A}(\mathcal{D}_{4\Delta}(S, P))$ of the entire free space and atomic intervals $\mathcal{A}(\mathcal{D}_{4\Delta}(e, P))$ of the free space restricted to some edge e of S, we will call the latter ***molecular intervals***, where every molecular interval is the union of (usually) multiple atomic intervals, and denote them by $\mathcal{M}(\mathcal{D}_{4\Delta}(S, P), e)$ (Figure 4.13).

4.7.2 The Algorithm

The algorithm can be seen in Algorithm 6. It repeatedly calls the subroutine from Theorem 4.6.21 to identify the element in $\mathcal{C}_S(E)$ maximizing the number of midpoints of atomic intervals. It does so via the midpoints of molecular intervals weighted by the number of atomic intervals contained within each molecular interval.

Lemma 4.7.4. *Let $\mathcal{D}$ be the 4Δ-free space of S and P. Let $A \subset [0,1]$ be a set. Let e be an edge of S. Let $W_e = \{(s,t) \in \mathcal{M}(\mathcal{D}, e) \mid |A \cap m| \neq 0\}$ together with $w(m) = |A \cap m|$ and the midpoint p_m of m for every $m \in W_e$ be given. Let $\mathfrak{s} \in \mathfrak{S}_e$ be a sweep-sequence. Then $\sum_{m \in W_e} \mathbb{1}[p_m \in \widehat{\mathrm{Cov}}_{\mathcal{D}}(e[s,t])] w(m) = |A \cap \widehat{\mathrm{Cov}}_{\mathcal{D}}(e[s,t])|$ can be computed for every $(s,t) \in \mathfrak{s}$ in total time $\mathcal{O}(|W_e| \log n + n \log n)$.*

Proof. This is an immediate consequence of Theorem 4.6.21 and the definition of the molecular intervals $\mathcal{M}(\mathcal{D}, e)$. □

Lemma 4.7.5. *At the beginning of each iteration of the **while**-loop in Line 10 of Algorithm 6 it holds that $A = A \setminus \bigcup_{r \in R} \widehat{\mathrm{Cov}}_{\mathcal{D}}(r)$ and $W_e = \{m \in \mathcal{M}(\mathcal{D}, e) \mid |A \cap m| \neq 0\}$ and for every $m \in W_e$ it holds that $w(m) = |A \cap m|$. Further it holds that for every Type (II)- and (III)-subcurve π (stored in some sweep sequence) the weight $\widehat{w}(\pi)$ coincides with $|A \cap \widehat{\mathrm{Cov}}_{\mathcal{D}}(\pi)|$.*

Algorithm 6 Covering a finite subset of $[0,1]$

procedure CoverA($\mathcal{D}$,$A \subset [0,1]$,$\{\mathfrak{S}_e \mid \text{edge } e \text{ of } S\}$,$\{W_e = \{m \in \mathcal{M}(\mathcal{D}, e) \mid |A \cap m| \neq 0\} \mid \text{edge } e \text{ of } S\}$)
  Compute all Type (I)-curves and their proxy coverage $\widehat{\mathrm{Cov}}_{\mathcal{D}}(\cdot)$
  Store A in a balanced binary tree
  For every edge e of S endow $m \in W_e$ with weight $w(m) = |A \cap m|$ and the midpoint p_m of m
  For every edge e of S store W_e with its endowed data in a balanced binary tree
  for edge e of S **do**
    for $\mathfrak{s} \in \mathfrak{S}_e$ **do**
      For every $(s,t) \in \mathfrak{s}$ let $\widehat{w}(e[s,t]) = \sum_{m \in W_e} \mathbb{1}[p_m \in \widehat{\mathrm{Cov}}_{\mathcal{D}}(e[s,t])] w(m)$
  $R \leftarrow \emptyset$
  while $A \neq \emptyset$ **do**
    for every Type (I)-curve π compute $\widehat{w}(\pi) = |A \cap \widehat{\mathrm{Cov}}_{\mathcal{D}}(\pi)|$
    Identify Type (I)-, (II)- or (III)-subcurve c^* maximizing $w(c^*)$
    $R \leftarrow R \cup \{c^*\}$, $A' = A \cap \widehat{\mathrm{Cov}}_{\mathcal{D}}(c^*)$, update tree storing A via $A \leftarrow A \setminus A'$
    for edge e of S **do**
      compute $W'_e = \{m \in W_e \mid |m \cap A'| \neq 0\}$
      For every $m \in W'_e$ compute $w'(m) = |A' \cap m|$
      for $\mathfrak{s} \in \mathfrak{S}_e$ **do**
        For every $(s,t) \in \mathfrak{s}$ let $\widehat{w}'(e[s,t]) = \sum_{m \in W'_e} \mathbb{1}[p_m \in \widehat{\mathrm{Cov}}_{\mathcal{D}}(e[s,t])] w'(m)$
        $\widehat{w}(e[s,t]) \leftarrow \widehat{w}(e[s,t]) - \widehat{w}'(e[s,t])$ for every $(s,t) \in \mathfrak{s}$
      for $m \in W'_e$ **do**
        $w(m) \leftarrow w(m) - w'(m)$
        If $w(m) = 0$ remove m from W_e updating its tree
  return R

Proof. Observe that at the beginning of the first iteration this is true, as $R = \emptyset$. Let c^* be the element added to R in some iteration of the **while**-loop. As $A' = A \cap \widehat{\mathrm{Cov}}_{\mathcal{D}}(c^*)$, A is correctly updated. As initially every set $W_e = \{m \in \mathcal{M}(\mathcal{D}, e) \mid |A \cap m| \neq 0\}$ and for every $m \in W_e$ the weight $w(m)$ is computed correctly, W'_e is computed correctly, and for every $m \in W_e$ it holds that $w'(m) = |A' \cap m| \neq 0 \iff m \in W'_e$. Hence W_e is updated correctly at the end of the **while**-loop. Further, $\widehat{w}'(e[s,t]) = |A' \cap \widehat{\mathrm{Cov}}_{\mathcal{D}}(e[s,t])|$ and thus at the end of the **while**-loop $\widehat{w}(e[s,t]) = |(A \setminus A') \cap \widehat{\mathrm{Cov}}_{\mathcal{D}}(e[s,t])|$, and hence $\widehat{w}(e[s,t]) = |A \cap \widehat{\mathrm{Cov}}_{\mathcal{D}}(e[s,t])|$ at the beginning of the next loop. □

Lemma 4.7.6. *Let P be a polygonal curve of complexity n and let $\Delta > 0$ and $\ell \leq n$ be given. Let S be a simplification of P. Let $\mathcal{D}$ be the 4Δ-free space of S and P. Let $A \subset [0,1]$ be a set of $\mathcal{O}(n^3)$ points. For every edge e of S let $\mathfrak{S}_e$ be $\mathcal{O}(\log n)$ sweep-sequences, each of length $\mathcal{O}(n)$, together containing all $\mathcal{O}(n \log n)$ Type (II)- and (III)-subedges of e, and let W_e be the set of molecular intervals m such that $|A \cap m| \neq 0$. Let k_Δ be the size of the smallest set $\mathcal{C}^* \subset \mathbb{X}^{d,\ell}$ such that $\bigcup_{c \in \mathcal{C}^*} \mathrm{Cov}_P(c, \Delta) = [0,1]$. The algorithm CoverA from Algorithm 6 computes a set $\mathcal{C} \subset \mathbb{X}^{d,\ell}$ of size $(48 \ln(n) + 64) k_\Delta$ such that $A \subset \widehat{\mathrm{Cov}}_{A_S}(C)$. Further, it does so in time*

$$\mathcal{O}(n^2 \ell \log n^2 + |A| \log n + k_\Delta n^2 \log^3 n + \log n \sum_e |W_e|).$$

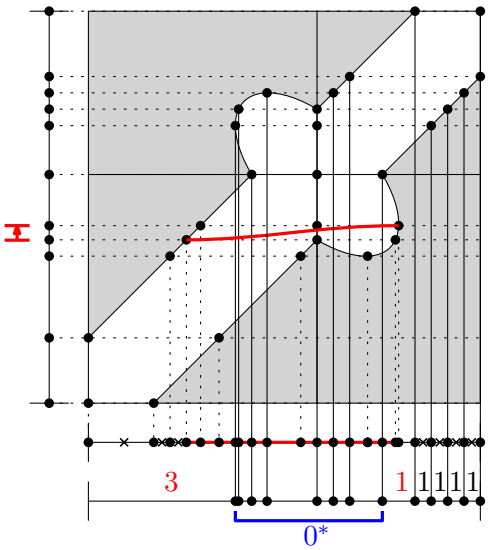


Figure 4.14: Illustration of $\widehat{w}(\cdot)$ after a curve has been added to the partial solution R. Only constantly many molecular intervals per cell are updated but do not have $\widehat{w}(\cdot)$ set to 0 (in red). All intervals that have $\widehat{w}(\cdot)$ set to 0 are updated for the final time and can be removed from W_e. These are further marked with a *.

Proof. By Observation 4.4.2, Theorem 4.7.2, and Lemma 4.7.5 the algorithm correctly outputs a solution as claimed after $\mathcal{O}(k_\Delta \log n)$ iterations of the **while**-loop in Line 10.

By Lemma 4.4.6, computing all Type (I)-curves and their proxy coverage takes a total of $\mathcal{O}(n^2 \ell \log n \log \ell)$ time. As A is stored in a tree, computing $w(m) = |A \cap m|$ for every $m \in W_e$ for every e takes total time $\mathcal{O}(\log n \sum_e |W_e|)$. By Theorem 4.6.21 computing and storing $\widehat{w}(e[s,t])$ for every $(s,t) \in \mathfrak{s}$ for every $\mathfrak{s} \in \mathfrak{S}_e$ for an edge e of S takes a total of $\mathcal{O}(|W_e| \log n + n^2 \log^2 n)$ time. Thus the precomputation steps before the **while**-Loop take a total of $\mathcal{O}(n^2 \ell \log n \log \ell + \sum_e (|W_e| \log n + n \log n))$ time. Assume that W_e and W'_e are sorted.

Now, for the i^{th} iteration of the **while**-Loop let R_i denote the set stored in R and let A_i denote the set stored in A coinciding with $A \setminus \widehat{\text{Cov}}_{\mathcal{D}}(R_i)$. Let r_i be the element added to R in this iteration. Let further $W_{e,i}$ denote the set of molecular intervals stored in W_e, and similarly $W'_{e,i}$ be the set of molecular intervals stored in W'_e. Observe that $W_{e_i} = \{m \in \mathcal{M}(\mathcal{D}, e) \mid |m \cap A_i| \neq 0\}$ and $W'_{e_i} = \{m \in \mathcal{M}(\mathcal{D}, e) \mid |m \cap (A_i \cap \widehat{\text{Cov}}_{\mathcal{D}}(r_i))| \neq 0\}$. The coverage $\widehat{\text{Cov}}_{\mathcal{D}}(r_i)$ consists of at most n intervals and hence there are at most n molecular intervals $m \in W_{e_i}$ such that neither $m \subset \widehat{\text{Cov}}_{\mathcal{D}}(r_i)$ nor $m \cap \widehat{\text{Cov}}_{\mathcal{D}}(r_i) = \emptyset$ holds. Hence $|W'_{e,i}| \leq |W_{e_i} \setminus W_{e,i+1}| + n$ (refer to Figure 4.14). Thus $W'_{e,i}$ can be computed in time $\mathcal{O}(|W_{e_i} \setminus W_{e,i+1}| + n \log n)$. By Theorem 4.6.21, computing $\widehat{w}'(e[s,t])$ for a fixed edge e, fixed $\mathfrak{s} \in \mathfrak{S}_e$, and every $(s,t) \in \mathfrak{s}$ can be done in $\mathcal{O}(|W'_{e,i}| + n \log n) = \mathcal{O}(|W_{e_i} \setminus W_{e,i+1}| + n \log n)$ time. After having computed $\widehat{w}'(e[s,t])$ for the edge e, every $\mathfrak{s} \in \mathfrak{S}_e$, and every $(s,t) \in \mathfrak{s}$, the update to W_{e_i} and $\widehat{w}(e[s,t])$ resulting in $W_{e,i+1}$ takes $\mathcal{O}(|W_{e_i} \setminus W_{e,i+1}| + n \log n)$. Updating A_i takes total time $\mathcal{O}(|A_i \setminus A_{i+1}| \log n + n \log n)$.

Hence, k iterations of the **while**-Loop take time in

$$\mathcal{O}\left(\sum_{i=0}^{k}\left(|A_i \setminus A_{i+1}| \log n + n \log n + \left(\sum_e \sum_{\mathfrak{s} \in \mathfrak{S}_e} (|W_{e,i} \setminus W_{e,i+1}| + n \log n)\right)\right)\right)$$

Algorithm 7 Cubic Subtrajectory Covering

1: **procedure** ComputeSubtrajectoryCovering(P,Δ)
2: Compute $(2\Delta, 0)$-maximal simplification S of P
3: Compute $\mathcal{D} = \mathcal{D}_{4\Delta}(S, P)$ and with it $\mathcal{E}(\mathcal{D})$ and $\mathcal{A}(\mathcal{D})$
4: Compute the set E of y-coordinates of $\mathcal{E}(\mathcal{D})$
5: Compute the set $\mathcal{C}_S(E)$ of Type (I)-, (II)- and (III)-subcurves of S
6: Store Type (II)- and (III)-subcurves of S in sweep sequences $\{\mathfrak{S}_e \mid \text{edge } e \text{ of } S\}$
7: Compute midpoints A of atomic intervals $\mathcal{A}(\mathcal{D})$
8: Compute molecular intervals $\mathcal{M}(\mathcal{D}, e)$ for every edge e of S based on $\mathcal{D}_{4\Delta}(S, P)$
9: **return** Covera($\mathcal{D}$,A,$\{\mathfrak{S}_e \mid \text{edge } e \text{ of } S\}$,$\{\mathcal{M}(\mathcal{D}, e) \mid \text{edge } e \text{ of } S\}$)

$$
\begin{aligned}
&= \mathcal{O}\left(|A|\log n + kn\log n + \sum_{i=0}^{k}\left(\log n \sum_{e}\left(|W_{e,i} \setminus W_{e,i+1}| + n\log n\right)\right)\right)\\
&= \mathcal{O}\left(|A|\log n + kn\log n + kn^2\log^2 n + \log n \sum_{e}\sum_{i=0}^{k}\left(|W_{e,i} \setminus W_{e,i+1}|\right)\right)\\
&= \mathcal{O}\left(|A|\log n + kn^2\log^2 n + \log n \sum_{e}|W_e|\right).
\end{aligned}
$$

By Theorem 4.7.2 the algorithm terminates after $\mathcal{O}(k_\Delta \log n)$ rounds, and hence its running time is in $\mathcal{O}(n^2\ell\log n^2 + |A|\log n + k_\Delta n^2\log^3 n + \log n\sum_e |W_e|)$ time, concluding the proof. □

Theorem 4.7.1. *Let P be a polygonal curve of complexity n and let $\Delta > 0$ and $\ell \leq n$ be given. Let k_Δ be the size of the smallest set $C^* \subset \mathbb{X}^{d,\ell}$ such that $\bigcup_{c\in C^*} \mathrm{Cov}_P(c, \Delta) = [0, 1]$. There is an algorithm that given P, Δ and ℓ computes a set $C \subset \mathbb{X}^{d,\ell}$ of size $\mathcal{O}(k_\Delta \log n)$ such that $\bigcup_{c\in C} \mathrm{Cov}_P(c, 4\Delta) = [0, 1]$. Further, it does so in $\mathcal{O}(n^3\log^2 n + k_\Delta n^2\log^3 n)$ time using $\mathcal{O}(n^3 \log n)$ space.*

Proof. Consider Algorithm 7. The algorithm first computes a simplification S of P. It then computes the 4Δ-free space of S and P and with it $\mathcal{E}(\mathcal{D}_{4\Delta}(S, P))$ as well as $\mathcal{A}(\mathcal{D}_{4\Delta}(S, P))$ in $\mathcal{O}(n^3 \log n)$ time. Finally, in total time $\mathcal{O}(n^2\log^2 n)$ time per edge e, it computes all molecular intervals $\mathcal{M}(\mathcal{D}_{4\Delta}(S, P), e)$.

Lastly we compute A of size $\mathcal{O}(n^3)$ via Observation 4.4.2, storing their midpoints in a balanced binary tree, and with it compute the sets $W_e = \{m \in \mathcal{M}(\mathcal{D}_{4\Delta}(S, P), e) \mid |m \cap A| \neq 0\}$ for every edge e of S, with $\sum_e |W_e| = \mathcal{O}(n^3)$. Then Lemma 4.7.6 concludes the proof. □

4.8 Improved Subtrajectory Covering

In this section we want to reduce the dependency on n. As in the previous section, we assume P to be a polygonal curve of complexity n, S a 2Δ-maximal simplification and $\mathcal{D}$ the 4Δ-free space of S and P. Let further E be the set of y-coordinates of $\mathcal{E}(\mathcal{D})$.

Observe that there are two steps in the algorithm described in Theorem 4.7.1 that can take up to $\widetilde{\mathcal{O}}(n^3)$ time that are somewhat inevitable. The first is the computation of the discretization, that is, $\mathcal{A}(\mathcal{D})$ may be of cubic size, and similarly the n coarsenings

via molecular intervals $\mathcal{M}(\mathcal{D}, e)$ in total take $\mathcal{O}(n^3)$ time to compute. The later updates depend on the number of molecular intervals with non-zero weight. In this section we introduce a subproblem which is closely related to rank-selection [FJ84] in matrices in order to pick $\mathcal{O}(n^\alpha)$ interval boundaries of $\mathcal{A}(\mathcal{D})$ which are 'well-separated' in time $\widetilde{\mathcal{O}}(n^{1+\alpha})$. These boundaries are separated in the sense that between two consecutive boundaries (defining a coarse interval) there are no more than $\mathcal{O}(n^{3-\alpha})$ intervals of $\mathcal{A}(\mathcal{D})$. Morally, the midpoints of coarse intervals play a similar role to coresets (compare Chapter 3), where a solution covering the midpoints is already 'pretty good'.

The algorithm will in a first round solve the set cover instance defined by the midpoints of these coarse intervals. The coverage of such a solution can be described by the union of $\mathcal{O}(k_\Delta n \log n)$ intervals, and thus at most $\mathcal{O}(k_\Delta n \log n)$ of the coarse intervals are not yet covered. As no coarse interval contains too many intervals of $\mathcal{A}(\mathcal{D})$, at most $\mathcal{O}(k_\Delta n^{4-\alpha})$ molecular intervals are not yet covered by the solution. These we determine in an output-sensitive manner, and then cover with the algorithm described in Section 4.7. The result is an algorithm with running time in $\widetilde{\mathcal{O}}(k_\Delta n^{5/2})$ by setting $\alpha = 3/2$. By multiplicatively testing for k_Δ and being slightly more careful how α is chosen the running time improves to $\widetilde{\mathcal{O}}(\sqrt{k_\Delta} n^{\frac{5}{2}})$ which is truly subcubic if $k_\Delta \in \mathcal{O}(n^{1-\varepsilon})$.

4.8.1 Subquadratic Coarsening of Atomic Intervals

We now describe a subroutine used for constructing the aforementioned coarse intervals.

Theorem 4.8.1. *Let n lists L_i be given, each containing m_i sorted values such that all values are distinct, for every list L_i and for every j identifying the item at position j takes $\mathcal{O}(\log m_i)$ time, and for given v determining the maximal index j such that the j^{th} item is less than v takes $\mathcal{O}(\log m_i)$ time. Then for every $K \leq \sum_i m_i$ in $\mathcal{O}(Kn \log \max_i m_i)$ time one can determine $\mathcal{O}(K)$ values $v_1, \ldots$ such that*

1. *$v_i < v_{i+1}$ for all i,*
2. *for every $x \in \bigcup_i L_i$ there is an i such that $x \in [v_i, v_{i+1}]$ and*
3. *$|[v_i, v_{i+1}] \cap \bigcup_i L_i| = O\left(\frac{\sum_i m_i}{K}\right)$ for all i.*

Proof. Let $N = \sum_i m_i$. Initially, identify the minimal v and maximal $\hat{v}$ value in $\bigcup_i L_i$ in $\mathcal{O}(n \log \max_i m_i)$ time. We define a subproblem instance via v and $\hat{v}$ by n intervals of indices describing the interval of L_i that lies between v and $\hat{v}$. Initially all intervals have the form $[1, m_i]$. The goal is to iteratively split this subproblem into smaller subproblems until each subproblem contains at most N/K values.

Let now a subproblem instance be given, defined by values l, r, and n intervals with l_i values for $i \leq n$. If $\sum_i l_i \leq N/K$, then there is no need to split it.

If instead $\sum_i l_i \leq 20n$, then we simply sort all values in the n intervals in total time $\mathcal{O}(n \log n)$ time and identify the value m splitting the sorted list into half. From this construct two subproblems in $\mathcal{O}(n \log n)$ time where the number of values in each subproblem is $\lceil \sum_i l_i/2 \rceil$ and $\lfloor \sum_i l_i/2 \rfloor$ respectively.

If $\sum_i l_i > 20n$, then for every list L_i, identify 5 values splitting L_i into 5 pieces all containing either $\lfloor l_i/5 \rfloor$ or $\lceil l_i/5 \rceil$ values. This results in 5 intervals, where each interval is endowed with a weight w_I corresponding to the number of values from L_i that lie in it. This results in a total set $\mathcal{I}$ of $\mathcal{O}(n)$ intervals which can be computed in $\mathcal{O}(n \log \max_i m_i)$ time, and so can its arrangement. For every interval in the arrangement pick its midpoint

as the potential splitting value, and collect them in a sorted list C. For every $c \in C$ define

$$w(c) = \sum_{I \in \mathcal{I}, I < c} w_I - \sum_{I \in \mathcal{I}, I > c} w_I.$$

All these values can be computed in $\mathcal{O}(n \log \max_i m_i)$ time as $w(c)$ and $w(c')$ differ by at most two w_I if c and c' lie in neighboring cells of the arrangement, thus we can sweep through the arrangement to compute these values. Now identify the last $c \in C$ such that $w(c) \geq 0$, that is, in particular $w(c) \leq \lceil \max_i l_i/5 \rceil \leq \lceil \sum_i l_i/5 \rceil \leq \sum_i l_i/4$. And hence

$$\sum_{I \in \mathcal{I}, I > c} w_I \leq \sum_{I \in \mathcal{I}, I < c} w_I \leq \sum_{I \in \mathcal{I}, I > c} w_I + \sum_i l_i/4$$

Further observe that

$$\begin{aligned} \sum_{I \in \mathcal{I}, I < c} w_I + \sum_{I \in \mathcal{I}, I > c} w_I &\geq \sum_{I \in \mathcal{I}} w_I - \sum_{I \in \mathcal{I}, c \in I} w_I \geq \sum_i l_i - \sum_i \lceil l_i/5 \rceil \\ &\geq \sum_i l_i - \sum_i l_i/5 - n \geq \sum_i l_i - \sum_i l_i/4 = 3/4 \sum_i l_i \end{aligned}$$

Thus we conclude that both $\sum_{I \in \mathcal{I}, I > c} w_I \geq 1/4 \sum_i l_i$ and $\sum_{I \in \mathcal{I}, I < c} w_I \geq 1/4 \sum_i l_i$. Hence splitting at c will result in two subproblems, each of which contains at most $3/4 \sum_i l_i$ points. These subproblems can similarly be constructed in $\mathcal{O}(n \log \max m_i)$ time. As each split decreases the number of values in the subproblem by a constant fraction, after $\mathcal{O}(K)$ splits each subproblem will have at most $\mathcal{O}\left(\frac{\sum_i m_i}{K}\right)$ values concluding the proof. □

Lemma 4.8.2. *For every $\alpha \in [0, 3]$ in $\mathcal{O}(n^{1+\alpha} \log n)$ time one can determine $\mathcal{O}(n^\alpha)$ intervals partitioning $\mathcal{A}(\mathcal{D})$, each containing at most $\mathcal{O}(n^{3-\alpha})$ intervals of $\mathcal{A}(\mathcal{D})$.*

We will call a set of intervals partitioning $\mathcal{A}(\mathcal{D})$ such that any interval contains $\mathcal{O}(n^{3-\alpha})$ intervals of $\mathcal{A}(\mathcal{D})$ a set of ***α-coarse*** intervals.

Proof. For a fixed edge e we will implicitly provide two lists L_u and L_l such that

1. $L_u \cup L_l$ coincides with all boundaries of molecular intervals in $\mathcal{M}(\mathcal{D}, \varepsilon)$,
2. $|L_u|$ and $|L_l|$ is known,
3. for either list the value at position j can be computed in $\mathcal{O}(\log n)$ time, and
4. for any value v the maximal index j such that the value at position j is less than v can be computed in $\mathcal{O}(\log n)$ time.

To construct L_u we compute in every cell the at most 4 closures of the at most 4 pieces of the free space boundary (i.e., an ellipse) that lie in the interior of the cell and are x- and y-monotone. From this we pick the at most 2 pieces that lie above (in y-direction) the free space. Refer to Figure 4.15. Observe that for these $\mathcal{O}(n)$ pieces from all cells it holds that their projections onto the x-axis are interior-disjoint and thus sorted. For every piece we compute the interval and number of elements of E_e in $\mathcal{O}(\log n)$ time that lie in the projection onto the y-axis of each piece. These form the list L_u. Observe that we can compute $|L_u|$ in $\mathcal{O}(n)$ time and store it. Further, with the information imbued on each piece property 3 and property 4 also hold: to find for example the boundary at position j among the boundaries induced by L_u, we first search over the $\mathcal{O}(n)$ pieces identifying the piece inducing the j^{th} boundary, and then search over the corresponding

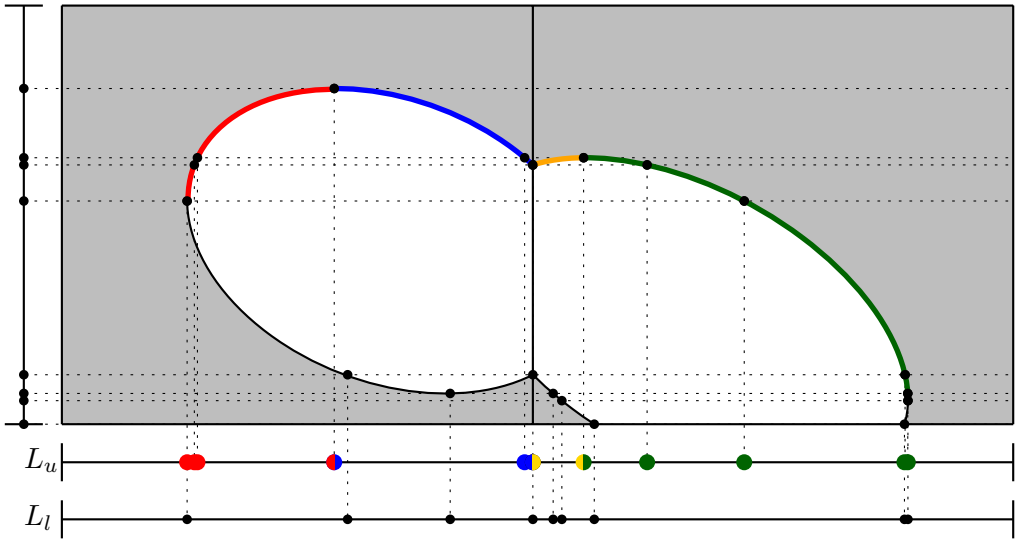


Figure 4.15: Illustration to proof of Lemma 4.8.2. The boundary of the free space in every cell is split into its four monotone pieces. If the y-coordinates of the extremal points are sorted, then the boundaries of molecular intervals induced by extremal points intersecting the upper pieces (in red, blue, orange, and green) of the two cells are already sorted as well and can be binary searched over without explicit computation of all molecular interval boundaries.

interval of E_e in $\mathcal{O}(\log n)$ total time. We similarly compute L_l via the lower, instead of the upper pieces of the boundary. Observe that the union over both lists form $\mathcal{M}(\mathcal{D}, e)$. Further, all values that occur with multiplicity in each list can be made unique with two points from different cells being compared first w.r.t their x-coordinate and second w.r.t. the lexicographic order of the indices of their respective cells.

Computing this for every edge in $\mathcal{O}(n^2 \log n)$ time leaves us with $\mathcal{O}(n)$ lists each containing at most $\mathcal{O}(n^2)$ values without any multiplicities. Thus, applying Theorem 4.8.1 with $K \leftarrow n^\alpha$ results in $\mathcal{O}(n^\alpha)$ interval boundaries of $\mathcal{A}(\mathcal{D})$ in $\mathcal{O}(n^{1+\alpha} \log n)$ time such that between two such values there are at most $\mathcal{O}(n^{3-\alpha})$ intervals of $\mathcal{A}(\mathcal{D})$. □

Observation 4.8.3. *We remark that the implicitly provided lists L_l and L_u for the edge e from the proof of Lemma 4.8.2 can be constructed in $\mathcal{O}(n \log n)$ time and can be used, to identify for a given point $p \in [0,1]$ the molecular interval in $\mathcal{M}(\mathcal{D}, e)$ containing p in $\mathcal{O}(\log n)$ time.*

4.8.2 Subtrajectory Covering Without Explicit Atomic Intervals

The final piece of the puzzle is the output-sensitive identification of all atomic and molecular intervals that are not part of some initial coarse solution that we compute based on coarse intervals.

Lemma 4.8.4. *For every $\alpha \in [0,3]$ and every K let $C \subset \mathcal{C}_S(E)$ be a solution covering all midpoints of α-coarse intervals of size $\mathcal{O}(K \log n)$. Then there are at most $\mathcal{O}(Kn^{4-\alpha} \log n)$ atomic intervals and $\mathcal{O}(Kn^{4-\alpha} \log n + Kn^2 \log n)$ molecular intervals that are not contained in $\widehat{\mathrm{Cov}}_{\mathcal{D}}(C)$. Further they can be computed in $\mathcal{O}(Kn^{4-\alpha} \log^2 n + Kn^2 \log^2 n)$ time.*

Proof. First observe that there are at most $\mathcal{O}(Kn^{4-\alpha}\log n)$ atomic intervals that are not in $\widehat{\mathrm{Cov}}_{\mathcal{D}}(C)$, as $\widehat{\mathrm{Cov}}_{\mathcal{D}}(C)$ does not contain at most $\mathcal{O}(Kn\log n)$ α-coarse intervals and hence at most $\mathcal{O}(Kn^{4-\alpha}\log n)$ atomic intervals. Similarly, the set of molecular intervals that are not contained in $\widehat{\mathrm{Cov}}_{\mathcal{D}}(C)$ consists of all molecular intervals that have a boundary that is not in $\widehat{\mathrm{Cov}}_{\mathcal{D}}(C)$, or that contain some interval boundary of $\widehat{\mathrm{Cov}}_{\mathcal{D}}(C)$. Thus in total there are $\mathcal{O}(Kn^{4-\alpha}\log n + Kn^2\log n)$ such molecular intervals.

To compute them observe that $\widehat{\mathrm{Cov}}_{\mathcal{D}}(C)$ consists of $\mathcal{O}(Kn\log n)$ disjoint intervals, and thus so does $[0,1]\setminus\widehat{\mathrm{Cov}}_{\mathcal{D}}(C)$. Let $\mathcal{I}$ be the set of $\mathcal{O}(Kn\log n)$ intervals that result from the disjoint intervals in $[0,1]\setminus\widehat{\mathrm{Cov}}_{\mathcal{D}}(C)$ after splitting them at every vertex parameter of P. That is, any interval in $\mathcal{I}$ lies on only one edge. The intervals in $\mathcal{I}$ are not necessarily disjoint, but the intersection of any two intervals is in at most one point. Let $[a,b]\in\mathcal{I}$ lie on edge i of P. We compute the sets represented as $\mathcal{O}(1)$ contiguous intervals in E_e

1. $S_{e,1}=\{l_i(s)\mid s\in E_e : l_i(s)\in[a,b]\}$,
2. $S_{e,2}=\{r_i(s)\mid s\in E_e : r_i(s)\in[a,b]\}$,
3. $S_{e,3}=\{l_i(s)\mid s\in E_e : l_i(s)\le a\}$ and $S_{e,4}=\{r_i(s)\mid s\in E_e : r_i(s)\le a\}$, and
4. $S_{e,5}=\{l_i(s)\mid s\in E_e : l_i(s)\ge b\}$ and $S_{e,6}=\{r_i(s)\mid s\in E_e : r_i(s)\ge b\}$.

in total time $\mathcal{O}(\log n)$ via binary searches over E_e. Then the union $\{a,b\}\cup\bigcup_{e\in S}(S_{e,1}\cup S_{e,2})$ is precisely the set of all interval boundaries of atomic intervals that lie in $[a,b]$. These can thus be computed in $\mathcal{O}((|[a,b]\cap\mathcal{A}(\mathcal{D})|+1)\log n)$ time. Thus overall computing all atomic intervals that intersect $[0,1]\setminus\widehat{\mathrm{Cov}}_{\mathcal{D}}(C)$ can be done in $\mathcal{O}(Kn^{4-\alpha}\log^2 n + Kn\log^2 n)$ time.

Similarly $\{\max(S_{e,3}\cup S_{e,4}),\max(S_{e,5}\cup S_{e,6})\}\cup S_{e,1}\cup S_{e,2}$ contain all interval boundaries of molecular intervals of e and P that intersect $[a,b]$ and can be computed in time $\mathcal{O}((|\{m\in\mathcal{M}(\mathcal{D},e)\mid m\subset[a,b]\}|+1)\log n)$ time. Thus computing all molecular intervals that intersect $[0,1]\setminus\widehat{\mathrm{Cov}}_{\mathcal{D}}(C)$ can be done in $\mathcal{O}(Kn^{4-\alpha}\log^2 n + Kn^2\log^2 n)$ time. □

Theorem 4.2.1. *There is a $(96\ln(n)+128,4)$-approximation for SC. Given a curve P of complexity n, together with values $\Delta>0$ and $\ell\le n$, its running time is in $\mathcal{O}\left(\left(n^2\ell+\sqrt{k_\Delta}n^{\frac{5}{2}}\right)\log^2 n\right)$, where k_Δ is the size of the smallest subset $C^*\subset\mathbb{X}_\ell^d$ such that $\mathrm{Cov}_P(C^*,\Delta)=[0,1]$.*

Proof. Consider Algorithm 8. Line 1 to 6 take time $\mathcal{O}(n^2\ell\log n\log\ell)$ by Lemma 4.4.6, the fact that the extremal points in all cells of the free space can be computed in $\mathcal{O}(n^2)$ time, and Lemma 4.6.2.

Now let K be as in the beginning of some iteration of the **while**-Loop in Line 8. We show that in one iteration of the **while**-Loop the algorithm computes a solution of size $2k_\Delta\log n$ or correctly determines that $K<k_\Delta$ and sets $K\leftarrow 2K$. Initially, $K=1\le k_\Delta$. Let $\alpha=\frac{3}{2}+\frac{\log K}{2\log n}+\frac{\log\log n}{\log n}$ and let $\lambda=(48\ln n+64)\ge 16(\ln(16n^3)+1)$. By Lemma 4.8.2, the algorithm first computes a set of α-coarse intervals, and thus their midpoints A correctly. It then computes all molecular intervals containing some point of A via Observation 4.8.3. Next it attempts to cover them in λK rounds of CoverA from Algorithm 5. If it terminates within λK rounds, then CoverA returns a solution of size at most λk_Δ covering all midpoints of the α-coarse intervals. Otherwise $\lambda K<\lambda k_\Delta$ and in particular $K<k_\Delta$, in which case we correctly set $K\leftarrow 2k$ and restart the **while**-Loop.

Via Lemma 4.8.4, the algorithm then determines all uncovered atomic intervals and the molecular intervals which contain these points. Next, the algorithm invokes CoverA

Algorithm 8 Covering $[0,1]$ in subcubic time

procedure CoverAFast(P,Δ,ℓ)
  Compute a simplification S of P
  Compute the 4Δ-free space $\mathcal{D}$ of S and P
  Compute the set E of y-coordinates of $\mathcal{E}(\mathcal{D})$
  Compute the sweep-sequences $\mathfrak{S}_e$ of E_e for every edge e
  Compute the proxy coverage for every Type (I)-curve and provide them to CoverA from Algorithm 6
  $K \leftarrow 1$, covers $\leftarrow$ False, $\lambda \leftarrow (48 \ln |P| + 64)$
  while covers is False **do**
    $\alpha \leftarrow \frac{3}{2} + \frac{\log K}{2 \log n} + \frac{\log \log n}{\log n}$, covers $\leftarrow$ True and $R \leftarrow \emptyset$
    Compute a set of α-coarse intervals and from them their midpoints A
    Compute set of molecular intervals W_e with $w_A(\cdot) \neq 0$ for every edge e of S
    if CoverA($\mathcal{D}$,A,$\{\mathfrak{S}_e\}$,$\{W_e\}$) does not terminate after λK rounds **then**
      $K \leftarrow 2K$, covers $\leftarrow$ False
    else
      $R \leftarrow$ CoverA($\mathcal{D}$,A,$\{\mathfrak{S}_e\}$,$\{W_e\}$)
      Compute all atomic intervals in $\mathcal{A}(S,P) \setminus \widehat{\mathrm{Cov}}_{\mathcal{D}}(R)$ and their midpoints A
      Compute set of molecular intervals W_e with $w_A(\cdot) \neq 0$ for all edges e of S
      if CoverA($\mathcal{D}$,A,$\{\mathfrak{S}_e\}$,$\{W_e\}$) does not terminate after λK rounds **then**
        $K \leftarrow 2K$, covers $\leftarrow$ False
      else
        $R \leftarrow R \cup$ CoverA($\mathcal{D}$,A,$\{\mathfrak{S}_e\}$,$\{W_e\}$), covers $\leftarrow$ True
  return R

again, this time with the set of midpoints of uncovered atomic intervals, and the set of molecular intervals that contain at least one of these midpoints. If it terminates within λK rounds, then CoverA returns a solution of size at most λk_Δ covering all midpoints of the α-coarse intervals. Otherwise $\lambda K < \lambda k_\Delta$ and in particular $K < k_\Delta$, in which case we correctly set $K \leftarrow 2k$ and restart the **while**-Loop.

Thus, within the **while**-Loop, the algorithm correctly determines that either $K < k_\Delta$ and restarts with $K \leftarrow 2K$ or outputs a solution of size $(96 \ln(n) + 128) k_\Delta$. This ensures that the algorithm correctly computes a solution of claimed size.

As $K \in \mathcal{O}(k_\Delta)$ and thus $K \in \mathcal{O}(n)$, the running time of the **while**-Loop is

$$\begin{aligned}
&\mathcal{O}(n^{1+\alpha} \log n + K n^{4-\alpha} \log^3 n + K n^2 \log^2 n) \\
&= \mathcal{O}(n^{1+\alpha} \log n + n^{4-\alpha+\frac{\log K}{\log n}} \log^3 n + K n^2 \log^2 n) \\
&= \mathcal{O}(K^{\frac{1}{2}} n^{\frac{5}{2}} \log^2 n + K^{\frac{1}{2}} n^{\frac{5}{2}} \log^2 n + K n^2 \log^2 n) = \mathcal{O}(K^{\frac{1}{2}} n^{\frac{5}{2}} \log^2 n).
\end{aligned}$$

And thus overall the running time of the algorithm is

$$\begin{aligned}
&\mathcal{O}\left(n^2 \ell \log^2 n + \sum_{K=1}^{\log(k_\Delta)} \left(\left(2^K\right)^{\frac{1}{2}} n^{\frac{5}{2}} \log^2 n \right) \right) \\
&= \mathcal{O}\left(n^2 \ell \log^2 n + {k_\Delta}^{\frac{1}{2}} n^{\frac{5}{2}} \log^2 n \right).
\end{aligned}$$

□

Proposition 4.8.5. *If $k_\Delta = \mathcal{O}(n^{1-\varepsilon})$ and $\ell = \mathcal{O}(n^{1-\varepsilon})$ then the running time is subcubic, namely $\widetilde{\mathcal{O}}(n^{3-\varepsilon/2})$.*

4.9 Subtrajectory Coverage Maximization

The goal of this section is the following theorem.

Theorem 4.2.2. *Let $\varepsilon \in (0,1]$. There is an $(\frac{e-1}{16e}, 4+\varepsilon)$-approximation algorithm for SCM, where e is the base of the natural logarithm. Given a polygonal curve P of complexity n, $\Delta > 0$, $\ell \leq n$, and $k > 0$, its running time is in $\mathcal{O}((k+\ell)n^2\varepsilon^{-2}\log^2 n \log^2(\varepsilon^{-1}))$.*

Throughout this section we assume that we are given P and we have computed the simplification S via Theorem 4.4.4 and an $(1+\varepsilon, 4\Delta)$-free space A_S of S and P via Theorem 2.3.4 consisting of convex polygons of complexity $\mathcal{O}(\varepsilon^{-2})$ in each cell.

The quality of the algorithm hinges upon the following theorem.

Theorem 4.9.1. *Let P be a curve of complexity n, let Δ and ℓ be given. Let S be a simplification of P. Let E be the y-coordinates of the extremal points and vertices describing the polygon defining A_S in each cell. Any algorithm that iteratively adds the curve c among $\mathcal{C}_S(E)$ to R maximizing*

$$\lambda\left(\widehat{\mathrm{Cov}}_{A_S}(c) \setminus \left(\bigcup_{r\in R} \widehat{\mathrm{Cov}}_{A_S}(r)\right)\right),$$

computes after k rounds a set $R \subset \mathbb{X}_\ell^d$ such that for any other set C^* of cardinality k it holds that

$$\lambda(\mathrm{Cov}_P(R,(4+\varepsilon)\Delta)) \geq \frac{e-1}{16e}\lambda(\mathrm{Cov}_P(C^*,\Delta)).$$

Proof. Let $\textsc{Opt}^*_{\mathrm{Cov},k}$ be the subset of $\mathcal{C}_S(E)$ of size k maximizing $\lambda(\mathrm{Cov}_{A_S}(\cdot))$. Let $\textsc{Opt}_{\mathrm{Cov},k}$ be some subset of $\mathbb{X}_\ell^d$ of size k maximizing $\lambda(\mathrm{Cov}_P(\cdot,\Delta))$. Then by Theorem 4.3.16 we know that

$$\lambda(\mathrm{Cov}_{A_S}(\textsc{Opt}^*_{\mathrm{Cov},k})) \geq \frac{1}{8}\lambda(\mathrm{Cov}(\textsc{Opt}_{\mathrm{Cov},k},\Delta)).$$

Let $\overleftarrow{\textsc{Opt}^*_{\mathrm{Cov},k}}$ be the set of reversed sub-edges in $\textsc{Opt}^*_{\mathrm{Cov},k}$. Then by Lemma 4.6.10

$$\lambda\left(\widehat{\mathrm{Cov}}_{A_S}\left(\textsc{Opt}^*_{\mathrm{Cov},k} \cup \overleftarrow{\textsc{Opt}^*_{\mathrm{Cov},k}}\right)\right) \geq \lambda\left(\mathrm{Cov}_{A_S}(\textsc{Opt}^*_{\mathrm{Cov},k})\right).$$

Let further $\textsc{Opt}^*_{\widehat{\mathrm{Cov}},k}$ and $\textsc{Opt}^*_{\widehat{\mathrm{Cov}},2k}$ be the sets of size k and $2k$ in $\mathcal{C}_S(E)$ respectively, maximizing $\|\widehat{\mathrm{Cov}}_{A_S}(\cdot)\|$. Then by sub-additivity of $\lambda(\cdot)$

$$\begin{aligned}2\,\lambda\left(\widehat{\mathrm{Cov}}_{A_S}\left(\textsc{Opt}^*_{\widehat{\mathrm{Cov}},k}\right)\right) &\geq \lambda\left(\widehat{\mathrm{Cov}}_{A_S}\left(\textsc{Opt}^*_{\widehat{\mathrm{Cov}},2k}\right)\right)\\ &\geq \lambda\left(\widehat{\mathrm{Cov}}_{A_S}\left(\textsc{Opt}^*_{\mathrm{Cov},k} \cup \overleftarrow{\textsc{Opt}^*_{\mathrm{Cov},k}}\right)\right).\end{aligned}$$

Lastly as $\lambda(\widehat{\mathrm{Cov}}_{A_S}(\cdot))$ is submodular, by standard greedy submodular function maximization arguments [NWF78, KG14] it holds that

$$\lambda\left(\widehat{\mathrm{Cov}}_{A_S}(R)\right) \geq \frac{e-1}{e}\,\lambda\left(\widehat{\mathrm{Cov}}_{A_S}\left(\textsc{Opt}^*_{\widehat{\mathrm{Cov}},k}\right)\right).$$

And finally

$$\lambda(\mathrm{Cov}_P(R,(4+\varepsilon)\Delta)) \geq \lambda(\mathrm{Cov}_{A_S}(R)) \geq \lambda(\widehat{\mathrm{Cov}}_{A_S}(R)).$$

Thus overall, we observe that after k iterations it holds that

$$\lambda(\mathrm{Cov}_P(R,(4+\varepsilon)\Delta)) \geq \frac{e-1}{16e}\lambda(\mathrm{Cov}_P(\textsc{Opt}_{\mathrm{Cov},k},\Delta)).$$ □

The following lemma constructs values so that the Lebesgue measure of the coverage of elements in sweep sequences can be computed via a simple sweep algorithm in logarithmic time per element in a sweep-sequence.

Lemma 4.9.2. *Let $(\tau_1, \dots, \tau_n)$ be the vertex parameters of P. Let $I \subset [0,1]$ be a set of intervals. Let e be an edge of S and let $((s_1,t_1),\dots) = \mathfrak{s} \in \mathfrak{S}_e$ be a sweep-sequence. There are values $L^I_{(s,t)}$, $R^I_{(s,t)}$ and $C^I_{(s,t)}$ for every $(s,t) \in \mathfrak{s}$ such that for every $(s_k,t_k) \in \mathfrak{s}$ it holds that*

$$\lambda(\widehat{\mathrm{Cov}}_A(e[s_k,t_k]) \setminus I) = \left(\sum_{i\le k} L^I_{(s_i,t_i)}\right) s_k + \left(\sum_{i\le k} R^I_{(s_i,t_i)}\right) t_k + \left(\sum_{i\le k} C^I_{(s_i,t_i)}\right).$$

If $I = \emptyset$, then the values can be computed in $\mathcal{O}(n\varepsilon^{-2}\log n \log \varepsilon^{-1})$ time. Starting with $I = \emptyset$ we may iteratively add a (previously unknown) set J consisting of $\mathcal{O}(n)$ disjoint intervals where at most $\mathcal{O}(1)$ intersect any $[\tau_i, \tau_{i+1}]$ to I updating $L^I_{(s,t)}$, $R^I_{(s,t)}$ and $C^I_{(s,t)}$ to $L^{I\cup J}_{(s,t)}$, $R^{I\cup J}_{(s,t)}$ and $C^{I\cup J}_{(s,t)}$. If we add such a set K times, the total time for these updates takes $\mathcal{O}(Kn\varepsilon^{-2}\log n \log \varepsilon^{-1})$ time.

Proof. By Theorem 2.3.4 and Corollary 4.6.20 there are $m = \mathcal{O}(n\varepsilon^{-2})$ index pairs $p_1 = (i_1,j_1),\dots,p_m = (i_m,j_m)$ and contiguous subsets $I_i \subset \mathfrak{s}$ such that

$$\widehat{\mathrm{Cov}}_A(e[s_k,t_k]) = \bigsqcup_{a\le m,(s_k,t_k)\in I_a} [l^a_{i_a}(s_k), r^a_{j_a}(t_k)],$$

where $l^a_{i_a}(\cdot) = l_{i_a}(\cdot)$ if i_a is good for all $(s,t) \in I_a$, and $l^a_{i_a}(\cdot) = r_{i_a}(\cdot)$ if i_a is bad for all $(s,t) \in I_a$. Similarly $r^a_{j_a}(\cdot) = r_{j_a}(\cdot)$ if j_a is good for all $(s,t) \in I_a$, and $r^a_{j_a}(\cdot) = l_{j_a}(\cdot)$ if j_a is bad for all $(s,t) \in I_a$. Now let $M(i,j,I) = \lambda([\tau_i,\tau_j] \setminus I)$ if $i \le j$, and $M(i,j,I) = -\lambda([\tau_j,\tau_i] \setminus I)$ if $i > j$. Then

$$\lambda\left(\widehat{\mathrm{Cov}}_A(e[s_k,t_k]) \setminus I\right) = \sum_{a\le m,(s_k,t_k)\in I_a} \lambda([l^a_{i_a}(s_k), r^a_{j_a}(t_k)] \setminus I)$$
$$= \sum_{a\le m} \mathbb{1}[(s_k,t_k) \in I_a] \cdot \left(\lambda([l^a_{i_a}(s_k), \tau_{i_a+1}] \setminus I) + M(i_a+1, j_a, I) + \lambda([\tau_{j_a}, r^a_{j_a}(t_k)] \setminus I)\right).$$

The m values $M(i_a+1, j_a, I)$ for every $a \le m$ do not depend on k. Hence for every a and its contiguous set $I_a = \{(s_u,t_u),\dots,(s_v,t_v)\}$ we may define $M^{I,a}_{(s_u,t_u)} = M(i_a+1, j_a, I)$, and $M^{I,a}_{(s_{v+1},t_{v+1})} = -M(i_a+1, j_a, I)$ such that for every $(s_k,t_k) \in \mathfrak{s}$ it holds that

$$\sum_{a\le m} \mathbb{1}[(s_k,t_k) \in I_a] \cdot M(i_a+1, j_a, I) = \sum_{i\le k} \left(\sum_{a\le m} M^{I,a}_{(s_i,t_i)}\right),$$

and hence (by symmetry for $l_i(\cdot)$ and $r_j(\cdot)$) it suffices to show that for every a it holds that there are values $L^{I,a}_{(s,t)}$ and $C^{I,a}_{(s,t)}$ with

$$\mathbb{1}[(s_k,t_k) \in I_a] \cdot \lambda([l_{i_a}(s_k), \tau_{i_a+1}] \setminus I) = \left(\sum_{i\le k} L^{I,a}_{(s_i,t_i)}\right) s_k + \left(\sum_{i\le k} C^{I,a}_{(s_i,t_i)}\right).$$

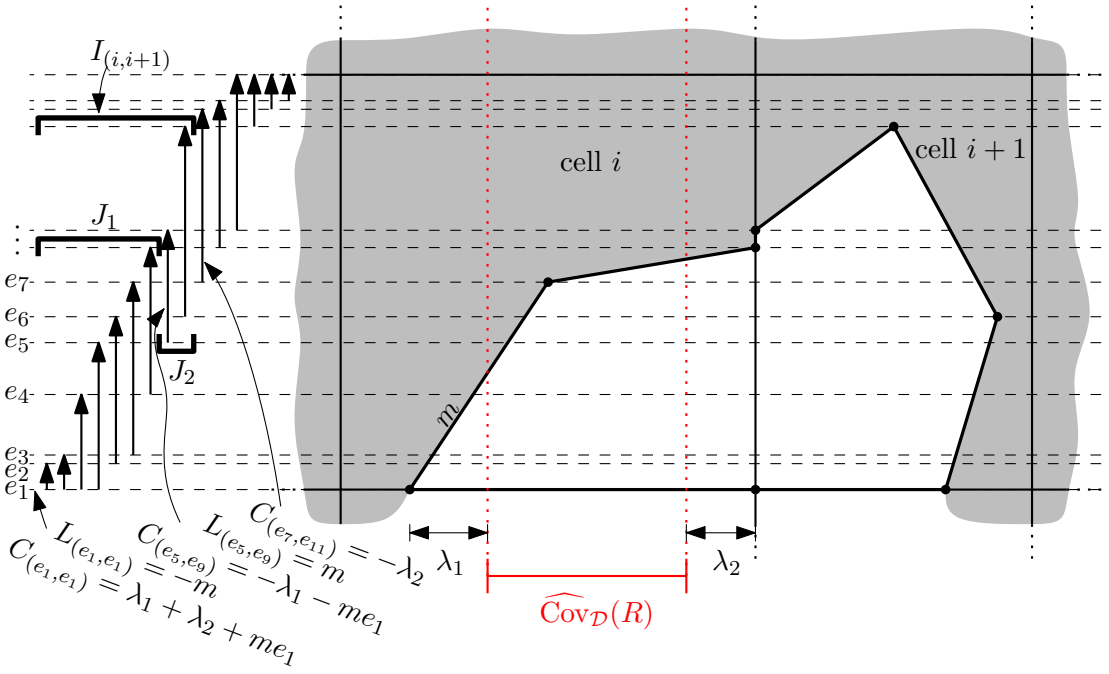


Figure 4.16: Construction of the sets $J_1, \ldots \subset I_{(i,i+1)} \subset \mathfrak{s}$ from the proof of Lemma 4.9.2 along which $l_i^{(i,i+1)}(\cdot)$ is the same linear function. Its parameters are stored in the first item of each J_i and the first item after each J_i. Only non-zero $L_{(a,b)}$ and $C_{(a,b)}$ are shown.

We split the set I_a into two contiguous sets $I_a^+ \subset I_a$ and $I_a^- \subset I_a$ based on whether for the element $(s,t) \in I_a$ the value s is above or below all y-coordinates of leftmost points in the cell i_a. We handle I_a^+ and I_a^- analogously hence we focus on $I_a^+ = \{(s_1,t_1), \ldots, (s_r,t_r)\}$. We may further assume that $a < b$ implies $s_a < s_b$. Now I_a^+ is partitioned into contiguous sets $J_1 = \{(s_1,t_1), \ldots, (s_c,t_c)\}, J_2 = \{(s_{c+1},t_{c+1}), \ldots\}, \ldots$ depending on whether $l_{i_a}(s) \in I$ or not. For every second set J_{2i} for all elements $(s,t) \in J_i$ it holds that $l_{i_a}(s) \in I$ and for every other J_{2i+1} and its elements $(s,t) \in J_{2i+1}$ it holds that $l_{i_a}(s) \notin I$. We refine this sequence of sets one last time splitting every J_{2i+1} at the at most $\mathcal{O}(\varepsilon^{-2})$ elements (s,t) such that s corresponds to the y-coordinate of a vertex of the left boundary of the polygon in cell i_a defining the free space. We are now left with a partition of I_a^+ into few contiguous sets such that inside any set (i) the local slope of the left boundary of the polygon defining the free space in cell i_a is constant and (ii) either all or none of the x-coordinates $l_{i_a}(s)$ are inside I. Let $J_1, \ldots, J_r$ be this sequence of contiguous subsets (refer to Figure 4.16). Let initially all $L^{I,a}_{(s,t)}$ and $C^{I,a}_{(s,t)}$ be zero.

Fix some J_i and let (s^*,t^*) be its first element. If J_i is such that $l_{i_a}(s) \in I$ for all $(s,t) \in J_i$ then let $M_i = 0$ and $C_i = \lambda([l_{i_a}(s^*), \tau_{i_a+1}] \setminus I)$. If J_i is such that $l_{i_a}(s) \notin I$ for all $(s,t) \in J_i$, then let $M_i = -m$ be the local slope of the left boundary of the polygon defining the free space in cell i_a and $C_i = \lambda([l_{i_a}(s^*), \tau_{i_a+1}] \setminus I) + ms^*$. Then add M_i to $L^{I,a}_{(s^*,t^*)}$ and C_i to $C^{I,a}_{(s^*,t^*)}$ and $-M_i$ to $L^{I,a}_{(s',t')}$ and $-C_i$ to $C^{I,a}_{(s',t')}$ for the first element (s',t') in $\mathfrak{s}$ after the last element in J_i. Then for any $(s_k,t_k) \in J_i$ with $l_{i_a}(s) \in I$ we have

$$\left(\sum_{i \le k} L^{I,a}_{(s_i,t_i)}\right) s_k + \left(\sum_{i \le k} C^{I,a}_{(s_i,t_i)}\right) = M_i s_k + C_i = C_i = \lambda([l_{i_a}(s^*), \tau_{i_a+1}] \setminus I)$$
$$= \lambda([l_{i_a}(s_k), \tau_{i_a+1}] \setminus I).$$

For $(s_k, t_k) \in J_i$ with $l_{i_a}(s) \notin I$ we have

$$\left(\sum_{i \leq k} L^{I,a}_{(s_i,t_i)}\right) s_k + \left(\sum_{i \leq k} C^{I,a}_{(s_i,t_i)}\right) = M_i s_k + C_i$$
$$= \lambda([l_{i_a}(s^*), \tau_{i_a+1}] \setminus I) - m(s_k - s^*) = \lambda([l_{i_a}(s_k), \tau_{i_a+1}] \setminus I),$$

implying the claim.

To compute these values for $I = \emptyset$ observe that all values $M(\cdot, \cdot, \emptyset)$, M_i and C_i can each be computed in $\mathcal{O}(\log n \log \varepsilon^{-1})$ time. As there are $\mathcal{O}(\varepsilon^{-2})$ polygon vertices per cell, the claim follows. If we are instead updating the values from I to $I \cup J$ observe that firstly the values $M(\cdot, \cdot, I \cup J)$ can be computed in $\mathcal{O}(\log n)$ time each. Next observe for an index pair (i_a, j_a) and set $I_a \subset \mathfrak{s}$ and any interval boundary of $J \setminus I$ in $[\tau_{i_a}, \tau_{i_a+1}]$ at most constantly many updates are necessary to maintain the partition $J_1, \ldots, J_r$ of I_a and thus updating M_i and C_i takes $\mathcal{O}(\log n \log \varepsilon^{-1})$ time. Throughout the K updates any edge $[\tau_{i_a}, \tau_{i_a+1}]$ requires at most $\mathcal{O}(K)$ updates induced by an interval boundary of $J \setminus I$, via a simple charging argument. Thus the index pair (i_a, j_a) is considered across all updates at most $\mathcal{O}(K)$ times and as any sweep-sequence contains at most $n\varepsilon^{-2}$ elements the total time all updates take is bounded by $\mathcal{O}(Kn\varepsilon^{-2} \log n \log \varepsilon^{-1})$. □

This lemma suffices to construct the algorithm.

Theorem 4.2.2. *Let $\varepsilon \in (0, 1]$. There is an $(\frac{e-1}{16e}, 4 + \varepsilon)$-approximation algorithm for SCM, where e is the base of the natural logarithm. Given a polygonal curve P of complexity n, $\Delta > 0$, $\ell \leq n$, and $k > 0$, its running time is in $\mathcal{O}((k + \ell)n^2\varepsilon^{-2} \log^2 n \log^2(\varepsilon^{-1}))$.*

Proof. First compute a 2Δ-maximal simplification S of P and a $(1 + \varepsilon, 4\Delta)$-free space $\mathcal{D}$ of S and P such that the free space in each cell consists of a convex polygon of complexity $\mathcal{O}(\varepsilon^{-2})$ via Theorem 2.3.4. The algorithm maintains a solution R which is initially empty and a set I_R of disjoint intervals with $I_R = \bigcup_{r \in R} \widehat{\mathrm{Cov}}_{\mathcal{D}}(r)$. Based on $\mathcal{D}$ we compute the set of Type (I)-, (II)- and (III)-subcurves induced by all $\mathcal{O}(n^2)$ extremal points and $\mathcal{O}(n^2\varepsilon^{-2})$ vertices of polygons defining the free space. Type (II)- and (III)-subcurves we store in sweep sequences. For the Type (I)-subcurves we compute their proxy coverage $\widehat{\mathrm{Cov}}_{\mathcal{D}}(\cdot)$. This overall takes $\mathcal{O}(n^2\ell \log^2 n + n^2\varepsilon^{-2} \log n \log \varepsilon^{-1})$ time. Now in K rounds we repeatedly compute the Type (I)-, (II)- or (III)-subcurve c^* maximizing $\lambda(\widehat{\mathrm{Cov}}_{\mathcal{D}}(c^*) \setminus I_R)$. For the Type (I)-subcurves we compute $\lambda(\widehat{\mathrm{Cov}}_{\mathcal{D}}(\cdot) \setminus I_R)$ explicitly in total time $\mathcal{O}(n^2 \log^2 n)$ time. For Type (II)- and (III)-subcurves we maintain the values from Lemma 4.9.2 with $I \leftarrow I_R$. As there are $\mathcal{O}(\log(n\varepsilon^{-1}))$ sweep sequences per edge, and at most K updates to I_R, this allows us to compute $\lambda(\widehat{\mathrm{Cov}}_{\mathcal{D}}(\cdot) \setminus I_R)$ in total time $\mathcal{O}(Kn^2\varepsilon^{-2} \log^2 n \log^2 \varepsilon^{-1})$. Finally Lemma 4.9.2 and Theorem 4.9.1 imply the correctness of the output. □

Chapter 5

Approximate Nearest Neighbor under the Fréchet Distance

In this chapter, we study the Fréchet distance, and whether we can construct a data structure answering approximate nearest neighbor queries for a given set of curves.

Problem 4 ($(1+\varepsilon)$-Approximate Nearest Neighbor Problem). Let $(\mathcal{M}, \mathrm{d}_{\mathcal{M}})$ be a metric space. Let $P \subset \mathcal{M}$ be a set of points in $\mathcal{M}$ and a parameter $\varepsilon > 0$ be given. For a given point $q \in \mathcal{M}$ the $(1+\varepsilon)$-Approximate Nearest Neighbor Problem ($(1+\varepsilon)$-ANN) is to find a point $\hat{x} \in P$ whose distance to q approximates the distance to the nearest neighbor in P. Specifically, $\hat{x} \in P$ is a valid solution iff for all $x \in P$ it holds that

$$\mathrm{d}_{\mathcal{M}}(q, \hat{x}) \leq (1+\varepsilon)\ \mathrm{d}_{\mathcal{M}}(q, x).$$

The main content of this chapter previously appeared as the paper $(1+\varepsilon)$-*ANN Data Structure for Curves via Subspaces of Bounded Doubling Dimension* [CDK24] by Jacobus Conradi, Anne Driemel, and Benedikt Kolbe, which was published in the Special Issue of Selected Papers from the 39th European Workshop on Computational Geometry (EuroCG 2023) of the journal *Computing in Geometry and Topology*. An initial version of the work has also been presented at the 39^{th} *European Workshop on Computational Geometry (EuroCG 2023)* based on an extended abstract without formal publication.

5.1 Introduction

Most of the work on Nearest Neighbor data structures for curves has focused on the discrete Fréchet distance. One notable example is a data structure for the $(1+\varepsilon, r)$-Approximate *Near* Neighbor Problem for curves in $\mathbb{R}^d$ of size $n \cdot \mathcal{O}(1/\varepsilon)^{md}$ presented in [FFK23], with a query time of $\mathcal{O}(md)$, where n is the number of input curves, and m is the complexity of the input curves. In the $(1+\varepsilon, r)$-Approximate Near Neighbor Problem the goal is the construction of a data structure on a set of input curves which for a given query curve outputs an input curve that is at distance at most $(1+\varepsilon)r$ if there is at least one curve in the input set within distance r. When the complexity k of the query curve is small compared to the complexity m of the input curves, the space can be improved to $n \cdot \mathcal{O}(1/\varepsilon)^{kd}$ with query time $\mathcal{O}(kd \log(nkd/\varepsilon))$.

Results for the $(1+\varepsilon, r)$-Approximate Near Neighbor problem readily extend to the $(1+\varepsilon)$-Approximate *Nearest* Neighbor Problem [HPIM12]. Their reduction incurs merely

an additional polylogarithmic factor $\mathcal{O}(\log^2(n))$ in the size, and $\mathcal{O}(\log n)$ in the query time.

In contrast to the multitude of approaches to the discrete Fréchet distance for arbitrary dimension, results w.r.t. the continuous Fréchet distance appear harder to come by. Consider the naïve approach of approximating the continuous Fréchet distance via the discrete Fréchet distance. For this, let a set of n curves in $\mathbb{X}_\Lambda^{d,k}$ as well as the approximation parameter ε be given. For the discrete Fréchet distance to approximate the continuous Fréchet distance up to an additive term of $r\varepsilon$ for some $r > 0$, we require successive vertices to lie at most $\Theta(r\varepsilon)$ far apart. Thus we subdivide every edge into edges of length at most $r\varepsilon$, resulting in a set of n curves each of complexity $\mathcal{O}(k\Lambda/(r\varepsilon))$. Building the $(1+\varepsilon, r)$-Approximate Near Neighbor data structure from [FFK23] results in a space requirement of $n \cdot \mathcal{O}(1/\varepsilon)^{\mathcal{O}(dk\Lambda/(r\varepsilon))}$. As the radii used in the reduction can be as small as r^*/n, where r^* is roughly the distance of some two input curves, this extends to a data structure of size $\mathcal{O}(n\log^2(n)) \cdot \mathcal{O}(1/\varepsilon)^{\mathcal{O}(ndk\Lambda/(r^*\varepsilon))}$ for the $(1+\varepsilon)$-ANN problem, where the exponential dependence in n and Λ/r^* are particularly undesirable.

In one dimension, [BDNP22] showed that there is a $(1+\varepsilon, r)$-Approximate Near Neighbor data structure for the continuous Fréchet distance, which uses $n \cdot \mathcal{O}(\frac{m}{k\varepsilon})^k$ space, needs $\mathcal{O}(nm) \cdot \mathcal{O}(\frac{m}{k\varepsilon})^k$ expected preprocessing time, and achieves a query time of $\mathcal{O}(k2^k)$. They also show tightness of their data structure bounds in several scenarios. More precisely, they show conditional lower bounds based on the Orthogonal Vectors Hypothesis that give reason to believe that one cannot achieve both polynomial (in n) preprocessing time and query time in $\mathcal{O}(n^{1-\varepsilon'})$, when k is $1 \ll k \ll \log n$ and $m > k \cdot n^{c/k}$, for some c depending on ε and ε', even if $d = 1$. Their arguments also apply to the $(1+\varepsilon)$-ANN problem under the continuous Fréchet distance for any $\varepsilon < 1$.

In two dimensions, [AD18] presented a data structure based on semi-algebraic range searching that solves the (exact) Near Neighbor Problem under the Fréchet distance. The space required to construct this data structure is in $\mathcal{O}(n(\log\log n)^{\mathcal{O}(m^2)})$ and its query time is in $\mathcal{O}(\sqrt{n}\log^{\mathcal{O}(m^2)} n)$.

In higher dimensions, [Mir23] presented a data structure result for the $(1+\varepsilon, r)$-Approximate Near Neighbor Problem under the continuous Fréchet distance, using space in $n \cdot \mathcal{O}((\max(1, D)\sqrt{d}/\varepsilon^2)^{kd})$ and query time in $\mathcal{O}(kd)$, where D denotes the diameter of the underlying vertex set of the input curves. However, as presented this data structure works only if $\varepsilon < r$. The data structure covers the entirety of the input curves with a grid of edge-length roughly εr, to precompute an answer for every sequence of k gridpoints. A query curve is then snapped to the closest grid points and the precomputed answer is given as the output. For smaller values of r one would have to scale the input, increasing D accordingly. As a result, combining this data structure with the standard reduction from [HPIM12] does not lead to an efficient data structure for the ANN-problem.

Independent to our work, a $(1+\varepsilon)$-ANN data structure for polygonal curves in arbitrary dimension under the continuous Fréchet distance was presented in [CH23a]. The data structure uses space in $\widetilde{\mathcal{O}}_d\left(k(mnd^d/\varepsilon^d)^{\mathcal{O}(k+1/\varepsilon^2)}\right)$ and achieves query time in $\widetilde{\mathcal{O}}_d\left(k(mn)^{0.5+\varepsilon}/\varepsilon^d + k(d/\varepsilon)^{\mathcal{O}(dk)}\right)$, where $\widetilde{\mathcal{O}}_d(\cdot)$ hides polylogarithmic factors with exponents in $\mathcal{O}(d)$.

5.1.1 Results

In this chapter, we provide a $(1+\varepsilon)$-ANN data structure for a set of curves in arbitrary dimensions under the continuous Fréchet distance. The preprocessing time of this data

structure depends linearly on n, with its query time depending only logarithmically on n. This comes at the expense of a factor of roughly $(\varepsilon^{-1}\Lambda/r^*)^k$ in both the preprocessing time, space, and query time. One of the main ingredients to the construction of our data structure is the construction of a suitable subspace of the space of all polygonal curves. This subspace has small distance to the original space (i.e., small Gromov-Hausdorff distance as metric spaces) and additionally, unlike the space of all polygonal curves, bounded doubling dimension.

Throughout this chapter we will not distinguish between small or large k, that is, we assume $k = m$. In Section 5.5 we present the following result.

Theorem 5.1.1. *Given a set S of n polygonal curves in $\mathbb{X}_\Lambda^{d,k}$ and parameters $\varepsilon \in (0,1]$ and $\varepsilon' > 0$, one can construct a data structure that for given $q \in \mathbb{X}^{d,k}$ outputs an element $s^* \in S$ such that for all $s \in S$ it holds that $\mathrm{d}_{\mathcal{F}}(s^*, q) \leq (1+\varepsilon)\mathrm{d}_{\mathcal{F}}(s, q)+\varepsilon'$. The query time is $\mathcal{O}\left(2^{\mathcal{O}(d)}k(1+\Lambda/\varepsilon')\right)^k \log n + \mathcal{O}\left(2^{\mathcal{O}(d)}k(1+\Lambda/\varepsilon')\right)^{-k\log(\varepsilon)}$, the expected preprocessing time is $\mathcal{O}\left(2^{\mathcal{O}(d)}k(1+\Lambda/\varepsilon')\right)^k n \log n$ and the space used is $\mathcal{O}\left(2^{\mathcal{O}(d)}k(1+\Lambda/\varepsilon)\right)^k n$.*

To turn this into a data structure with a purely multiplicative error, we choose ε' as the smallest Fréchet distance of any two distinct input curves over ε. Thus, the running time and space complexity of our data structure polynomially depends on a numerical value, which we call the bundledness of the set of input curves.

Definition 5.1.2 (bundledness)**.** Given a set of curves $S \in \mathbb{X}^{d,k}$, the bundledness $\mathcal{G}(S)$ of S is defined as

$$\mathcal{G}(S) = \frac{\min_{s \neq s' \in S} \mathrm{d}_{\mathcal{F}}(s, s')}{\max_{e \in E(S)} \|e\|}$$

where $E(S)$ denotes the set of edges of curves in S.

Note that this measure is scale- and translation-independent, as translating all elements in S by the same offset changes neither their pairwise Fréchet distances nor the length of the edges of the curves. Similarly, scaling the elements of S scales both the pairwise Fréchet distances and the lengths of the edges of the curves, thus not changing the bundledness.

The bundledness is reminiscent of the global stretch—a measure of complexity on geometric graphs—which was introduced in [Eri05] and further analyzed in [BCL+10]. The bundledness is closely related to the spread of the set of vertices and edges of the input curves in that it is lower bounded by the reciprocal of the spread (see Lemma 5.5.3).

Definition 5.1.3 (spread)**.** For a point set P in some metric space $(\mathcal{M}, d_{\mathcal{M}})$ we define the spread $\Phi(P)$ as the ratio between the maximal and minimal pairwise distance of points in P. Similarly, define the spread $\Phi(S)$ of a collection of sets as the ratio between the maximal and minimal non-zero pairwise distances of sets in S, where the distance between two sets $A, B \subset \mathcal{M}$ is defined as $\mathrm{d}_{\mathcal{M}}(A, B) = \min_{a \in A} \min_{b \in B} \mathrm{d}_{\mathcal{M}}(a, b)$.

With these definitions, our principal results can be summarized as follows.

Theorem 5.1.4. *Given a set S of n polygonal curves in $\mathbb{X}^{d,k}$ and $\varepsilon \in (0,1]$, one can construct a data structure answering $(1+\varepsilon)$-approximate nearest neighbor queries. The query time is $F(d,k,S,\varepsilon)\log n + F(d,k,S,\varepsilon)^{-\log(\varepsilon)}$, the expected preprocessing time is $F(d,k,S,\varepsilon)n\log n$ and the space required for the data structure is at most $F(d,k,S,\varepsilon)n$, where $F(d,k,S,\varepsilon) = \mathcal{O}\left(2^{\mathcal{O}(d)}k(1+\mathcal{G}(S)^{-1}\varepsilon^{-1})\right)^k$.*

Replacing the bundledness with the more pessimistic spread of the vertices and edges of the input curves yields the following result.

Corollary 5.1.5. *Given a set S of n polygonal curves in $\mathbb{X}^{d,k}$ and $\varepsilon \in (0,1]$ one can construct a data structure answering $(1+\varepsilon)$-approximate nearest neighbor queries. The query time is $F(d,k,S,\varepsilon)\log n + F(d,k,S,\varepsilon)^{-\log(\varepsilon)}$, the expected preprocessing time is $F(d,k,S,\varepsilon)n\log n$ and the space required for the data structure is at most $F(d,k,S,\varepsilon)n$, where $F(d,k,S,\varepsilon) = \mathcal{O}\left(2^{\mathcal{O}(d)}k\Phi(S)\varepsilon^{-1}\right)^k$, where $\Phi(S)$ denotes the spread of the set of vertices and edges of the curves in S.*

In the special case that all curves in S are c-packed for some constant $c > 0$, the parameter $F(d,k,S,\varepsilon)$ is instead in $\mathcal{O}\left(2^{\mathcal{O}(d)}(1+\mathcal{G}(S)^{-1}\varepsilon^{-1})\right)^k$, or, more pessimistically, in $\mathcal{O}\left(2^{\mathcal{O}(d)}\Phi(S)\varepsilon^{-1}\right)^k$.

5.1.2 Technical Overview

In Section 5.2, we introduce a subspace $\mathbb{X}^*$ of $(\mathbb{X}^{d,k}, \mathrm{d}_\mathcal{F})$ and show how the subspace relates to $(\mathbb{X}^{d,k}, \mathrm{d}_\mathcal{F})$. A significant part of our work is concerned with analyzing properties of this subspace $\mathbb{X}^*$. In Section 5.3, we present the analysis of the upper bound on the doubling constant and dimension of $\mathbb{X}^*$, which constitutes our main technical result, Theorem 5.3.5. We show that for a curve Q to have a small Fréchet distance (at most Δ) to another curve C, there is only a small subset of $\mathbb{R}^d$ in which any vertex of Q can lie in. In fact, it turns out that this subset can be covered by roughly $2^d k\mu$ ($\mathbb{R}^d$-)balls of radius $\Delta/2$, depending on the doubling dimension $\Theta(d)$ of $\mathbb{R}^d$. As Q consists of k vertices, Q is then described by one of at most $(2^d k\mu)^k$ sequences of ($\mathbb{R}^d$-)balls, implying that the doubling dimension is at most $\log_2((2^d k\mu)^k) = k(d+\log_2(k\mu))$. We extend this analysis to the special case that the curves are c-packed, resulting in an improvement in the upper bound.

In Section 5.4, we extend the lower bound construction from [DKS16] to argue that our bounds on the doubling dimension of $\mathbb{X}^*$ are almost tight.

Finally, in Section 5.5, we use the bound on the doubling dimension of $\mathbb{X}^*$ to construct an $(1+\varepsilon)$-ANN data structure based on [HPM06].

5.2 Curve Spaces

Recall that [DKS16] showed that for $k \geq 3$, the doubling dimension of $(\mathbb{X}^{d,k}, \mathrm{d}_\mathcal{F})$ is unbounded. A straightforward modification to their construction yields the following theorem, a stronger version of which we will present in Section 5.4.

Theorem 5.2.1. *The doubling constant of $(\mathbb{X}^{d,k}_\Lambda, \mathrm{d}_\mathcal{F})$ is unbounded for any $k \geq 3$ and $\Lambda > 0$.*

Theorem 5.2.1 motivates the search for a subspace $(\mathbb{X}^*, \mathrm{d}_\mathcal{F})$ of $(\mathbb{X}^{d,k}_\Lambda, \mathrm{d}_\mathcal{F})$ of bounded doubling dimension. To answer $(1+\varepsilon)$-ANN queries in $\mathbb{X}^{d,k}_\Lambda$, the ansatz is to map the input curves to $(\mathbb{X}^*, \mathrm{d}_\mathcal{F})$ and answer queries in this subspace. The mapping decreases the quality guarantee of $(1+\varepsilon)$-ANN queries by an additional additive factor, roughly depending on the distortion of the map between $(\mathbb{X}^{d,k}_\Lambda, \mathrm{d}_\mathcal{F})$ and $(\mathbb{X}^*, \mathrm{d}_\mathcal{F})$.

Definition 5.2.2 ((μ,ε)-curves)**.** For any $\varepsilon > 0$ and $\mu \in \mathbb{N}$, define the space of (μ,ε)-curves in $\mathbb{X}^{d,k}$ as the subspace of $(\mathbb{X}^{d,k}, \mathrm{d}_\mathcal{F})$ induced by the set of polygonal curves in

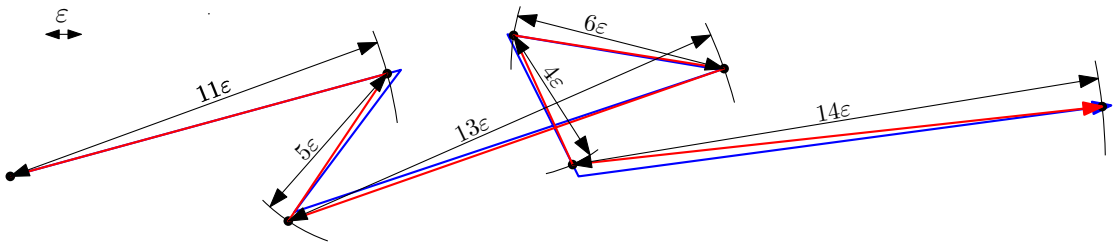


Figure 5.1: Example of a curve $P \in \mathbb{X}_{\Lambda}^{d,k}$ in blue, and an ε-curve close to P resulting from Lemma 5.2.3 in red.

$\mathbb{X}^{d,k}$ whose edge lengths are all exact multiples of ε. We further require the edge lengths to be bounded by $\mu\varepsilon$. The space of (μ,ε)-curves in $\mathbb{X}^{d,k}$ naturally forms a subspace of $(\mathbb{X}_{\mu\varepsilon}^{d,k}, \mathrm{d}_{\mathcal{F}})$.

We may abuse notation slightly, and not specify the ambient space $\mathbb{X}^{d,k}$ of the space of (μ,ε)-curves, if the ambient space is clear. We may likewise write $\mathcal{M}$ when talking about the metric space $(\mathcal{M}, \mathrm{d}_{\mathcal{M}})$, if the metric is clear.

Lemma 5.2.3. *Let $P \in \mathbb{X}_{\Lambda}^{d,k}$ be a polygonal curve and $\varepsilon > 0$. We can construct a $(\lceil \Lambda/\varepsilon \rceil + 1, \varepsilon)$-curve P' in $\mathbb{X}^{d,k}$ such that $\mathrm{d}_{\mathcal{F}}(P,P') \leq \varepsilon/2$ in $\mathcal{O}(k \log(\Lambda/\varepsilon))$ time.*

Proof. Refer to Figure 5.1. Let $p_1, \ldots, p_k$ be the vertices of P and $P' = \emptyset$. We begin by adding $p'_1 = p_1$ to P'. Now assume p'_{i-1} is the last vertex of P'. Compute the value $\mu_i \in \mathbb{N}$, such that the magnitude $|\mu_i\varepsilon - \|p_i - p'_{i-1}\||$ is minimal. Then add $p'_i = p'_{i-1} + (\mu_i\varepsilon)\frac{p_i - p'_{i-1}}{\|p_i - p'_{i-1}\|}$ to P'. Note that by construction $\|p_i - p'_i\| \leq \varepsilon/2$. Hence, $\mathrm{d}_{\mathcal{F}}(P,P') \leq \varepsilon/2$. The length of the edges of P' are bounded by $(\lceil \Lambda/\varepsilon \rceil + 1)\varepsilon$. Indeed, $\|p_i - p'_i\| \leq \varepsilon/2$ and $\|p_i - p_{i-1}\| \leq \Lambda$ imply that $\|p'_i - p'_{i-1}\| \leq \Lambda + \varepsilon$.

For the running time, for every $i \leq n$, the value μ_i such that the magnitude $|\mu_i\varepsilon - \|p_i - p'_{i-1}\||$ is minimal can be identified in $\mathcal{O}(\log(\Lambda/\varepsilon))$ time. This is done by first identifying the smallest power of 2 larger than $\|p_i - p'_{i-1}\|/\varepsilon$ and then binary searching over the integer multiples of ε up to this power of 2. As we do this once for every edge, the claimed running time follows. □

Note that via Lemma 5.2.3 we obtain a map from $\mathbb{X}^{d,k}$ to its subspace of (∞,ε)-curves.

The additive error incurred when answering ANN queries in the subspace depends on the distortion of the map from $\mathbb{X}_{\Lambda}^{d,k}$ to the space of (M,ε)-curves in $\mathbb{X}^{d,k}$. The ***Gromov-Hausdorff distance*** is a related measure of the distance between metric spaces. Intuitively, it measures the smallest possible distortion of maps between the two spaces. The space of (∞,ε)-curves in $\mathbb{X}^{d,k}$ has (depending on ε) small Gromov-Hausdorff distance to the ambient space $\mathbb{X}^{d,k}$ of curves:

Definition 5.2.4 (Gromov-Hausdorff distance)**.** The Gromov-Hausdorff distance is a distance measure on metric spaces. Let $\mathcal{M}$ and $\mathcal{N}$ be two metric spaces. Then the Gromov-Hausdorff distance is defined as

$$\mathrm{d}_{GH}(\mathcal{M},\mathcal{N}) = \inf_{\mathcal{Z}} \{\mathrm{d}_H^{\mathcal{Z}}(f(\mathcal{M}), g(\mathcal{N})) \mid f : \mathcal{M} \to \mathcal{Z},\ g : \mathcal{N} \to \mathcal{Z} \text{ isometric embeddings}\},$$

where $\mathcal{Z}$ ranges over metric spaces and $\mathrm{d}_H^{\mathcal{Z}}(X,Y)$ denotes the Hausdorff distance of two sets in $\mathcal{Z}$ defined as $\max\{\sup_{x\in X} \inf_{y\in Y} d_{\mathcal{Z}}(x,y), \sup_{y\in Y} \inf_{x\in X} d_{\mathcal{Z}}(x,y)\}$.

Corollary 5.2.5. *Let d, k, and $\varepsilon > 0$ be given. For any $\mu \in \mathbb{N}$, denote by C_μ the space of (μ,ε)-curves in $\mathbb{X}^{d,k}$. Then $\mathrm{d}_{GH}(\mathbb{X}^{d,k}, C_\infty) \leq \varepsilon/2$.*

Proof. For every $P \in \mathbb{X}^{d,k}$ and $\varepsilon > 0$ there is a finite integer $M < \infty$, such that $P \in \mathbb{X}^{d,k}_{M\varepsilon}$. Then by Lemma 5.2.3 there is a $P' \in C_{M+1} \subset C_\infty$ with $\mathrm{d}_{\mathcal{F}}(P,P') \leq \varepsilon/2$. As C_∞ is a subspace of $\mathbb{X}^{d,k}$, the claim holds. □

5.3 Upper Bound for Doubling Dimension of (μ,ε)-Curves

In this section, we study the doubling dimension of the space of (μ,ε)-curves in $\mathbb{X}^{d,k}$. Unfortunately, our bound is non-constructive. As such, it does not provide a doubling oracle that, for a given ball of radius r in the metric space, outputs a set of balls of radius $r/2$ which cover the ball of radius r.

5.3.1 Properties of the Euclidean Space

Before diving into the analysis of the doubling dimension of the space of (μ,ε)-curves, we begin by analyzing properties of the ambient space $\mathbb{R}^d$. For this we often inspect so called Δ-neighborhoods of subsets of $\mathbb{R}^d$. For any subset A, the **Δ-neighborhood** of A is defined by $N_\Delta(A) = \{x \in \mathbb{R}^d \mid \exists a \in A : d(x,a) \leq \Delta\}$. Note that the Δ-neighborhood of a single point x coincides with a ball of radius Δ centered at x.

Recall that the doubling dimension of $\mathbb{R}^d$ under the Euclidean norm is in $\Theta(d)$. We prove a more general statement which relates the volume of some arbitrary set to the amount of balls needed to cover it. For this, we denote by λ^d the Lebesgue measure in $\mathbb{R}^d$ and by $\lambda^d(A)$ the **volume** of a (measurable) set $A \subset \mathbb{R}^d$.

Lemma 5.3.1. *Let $A \subset \mathbb{R}^d$ be a bounded set, and let $r > 0$ be fixed. Then there is a set of points $C \subset A$ of cardinality $\lceil \lambda^d(N_{r/2}(A))/V^d_{r/2} \rceil$ such that $A \subset \bigcup_{c\in C} \mathrm{D}_r(c)$, where $V^d_{r/2}$ denotes the volume of a d-dimensional ball of radius $r/2$.*

Proof. We construct the set C greedily. For this we start with $C = \emptyset$, and then iteratively add some point from $A \setminus \bigcup_{c\in C} \mathrm{D}_r(c)$, until $A \subset \bigcup_{c\in C} \mathrm{D}_r(c)$. As any two points in C have a distance of at least r, balls centered at points of C with radius $r/2$ are disjoint and clearly contained in $N_{r/2}(A)$. Thus $|C|$ is bound by $\lceil \lambda^d(N_{r/2}(A))/V^d_{r/2} \rceil$. □

Lemma 5.3.2. *For any $r > 0$ and $c > 1$, any ball $\mathrm{D}_r(p) \subset \mathbb{R}^d$ can be covered by $\mathcal{O}((2c+1)^d)$ balls of radius r/c.*

Proof. This follows from the classical result that $V^d_r = \frac{\pi^{d/2}}{\Gamma(d/2+1)} r^d$, where Γ denotes the Gamma-function [Par13]. As $N_{r/2c}(\mathrm{D}_r(p)) = \mathrm{D}_{r(2c+1)/2c}(p)$, Lemma 5.3.1 implies that any $\mathrm{D}_r(p)$ can be covered with $\lceil (r(2c+1)/2c)^d/(r/2c)^d \rceil = \mathcal{O}((2c+1)^d)$ balls of radius r/c. □

We will use this lemma to obtain a set of points (i.e., the centers of the balls used in such a covering) as candidates for vertices of curves that will be centers of balls in $\mathbb{X}^{d,k}$.

When analyzing the doubling dimension, we will consider what an edge might look like that has Fréchet distance at most $\Delta > 0$ to a subcurve of an input curve. The basic tool we use for this analysis is the observation that the edge under consideration then is a Δ-stabber of the vertices (and indeed any ordered set of points along the subcurve) of the subcurve.

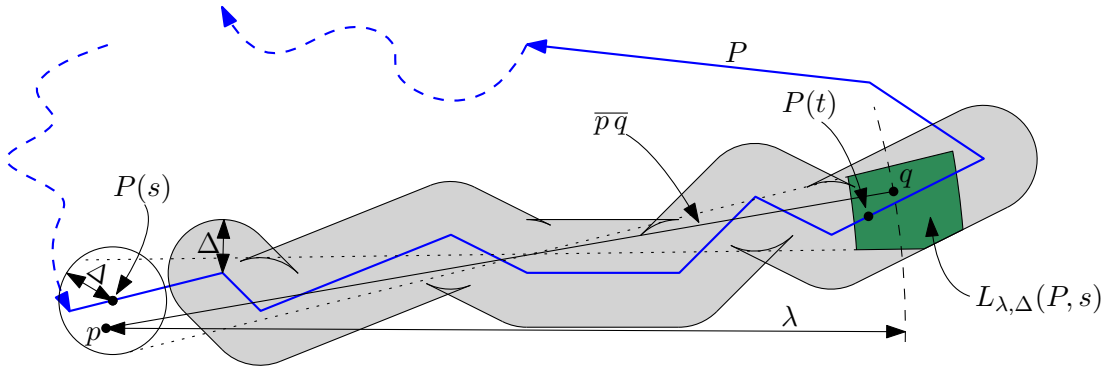


Figure 5.2: Illustration of the set $L_{\lambda,\Delta}(P,s)$ in dark green, together with the points p and $P(t)$ realizing a point q in $L_{\lambda,\Delta}(P,s)$, that is, $\|p-q\| = \lambda$ and $\mathrm{d}_{\mathcal{F}}(P[s,t], \overline{p\,q}) \leq \Delta$.

Definition 5.3.3 (Δ-stabber)**.** Let an ordered set of points $(p_1, \ldots, p_n)$ in $\mathbb{R}^d$ be given. An edge $l = \overline{a\,b}$ is called a Δ-stabber of $(p_1, \ldots, p_n)$ if there are values $0 \leq t_1 \leq \ldots \leq t_n \leq 1$ such that $\|l(t_i) - p_i\| \leq \Delta$ for all $1 \leq i \leq n$.

The notion of Δ-stabbers has been introduced in [GHMS94], and is closely related to the Fréchet distance. Any edge that has Fréchet distance at most Δ to a polygonal curve defined by vertices $p_1, \ldots, p_n$ is a Δ-stabber of the ordered point set $(p_1, \ldots, p_n)$. Similarly, a Δ-stabber of the ordered point set $(p_1, \ldots, p_n)$ contains an edge that has Fréchet distance at most Δ to the polygonal curve defined by the vertices $p_1, \ldots, p_n$.

Observation 5.3.4. *Let a polygonal curve P and $\Delta > 0$ be given. Let $e = \overline{p\,q}$ be an edge such that for given $0 \leq s \leq t \leq 1$ the Fréchet distance $\mathrm{d}_{\mathcal{F}}(P[s,t], e)$ is at most Δ. Then for any $s \leq m \leq t$ the edge e is a Δ-stabber of $(P(s), P(m), P(t))$.*

5.3.2 Packing the Metric Ball

We now turn to proving the following theorem.

Theorem 5.3.5. *Let $k, \mu, d \in \mathbb{N}$ and $\varepsilon > 0$. The doubling constant of the space of (μ,ε)-curves in $\mathbb{X}^{d,k}$ is bounded by $\mathcal{O}(43^d k\mu)^k$ and thus the doubling dimension of the space of (μ,ε)-curves is bounded by $\mathcal{O}(k(d + \log(k\mu)))$.*

Let P be a (μ,ε)-curve in $\mathbb{X}^{d,k}$. Our objective is to cover the Δ-neighborhood of P with respect to $\mathrm{d}_{\mathcal{F}}$ with balls of radius $\Delta/2$. We encounter the question of where, given $p \in \mathbb{R}^d$, we may place a second point q such that there is a subcurve of P that is close to $\overline{p\,q}$. Indeed, for any curve Q with $\mathrm{d}_{\mathcal{F}}(P,Q) \leq \Delta$, the endpoint q of any edge $\overline{p\,q}$ of Q is a potential such point for the start point p. Thus if we answer the above stated question, we can iteratively build up any curve Q with $\mathrm{d}_{\mathcal{F}}(P,Q) \leq \Delta$.

Definition 5.3.6. Let P be a polygonal curve and $\lambda \geq 0$ and $\Delta \geq 0$. For $s \in [0,1]$ define the locus of edge endpoints of edges close to subcurves of P starting at the parameter s (refer to Figure 5.2) as the set

$$L_{\lambda,\Delta}(P,s) = \left\{ q \in \mathbb{R}^d \mid \exists p \in \mathbb{R}^d \text{ and } \exists t \in [s,1] \text{ with } \|p-q\| = \lambda,\ \mathrm{d}_{\mathcal{F}}(P[s,t], \overline{p\,q}) \leq \Delta \right\}.$$

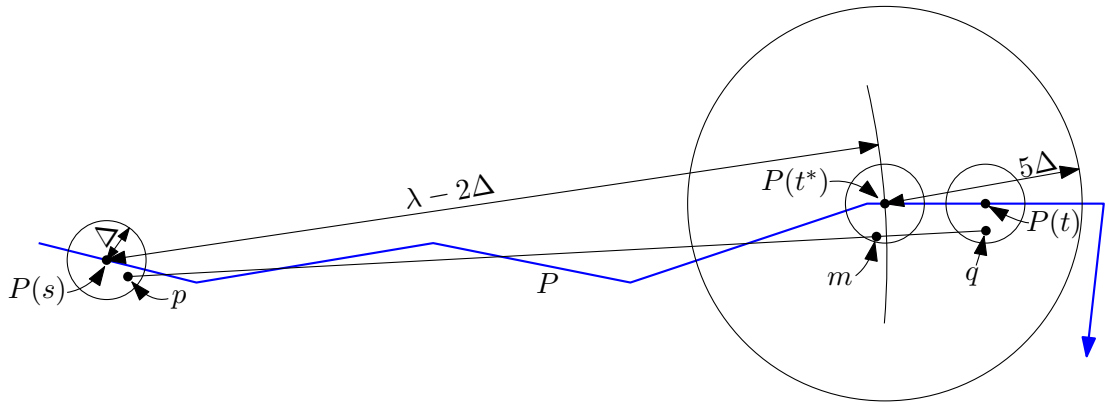


Figure 5.3: Illustration to the proof of Lemma 5.3.7.

The following lemma motivates discretizing the lengths of edges, as for a fixed length λ the set $L_{\lambda,\Delta}(P,s)$ is contained in a single ball of constant size.

Lemma 5.3.7. *Let P be a polygonal curve. Let $\lambda \geq 0$ and $\Delta \geq 0$ be given. Then for every $s \in [0,1]$ there is a point $p^* \in \mathbb{R}^d$, such that*

$$L_{\lambda,\Delta}(P,s) \subset \mathrm{D}_{5\Delta}(p^*).$$

Proof. Assume $L_{\lambda,\Delta}(P,s)$ is non-empty, as otherwise we are done. Similarly assume $\lambda \geq 4\Delta$ as otherwise $L_{\lambda,\Delta}(P,s)$ is trivially contained in $\mathrm{D}_{\lambda+\Delta}(P(s)) \subset \mathrm{D}_{5\Delta}(P(s))$. Now let $t^* \geq s$ be the smallest value such that $\|P(s) - P(t^*)\| \geq \lambda - 2\Delta$. Then we claim that the sought-after point is $p^* = P(t^*)$.
For the remainder of this proof refer to Figure 5.3. Let $q \in L_{\lambda,\Delta}(P,s)$ be given. Then by definition there is a point p and a value t, such that $\overline{p\,q}$ has length λ and $\mathrm{d}_{\mathcal{F}}(P[s,t], \overline{p\,q}) \leq \Delta$. Thus $\|P(s) - p\| \leq \Delta$ and similarly $\|P(t) - q\| \leq \Delta$. Then by the triangle inequality $\|P(t) - P(s)\| \geq \lambda - 2\Delta$. Thus $t \geq t^*$. Hence, by Observation 5.3.4, $\overline{p\,q}$ is a Δ-stabber of $(P(s), P(t^*), P(t))$. This implies that there is a point m along $\overline{p\,q}$ with $\|m - P(t^*)\| \leq \Delta$. As m lies on $\overline{p\,q}$, we have that $\|p-q\| = \|p-m\| + \|m-q\|$. As $\|P(s) - P(t^*)\| \geq \lambda - 2\Delta$, we get that $\|p-m\| \geq \lambda - 4\Delta$, and thus $\|m-q\| \leq 4\Delta$. And thus finally $\|P(t^*) - q\| \leq 5\Delta$, implying the claim. □

Definition 5.3.8. Let P be a polygonal curve and $\lambda \geq 0$ and $\Delta \geq 0$. For $p \in \mathbb{R}^d$ define the locus of edge endpoints of edges starting at p which are close to subcurves of P as the set

$$\mathcal{L}_{\lambda,\Delta}(P,p) = \left\{ q \in \mathbb{R}^d \;\middle|\; \|p-q\| = \lambda \text{ and } \exists s,t \text{ with } 0 \leq s \leq t \leq 1,\ \mathrm{d}_{\mathcal{F}}(P[s,t], \overline{p\,q}) \leq \Delta \right\}.$$

Similarly to Lemma 5.3.7, we can identify balls that cover the entirety of $\mathcal{L}_{\lambda,\Delta}(P,p)$ for given P, λ, Δ and p. However, instead of a constant number of balls we need up to k balls of constant radius to cover this set.

Lemma 5.3.9. *Let $P \in \mathbb{X}^{d,k}$ be a polygonal curve. Let $\lambda \geq 0$ and $\Delta \geq 0$ be given. Then for every $p \in \mathbb{R}^d$ there are k points $p_1^*, \ldots, p_k^* \in \mathbb{R}^d$ such that*

$$\mathcal{L}_{\lambda,\Delta}(P,p) \subset \bigcup_{i=1}^{k} \mathrm{D}_{5\Delta}(p_i^*).$$

Proof. The set $I = \{s \in [0,1] \mid P(s) \in \mathrm{D}_\Delta(p)\}$ can be described as a disjoint union of at most k closed intervals, as the complexity of P is bounded by k. Assume that it is described by exactly k such intervals, that is, $I = \bigcup_{i=1}^{k}[l_i, r_i]$.

It suffices to show that $\mathcal{L}_{\lambda,\Delta}(P,p) \subset \bigcup_{i=1}^{k} L_{\lambda,\Delta}(P,l_i)$, as Lemma 5.3.7 then implies the claim. Assume that an arbitrary $q \in \mathcal{L}_{\lambda,\Delta}(P,p)$ is given. Then by definition there are values $0 \le s \le t \le 1$ such that $\mathrm{d}_{\mathcal{F}}(P[s,t], \overline{p\,q}) \le \Delta$. This implies that $\|p - P(s)\| \le \Delta$, and hence $s \in I$ and in turn $s \in [l_i, r_i]$ for some $1 \le i \le k$. Then the subcurve $P[l_i, s]$ is contained in $\mathrm{D}_\Delta(p)$, and thus $\mathrm{d}_{\mathcal{F}}(P[l_i,t], \overline{p\,q}) \le \Delta$ implying that $q \in L_{\lambda,\Delta}(P,l_i)$ and thus the claim. □

Corollary 5.3.10. *For every polygonal curve P in $\mathbb{R}^d$, $\Delta > 0$, $\lambda > 0$, $c > 1$ and point $p \in \mathbb{R}^d$, the set $N_{\Delta/c}\left(\mathcal{L}_{\lambda,(1+c^{-1})\Delta}(P,p)\right)$ can be covered by a set of balls of radius Δ/c centered at $\mathcal{O}(k(10c+3)^d)$ points.*

Proof. For any point $p \in \mathbb{R}^d$, $\Delta > 0$ and $c > 1$, the sets $N_{\Delta/c}(\mathrm{D}_{5\Delta}(p))$ and $\mathrm{D}_{(5+c^{-1})\Delta}(p)$ coincide, so Lemma 5.3.2 and Lemma 5.3.9 imply the claim. □

Lemma 5.3.11. *Let P be a polygonal curve. Let $\lambda \ge 0$, $\Delta \ge 0$ and $c \ge 1$ be given. Then for every $p \in \mathbb{R}^d$ and $p' \in \mathbb{R}^d$ with $\|p - p'\| \le \Delta/c$ we have that*

$$\mathcal{L}_{\lambda,\Delta}(P,p) \subset N_{\Delta/c}\left(\mathcal{L}_{\lambda,(1+c^{-1})\Delta}(P,p')\right).$$

Proof. Let $q \in \mathcal{L}_{\lambda,\Delta}(P,p)$. This implies that there are values $0 \le s \le t \le 1$ such that $\mathrm{d}_{\mathcal{F}}(P[s,t], \overline{p\,q}) \le \Delta$ and $\|p - q\| = \lambda$. Let $q' = q + (p' - p)$. Then, as $\|p - p'\| \le \Delta/c$ and thus $\|q - q'\| \le \Delta/c$, we get that $\mathrm{d}_{\mathcal{F}}(\overline{p\,q}, \overline{p'\,q'}) \le \Delta/c$ by Observation 2.1.1, and thus $\mathrm{d}_{\mathcal{F}}(P[s,t], \overline{p'\,q'}) \le (1+c^{-1})\Delta$. Finally, $\|p' - q'\| = \|p - q\| = \lambda$ and thus the point q' lies in $\mathcal{L}_{\lambda,(1+c^{-1})\Delta}(P,p')$, implying the claim. □

We now prove a stronger version of Theorem 5.3.5, which allows us to analyze the doubling constant of (μ,ε)-curves in $\mathbb{X}^{d,k}$.

Lemma 5.3.12. *Let $k, \mu, d \in \mathbb{N}$ and $\varepsilon > 0$ be given. Further, let a (μ,ε)-curve P in $\mathbb{X}^{d,k}$ be given, as well as $\Delta > 0$ and $c \ge 1$. There is a family of curves $\mathcal{C}_P \subset \mathbb{X}^{d,k}_{\mu\varepsilon+\Delta/c}$ of size $\mathcal{O}(k\mu(10c+3)^d)^k$, such that for any (μ,ε)-curve Q with $\mathrm{d}_{\mathcal{F}}(P,Q) \le \Delta$ there is a $Q^* \in \mathcal{C}_P$ with $\mathrm{d}_{\mathcal{F}}(Q,Q^*) \le \Delta/c$.*

Proof. We construct the set $\mathcal{C}_P$ as follows. First, choose an element $(m_1,\dots,m_{k-1}) \in \{1,\dots,\mu\}^{k-1}$. Next, choose one circle center of a cover of $\mathrm{D}_\Delta(P(0))$ consisting of $\mathcal{O}((2c+1)^d)$ many balls of radius r/c, which exists by Lemma 5.3.2. Iteratively choose one point among the circle centers of a cover of $N_{\Delta/c}\left(\mathcal{L}_{m_{i-1}\varepsilon,(1+c^{-1})\Delta}(P,q^*_{i-1})\right)$ of Corollary 5.3.10, consisting of $\mathcal{O}(k(10c+3)^d)$ many balls of radius r/c as the vertex q^*_i of Q^* for $i \le k$. Then $Q^* \in \mathbb{X}^{d,k}_{\mu\varepsilon+\Delta/c}$, as for any i the fact that q^*_i lies in $N_{\Delta/c}\left(\mathcal{L}_{m_{i-1}\varepsilon,(1+c^{-1})\Delta}(P,q^*_{i-1})\right)$ implies that there is a point $q \in \mathcal{L}_{m_{i-1}\varepsilon,(1+c^{-1})\Delta}(P,q^*_{i-1})$, with $\|q^*_{i-1} - q\| = m_{i-1}\varepsilon$ and $\|q - q^*_i\| \le \Delta/c$. Hence, $\|q^*_i - q^*_{i-1}\| \le m_1\varepsilon + \Delta/c \le \mu\varepsilon + \Delta/c$. Accounting for all the choices, we have that $|\mathcal{C}_P| = \mathcal{O}(k\mu(10c+3)^d)^k$.

Let Q be a given (μ,ε)-curve, with $\mathrm{d}_{\mathcal{F}}(P,Q) \le \Delta$. The curve Q consists of $k-1$ edges and induces an ordered set $(m_1,\dots,m_{k-1}) \in \{0,\dots,\mu\}^{k-1}$ representing the lengths of the edges in order. Let $q_1,\dots,q_k$ be the vertices of Q. For all $1 \le i \le k$ it holds that $q_i \in \mathcal{L}_{m_{i-1}\varepsilon,\Delta}(P,q_{i-1})$, by construction.

As $\mathrm{d}_{\mathcal{F}}(P,Q) \le \Delta$, the first vertex q_1 lies in $\mathrm{D}_\Delta(P(0))$, and thus there is a point q^*_1 of the cover of $\mathrm{D}_\Delta(P(0))$ consisting of balls of radius r/c that lies at distance at most Δ/c

to q_1. For every subsequent q_i, by Lemma 5.3.11 and because $q_i \in \mathcal{L}_{m_{i-1}\varepsilon,\Delta}(P, q_{i-1})$, $q_i \in N_{\Delta/c}\left(\mathcal{L}_{m_{i-1}\varepsilon,(1+c^{-1})\Delta}(P, q^*_{i-1})\right)$ and thus there is a point q^*_i of the Δ/c-cover of $N_{\Delta/c}\left(\mathcal{L}_{m_{i-1}\varepsilon,(1+c^{-1})\Delta}(P, q^*_{i-1})\right)$ that is at distance at most Δ/c to q_i. This implies that there is an element Q^* (defined by exactly this choice of points) in $\mathcal{C}_P$ that has distance $\mathrm{d}_{\mathrm{d}\mathcal{F}}(Q, Q^*) \leq \Delta/c$ and thus, by Observation 2.1.2, it holds that $\mathrm{d}_{\mathcal{F}}(Q, Q^*) \leq \Delta/c$. □

With Lemma 5.3.12 at hand, Theorem 5.3.5 follows immediately.

Proof of Theorem 5.3.5. Let P be a (μ,ε)-curve in $\mathbb{X}^{d,k}$ and a value Δ be given. By Lemma 5.3.12, there is a family $\mathcal{C}_P$ of curves of size $\mathcal{O}(k\mu(43)^d)^k$ in $\mathbb{X}^{d,k}_{\mu\varepsilon+\Delta/4} \subset \mathbb{X}^{d,k}$, such that for any (μ,ε)-curve Q with $\mathrm{d}_{\mathcal{F}}(P, Q) \leq \Delta$ there is curve Q^* in $\mathcal{C}_P$ with $\mathrm{d}_{\mathcal{F}}(Q, Q^*) \leq \Delta/4$. For any $Q^* \in \mathcal{C}_P$, identify some (μ,ε)-curve $\widehat{Q^*}$ such that $\mathrm{d}_{\mathcal{F}}(Q^*, \widehat{Q^*}) \leq \Delta/4$. If no such element exists, ignore Q^*. Otherwise for any (μ,ε)-curve Q with $\mathrm{d}_{\mathcal{F}}(P, Q) \leq \Delta$ there is a curve Q^* in $\mathcal{C}_P$ with $\mathrm{d}_{\mathcal{F}}(Q, Q^*) \leq \Delta/4$, and thus by the triangle inequality, there is a (μ,ε)-curve $\widehat{Q^*}$ with $\mathrm{d}_{\mathcal{F}}(Q, \widehat{Q^*}) \leq \Delta/2$, proving the bounded doubling dimension. □

Corollary 5.3.13. *Let $k, \mu, d \in \mathbb{N}$ and $\varepsilon > 0$. The doubling dimension of the space of (μ,ε)-curves in $\mathbb{X}^{d,k}$ under the discrete Fréchet distance is bounded by $\mathcal{O}(k(d+\log(k\mu)))$.*

Proof. This is a consequence of the proof of Lemma 5.3.12 and Theorem 5.3.5. By Observation 2.1.2, for any two curves P and Q in $\mathbb{X}^{d,k}$ it holds that $\mathrm{d}_{\mathcal{F}}(P, Q) \leq \mathrm{d}_{\mathrm{d}\mathcal{F}}(P, Q)$, and thus any Δ-ball centered at a curve P under the discrete Fréchet distance is contained in the Δ-ball under the continuous Fréchet distance. In the proof of Lemma 5.3.12 the Δ-ball is covered by $\Delta/2$-balls under the discrete Fréchet distance, thus a proof similar to that of Theorem 5.3.5 implies the claim. □

5.3.3 Improvements for c-Packed Curves

In this section, we make the additional assumption that the curves in the space of (μ,ε)-curves are c-packed, which leads to an improvement of the above bounds on the doubling dimension.

Lemma 5.3.14. *Let $P \in \mathbb{X}^{d,k}$ be a c-packed curve with complexity at most k. Let $\lambda \geq 0$ and $\Delta \geq 0$ be given. Then for every $p \in \mathbb{R}^d$ there are $2c = \mathcal{O}(c)$ points $p^*_1, \ldots, p^*_{2c} \in \mathbb{R}^d$ such that*

$$\mathcal{L}_{\lambda,\Delta}(P, p) \subset \bigcup_{i=1}^{2c} \mathrm{D}_{5\Delta}(p^*_i).$$

Proof. Assume $\lambda \geq 5\Delta$, as otherwise $\mathcal{L}_{\lambda,\Delta}(P, p) \subset \mathrm{D}_{5\Delta}(p)$ clearly holds, implying the claim. For the sake of contradiction, assume that $\mathcal{L}_{\lambda,\Delta}(P, p)$ cannot be covered by $2c$ balls of radius 5Δ. This implies that there are at least $2c+1$ points $\{p_1, \ldots\} =: \mathcal{P}$ in $\mathcal{L}_{\lambda,\Delta}(P, p)$ with a pairwise distance of at least 10Δ. For any $p_i \in \mathcal{P}$, we know that $\|p_i - p\| = \lambda$ and there are values $s_i < t_i$ such that $\mathrm{d}_{\mathcal{F}}(P[s_i, t_i], \overline{p\,p_i}) \leq \Delta$. For any two distinct $p_i, p_j \in \mathcal{P}$, the points p_i, p_j and p form an isosceles triangle with side lengths λ, λ and $\|p_i - p_j\| \geq 10\Delta$. This implies that for any point q along $\overline{p\,p_i}$, the distance to p_j is at least 2Δ. This means that t_j cannot lie in the interval $[s_i, t_i]$ as $\|p_j - P(t_j)\| \leq \Delta$. This in turn implies that all intervals $[s_1, t_1], \ldots$ are pairwise disjoint. We have thus identified $2c+1$ disjoint (in the domain) subcurves of P with a total length of at least $(2c+1)(\lambda - 2\Delta)$, contained in the ball $\mathrm{D}_\lambda(p)$. However, since $\lambda \geq 5\Delta$, we have that $(2c+1)(\lambda - 2\Delta) > c\lambda$, contradicting the fact that P is c-packed. This in turn implies the claim. □

Corollary 5.3.15. *Let $k, \mu, d \in \mathbb{N}$ and $c, \varepsilon > 0$. The doubling constant of the space of c-packed (μ, ε)-curves in $\mathbb{X}^{d,k}$ is bounded by $\mathcal{O}(43^d c\mu)^k$ and thus its doubling dimension is bounded by $\mathcal{O}(k(d + \log(c\mu)))$.*

Proof. This follows from a minor modification of Theorem 5.3.5 using Lemma 5.3.14. □

5.4 Lower Bound for Doubling Dimension of (μ, ε)-Curves

In this section we want to show that the bound on the doubling dimension of $\mathcal{O}(k(d + \log(k\mu)))$ is not too pessimistic. We begin with a straightforward argument which implies a lower bound of $\Omega(d)$, before discussing the lower bound of $\Omega(k \log \mu)$, which results in a lower bound of $\Omega(d + k \log \mu)$.

The lower bound of $\Omega(d)$ follows trivially as the space of (μ, ε)-curves in $\mathbb{X}^{d,1}$ consists of every singleton in $\mathbb{R}^d$ and thus the doubling dimension of the (μ, ε)-curves in $\mathbb{X}^{d,1}$ must be at least that of $\mathbb{R}^d$.

5.4.1 Lower Bound of $\Omega(k \log \mu)$

In this section, we give a lower bound for the doubling dimension of both c-packed and non-c-packed (μ, ε)-curves in $\mathbb{X}^{d,k}$ that shows the necessity of the factor μ^k in the bound of the doubling constant. The construction we give is an adaptation of the construction given in [DKS16] that shows an unbounded doubling dimension for the space $(\mathbb{X}^{1,3}, \mathrm{d}_{\mathcal{F}})$.

Lemma 5.4.1. *Let $d = 1$. Given $\mu > 1$, $k \in \mathbb{N}$, and $m \leq k/2$, we can construct a $(\mu, 1)$-curve C, of complexity $k - 2m$, and $\binom{(k-2m)\mu}{m}$ $(\mu, 1)$-curves G_i, such that $\mathrm{d}_{\mathcal{F}}(C, G_i) \leq 1/2$ for all i. Additionally, for all $i \neq j$ there is no $(\mu, 1)$-curve X in $\mathbb{X}^{d,k}$ such that $\mathrm{d}_{\mathcal{F}}(X, G_i) \leq 1/4$ and $\mathrm{d}_{\mathcal{F}}(X, G_j) \leq 1/4$.*

The main intuition behind this proof is a variant of the main ingredient of the proof of Theorem 5.2.1. Namely, we imagine a center curve C in $\mathbb{R}^1$ (refer to Figure 5.4) that goes strictly to the right (in positive direction) for some length L. Modifying C by introducing small zig-zags of length 1 results in a new curve G that has Fréchet distance $1/2$ to C. Importantly, any curve that has Fréchet distance $1/4$ to G must also imitate the zig-zag whenever G has a zig-zag. Now imagine there are two curves G and G' resulting from C by introducing m zig-zags each. The Fréchet distance of G and G' may be $1/2$, but as long as the introduced zig-zags are sufficiently far from one another, any $(\mu, 1)$-curve that has Fréchet distance $\leq 1/4$ to G has Fréchet distance $> 1/4$ to G'. Conversely, we also show that there is also no $(\mu, 1)$-curve M such that $\{G, G'\} \subset \mathrm{D}_{1/4}(M)$.

Proof of Lemma 5.4.3. Define C via the points $(c_1, \ldots, c_{k-2m+1})$, where $c_i = (i-1)\mu$ for $i \in [k-2m+1]$. We now identify a large set of curves such that no two distinct elements of this set are at a distance at most $1/4$ to any $(\mu, 1)$-curve in $\mathbb{X}^{1,k}$. For this, choose an ordered subset $(n_1, \ldots, n_m) \subset \{0, \ldots, (k-2m)\mu - 1\}$. There are $\binom{(k-2m)\mu}{m}$ such choices. Based on the choice, we construct a curve $G_{(n_1,\ldots,n_m)}$ from C by first cutting $m+1$ pieces $C_0, \ldots, C_m$ from C, where C_0 goes from 0 to $n_1 + 1$, C_i from n_i to $n_{i+1} + 1$ for $i \in [m]$, and C_m from n_m to $((k-2m)\mu)$. For $i \in [m-1]$, construct curves T_i defined by two vertices $s_i = (n_i + 1)$, $t_i = (n_i)$. Then, retrieve $G_{(n_1,\ldots,n_m)}$ via the concatenation $C_0 \oplus T_1 \oplus C_1 \oplus \cdots \oplus T_m \oplus C_m$. The set of curves constructed in this way forms the sought-after large set of curves. We show that $G_{(n_1,\ldots,n_m)}$ consists of k edges. For this

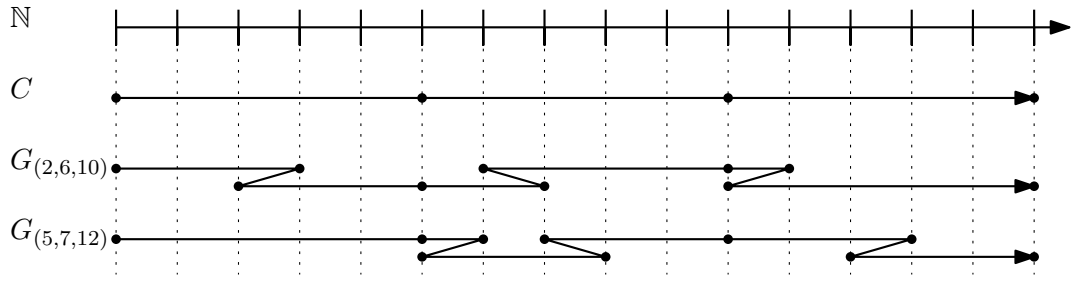


Figure 5.4: Illustration of the construction from Lemma 5.4.1 of two $(5,1)$-curves $G_{(2,6,10)}$ and $G_{(5,7,12)}$ in $\mathbb{X}^{1,9}$ that have Fréchet distance $1/2$ to the center curve C at the top.

we show that every cut introduces 2 new edges. If neither n_i nor n_i+1 are vertices of C, then we are done, so assume that n_i is a vertex of C. Then we introduce exactly two new edges: one from (n_i) to (n_i+1) and one from (n_i+1) to (n_i). Similarly, if n_i+1 is a vertex of C, we also introduce two new edges, one from (n_i+1) to (n_i) and one from (n_i) to (n_i+1). Observe that $\mathrm{d}_{\mathcal{F}}(C, G_{(n_1,\ldots,n_m)}) = 1/2$, as C goes to the right, whereas $G_{(n_1,\ldots,n_m)}$ follows C except in the introduced pieces T_i, where it goes left for a distance of 1.

Let X be a curve that has Fréchet distance at most $1/4$ to some curve $G = G_{(n_1,\ldots,n_m)}$ constructed above. Let $n_i \in (n_1,\ldots,n_m)$. Then the vertex $s_i = (n_i+1)$ and $t_i = (n_i)$ define the connecting piece T_i between C_{i-1} and C_i of G. Now, X has to first enter the interval $[n_i+3/4, n_i+5/4]$ and then $[n_i-1/4, n_1+1/4]$. As these two intervals are disjoint and $n_i < n_i+1$, the first interval lies to the right of the second interval. By construction, the vertex of G before s_i also lies to the left of t_i. Similarly, the vertex after t_i lies to the right of s_i. Hence, X has a vertex to the left of t_i, then a vertex in $[n_i+3/4, n_i+5/4]$, then a vertex inside $[n_i-1/4, n_i+1/4]$, and subsequently a vertex to the right of s_i. Thus, a curve X also has to go to the left near n_i+1. Note that as every edge of X has to have a length which is a multiple of 1, any curve that does not go left near n_i+1 has to have a Fréchet distance of at least $1/2$ to X. This has to hold for all $n_i \in (n_1,\ldots,n_m)$. Since for any two distinct such constructed curves, there is a point where one travels to the left while the other does not, this then implies the claim as there are $\binom{(k-2m)\mu}{m}$ such constructed curves. □

Corollary 5.4.2. *For $d=1$ and given μ and k, the doubling dimension of the space of $(\mu,1)$-curves in $\mathbb{X}^{d,k}$ is in $\Omega(k\log\mu)$.*

Proof. Lemma 5.4.1 with $m \leftarrow k/3$ implies the claim as

$$\binom{(k/3)\mu}{k/3} \geq \frac{((k/3)\mu)^{(k/3)}}{(k/3)^{(k/3)}} = \Omega\left(\mu^{(k/3)}\right). \qquad \square$$

Theorem 5.4.3. *For $d=1$ and given μ, k, and $\varepsilon>0$, the doubling dimension of the space of (μ,ε)-curves in $\mathbb{X}^{d,k}$ is in $\Omega(d+k\log\mu)$.*

Proof. This is a consequence of Corollary 5.4.2 and the trivial $\Omega(d)$ lower bound. □

Corollary 5.4.4. *For $d=1$, $c\geq 6$ and given μ, k, and $\varepsilon>0$, the doubling dimension of the space of c-packed (μ,ε)-curves in $\mathbb{X}^{d,k}$ is in $\Omega(d+k\log\mu)$.*

Proof. Observe that the constructed curves are 6-packed. Thus, Theorem 5.4.3 implies the claim. □

5.5 Approximate Nearest Neighbor

[HPM06] showed that one can construct a $(1+\varepsilon)$-ANN data structure in metric spaces of bounded doubling dimension.

Theorem 5.5.1 ([HPM06]). *Given a set S of n points in a metric space $\mathcal{M}$ of bounded doubling dimension ν, one can construct a data structure for answering $(1+\varepsilon)$-ANN queries. The query time is $2^{\mathcal{O}(\nu)}\log n+\varepsilon^{-\mathcal{O}(\nu)}$, the expected preprocessing time is bounded by $2^{\mathcal{O}(\nu)}n\log n$ and the space used is $2^{\mathcal{O}(\nu)}n$.*

A careful reading reveals an important specification for our purposes, namely that the doubling dimension is that of the n-point metric space defined by S induced by the metric of $\mathcal{M}$ and not the doubling dimension of the entire metric space $\mathcal{M}$. Note that by Lemma 5.5.2 the doubling dimension of the metric space induced on the subset is at most twice the doubling dimension of the ambient space. For an example where the doubling dimension increases, refer to Figure 5.5.

Lemma 5.5.2. *Let $(\mathcal{M},\mathrm{d}_{\mathcal{M}})$ be a metric space, and let S be some subset of $\mathcal{M}$. Then the doubling dimension of $(S,\mathrm{d}_{\mathcal{M}})$ is at most twice the doubling dimension of $(\mathcal{M},\mathrm{d}_{\mathcal{M}})$.*

Proof. Let ν be the doubling dimension of $\mathcal{M}$. Let $s\in S$ and $r>0$ be given and let $\mathrm{D}_r^S(s)\subset S$ be the ball in S, of radius r and centered at s, i.e., $\mathrm{D}_r^S(s)=\mathrm{D}_r^{\mathcal{M}}(s)\cap S$. The ball $\mathrm{D}_r^{\mathcal{M}}(s)$ can be covered by $(2^\nu)^2$ balls of radius $r/4$. For any such ball, check if the intersection with S is nonempty. If this is the case, pick some element from this intersection, and center a ball in S of radius $r/2$ around it. Clearly, any such larger ball contains the intersection of the smaller ball with S. Therefore, $\mathrm{D}_r^S(s)$ is contained in the union of at most $(2^\nu)^2$ balls of radius $r/2$ in S, which implies the claim, as $\log_2((2^\nu)^2)=2\nu$. □

By Theorem 5.3.5, we know that the doubling dimension of the space of (μ,ε)-curves in $\mathbb{X}^{d,k}$ is bounded. We further know that for any $\varepsilon>0$ we can map any curve of $\mathbb{X}_\Lambda^{d,k}$ into the space of $(\lceil\Lambda/\varepsilon\rceil+1,\varepsilon)$-curves in $\mathbb{X}^{d,k}$ with a distortion of at most $\varepsilon/2$, by Lemma 5.2.3. Hence, Theorem 5.5.1 together with Lemma 5.5.2 imply Theorem 5.1.1, a central piece to constructing a data structure solving the $(1+\varepsilon)$-ANN problem for polygonal curves under the Fréchet distance.

Theorem 5.1.1. *Given a set S of n polygonal curves in $\mathbb{X}_\Lambda^{d,k}$ and parameters $\varepsilon\in(0,1]$ and $\varepsilon'>0$, one can construct a data structure that for given $q\in\mathbb{X}^{d,k}$ outputs an element $s^*\in S$ such that for all $s\in S$ it holds that $\mathrm{d}_{\mathcal{F}}(s^*,q)\le(1+\varepsilon)\mathrm{d}_{\mathcal{F}}(s,q)+\varepsilon'$. The query time is $\mathcal{O}\left(2^{\mathcal{O}(d)}k(1+\Lambda/\varepsilon')\right)^k\log n+\mathcal{O}\left(2^{\mathcal{O}(d)}k(1+\Lambda/\varepsilon')\right)^{-k\log(\varepsilon)}$, the expected preprocessing time is $\mathcal{O}\left(2^{\mathcal{O}(d)}k(1+\Lambda/\varepsilon')\right)^k n\log n$ and the space used is $\mathcal{O}\left(2^{\mathcal{O}(d)}k(1+\Lambda/\varepsilon)\right)^k n$.*

Proof. Define $\hat{\varepsilon}=\varepsilon'/2$. Let $\mu=\lceil\Lambda/\hat{\varepsilon}\rceil+1=\Theta(1+\Lambda/\varepsilon')$. We begin by simplifying every polygonal curve $s\in S$ via Lemma 5.2.3, resulting in a set S' of $(\mu,\hat{\varepsilon})$-curves. This takes $\mathcal{O}\left(\log(\mu)nk\right)$ time. As S' lies in the space of $(\mu,\hat{\varepsilon})$-curves, the doubling dimension of the set S' with the Fréchet distance is bounded by $\nu=\mathcal{O}(k(d+\log(k(1+\Lambda/\varepsilon'))))$ via Theorem 5.3.5 and Lemma 5.5.2. Note that for every $s\in S$ and its simplification $s'\in S'$ it holds that $\mathrm{d}_{\mathcal{F}}(s,s')\le\hat{\varepsilon}/2$. We apply Theorem 5.5.1 to the set S' and ε. Note that Theorem 5.5.1 assumes that the distance between any two points in the metric space of (μ,ε)-curves can be computed in $\mathcal{O}(1)$ time. The computation of the continuous Fréchet distance takes polynomial time in k which is dominated by 2^k and hence the running

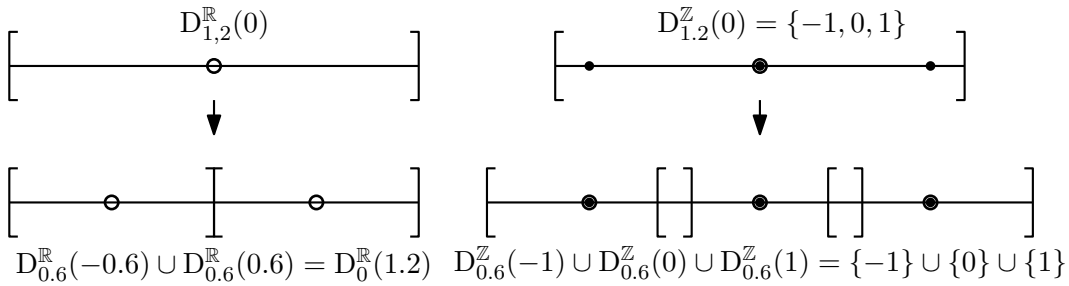


Figure 5.5: Example of a subset $\mathbb{Z}$ of the metric space $\mathbb{R}$ whose doubling dimension is larger than that of its ambient space. The disk centers are marked by circles.

time is as claimed. We then query the data structure with q, returning an element $\widehat{s'}$ such that for every $s' \in S'$ it holds that $\mathrm{d}_{\mathcal{F}}(q, \widehat{s'}) \leq (1+\varepsilon)\mathrm{d}_{\mathcal{F}}(q, s')$. Lastly, the element of S returned by the data structure will be the element $\widehat{s} \in S$ which corresponds to $\widehat{s'}$. We then get for every $s \in S$ that

$$\begin{aligned}\mathrm{d}_{\mathcal{F}}(q, \widehat{s}) &\leq \mathrm{d}_{\mathcal{F}}(q, \widehat{s'}) + \hat{\varepsilon}/2 \leq (1+\varepsilon)\mathrm{d}_{\mathcal{F}}(q, s') + \hat{\varepsilon}/2 \leq (1+\varepsilon)(\mathrm{d}_{\mathcal{F}}(q, s) + \hat{\varepsilon}/2) + \hat{\varepsilon}/2 \\ &\leq (1+\varepsilon)\mathrm{d}_{\mathcal{F}}(q, s) + \hat{\varepsilon} + \varepsilon\hat{\varepsilon}/2 = (1+\varepsilon)\mathrm{d}_{\mathcal{F}}(q, s) + \hat{\varepsilon}(1+\varepsilon/2) \\ &\leq (1+\varepsilon)\mathrm{d}_{\mathcal{F}}(q, s) + \varepsilon'.\end{aligned}$$

□

To get rid of the additive error, we want to set ε' to $\mathcal{O}(\min_{s \neq s' \in S} \mathrm{d}_{\mathcal{F}}(s, s'))$. As ε' impacts the running time, we reformulate the running time in terms of the spread.

Lemma 5.5.3. *Given a set of curves $S \in \mathbb{X}^{d,k}$ with pairwise non-zero Fréchet distance, then*

$$\mathcal{G}(S)^{-1} = \mathcal{O}(\Phi(S))$$

where $\Phi(S)$ denotes the spread of the set of vertices and edges of curves in S.

Proof. Note that the Fréchet distance between two curves P and Q is approximated up to a constant by the Euclidean distance of either one vertex of P and one vertex of Q or the Euclidean distance of a vertex and an edge (one of P and one of Q) as sets. Thus

$$\min_{s \neq s' \in S} \mathrm{d}_{\mathcal{F}}(s, s') = \Omega\left(\min_{\substack{o,o' \in V(S) \cup E(S) \\ \mathrm{d}(o,o') > 0}} \mathrm{d}(o, o')\right)$$

where $V(S)$ denotes the set of vertices and $E(S)$ denotes the set of edges defining the curves in S. Further note that as the length of any edge of a curve s in S is defined as the distance of two vertices defining s. As such, observe that

$$\max_{s \in S, e \in E(s)} \|e\| \leq \max_{p,q \in V(S)} \mathrm{d}(p, q) \leq \max_{o,o' \in V(S) \cup E(S)} \mathrm{d}(o, o').$$

Thus the claim follows. □

Theorem 5.1.4. *Given a set S of n polygonal curves in $\mathbb{X}^{d,k}$ and $\varepsilon \in (0, 1]$, one can construct a data structure answering $(1+\varepsilon)$-approximate nearest neighbor queries. The query time is $F(d, k, S, \varepsilon) \log n + F(d, k, S, \varepsilon)^{-\log(\varepsilon)}$, the expected preprocessing time is $F(d, k, S, \varepsilon) n \log n$ and the space required for the data structure is at most $F(d, k, S, \varepsilon) n$, where $F(d, k, S, \varepsilon) = \mathcal{O}\left(2^{\mathcal{O}(d)} k (1 + \mathcal{G}(S)^{-1} \varepsilon^{-1})\right)^k$.*

Proof. Let $\varepsilon' = \varepsilon/4$ and $\varepsilon'' = \varepsilon' \left(\min_{s \neq s' \in S} \mathrm{d}_{\mathcal{F}}(s, s')\right)$. Let $E(S)$ be the set of edges of curves in S and let further $\Lambda = \max_{e \in E(S)} \|e\|$, thus clearly $S \subset \mathbb{X}^{d,k}_{\Lambda}$. We then apply Theorem 5.1.1 with ε' and ε'' resulting in the described data structure. Let $q \in \mathbb{X}^{d,k}$ be given. Let s^* be the element in S minimizing the distance to q. Querying the data structure with q results in an element $\widehat{s}$ with the the property that

$$\mathrm{d}_{\mathcal{F}}(q, \widehat{s}) \leq (1+\varepsilon')\mathrm{d}_{\mathcal{F}}(q, s^*) + \varepsilon' \left(\min_{s \neq s' \in S} \mathrm{d}_{\mathcal{F}}(s, s') \right).$$

Assume first that $(2+\varepsilon')\mathrm{d}_{\mathcal{F}}(q, s^*) < (1-\varepsilon') \min_{s \neq s' \in S} \mathrm{d}_{\mathcal{F}}(s, s')$. Then for every $s' \neq s^*$ we know that

$$\begin{aligned}
\mathrm{d}_{\mathcal{F}}(q, s') &\geq \mathrm{d}_{\mathcal{F}}(s^*, s') - \mathrm{d}_{\mathcal{F}}(q, s^*) \geq \min_{s \neq s' \in S} \mathrm{d}_{\mathcal{F}}(s, s') - \mathrm{d}_{\mathcal{F}}(q, s^*) \\
&> (1-\varepsilon') \left(\min_{s \neq s' \in S} \mathrm{d}_{\mathcal{F}}(s, s') \right) + \varepsilon' \left(\min_{s \neq s' \in S} \mathrm{d}_{\mathcal{F}}(s, s') \right) - \mathrm{d}_{\mathcal{F}}(q, s^*) \\
&= (1+\varepsilon')\mathrm{d}_{\mathcal{F}}(q, s^*) + \varepsilon' \left(\min_{s \neq s' \in S} \mathrm{d}_{\mathcal{F}}(s, s') \right)
\end{aligned}$$

and thus $\widehat{s} = s^*$, implying $\mathrm{d}_{\mathcal{F}}(q, \widehat{s}) \leq (1+\varepsilon)\mathrm{d}_{\mathcal{F}}(q, s^*)$.

If instead $(2+\varepsilon')\mathrm{d}_{\mathcal{F}}(q, s^*) \geq (1-\varepsilon') \min_{s \neq s' \in S} \mathrm{d}_{\mathcal{F}}(s, s')$, then we observe that

$$\begin{aligned}
\mathrm{d}_{\mathcal{F}}(q, \widehat{s}) &\leq (1+\varepsilon')\mathrm{d}_{\mathcal{F}}(q, s^*) + \varepsilon' \left(\min_{s \neq s' \in S} \mathrm{d}_{\mathcal{F}}(s, s') \right) \\
&\leq (1+\varepsilon')\mathrm{d}_{\mathcal{F}}(q, s^*) + \varepsilon' \left(\frac{2+\varepsilon'}{1-\varepsilon'} \right) \mathrm{d}_{\mathcal{F}}(q, s^*).
\end{aligned}$$

Now since $\varepsilon \leq 1$, we know that $\varepsilon' \leq 1/4$ and thus $\frac{2+\varepsilon'}{1-\varepsilon'} \leq 3$. This then concludes the case-distinction, as

$$\begin{aligned}
\mathrm{d}_{\mathcal{F}}(q, \widehat{s}) &\leq (1+\varepsilon')\mathrm{d}_{\mathcal{F}}(q, s^*) + \varepsilon' \left(\frac{2+\varepsilon'}{1-\varepsilon'} \right) \mathrm{d}_{\mathcal{F}}(q, s^*) \\
&\leq (1+4\varepsilon')\mathrm{d}_{\mathcal{F}}(q, s^*) = (1+\varepsilon)\mathrm{d}_{\mathcal{F}}(q, s^*).
\end{aligned}$$

For the claimed bound of the running time, observe that $\Lambda / \min_{s \neq s' \in S} \mathrm{d}_{\mathcal{F}}(s, s') = \mathcal{G}(S)^{-1}$. Hence, as $\varepsilon' = \Theta(\varepsilon)$ and $\varepsilon'' = \Theta(\varepsilon \left(\min_{s \neq s' \in S} \mathrm{d}_{\mathcal{F}}(s, s')\right))$, the preprocessing time, query time, and space is as claimed. □

Corollary 5.1.5. *Given a set S of n polygonal curves in $\mathbb{X}^{d,k}$ and $\varepsilon \in (0, 1]$ one can construct a data structure answering $(1+\varepsilon)$-approximate nearest neighbor queries. The query time is $F(d, k, S, \varepsilon) \log n + F(d, k, S, \varepsilon)^{-\log(\varepsilon)}$, the expected preprocessing time is $F(d, k, S, \varepsilon) n \log n$ and the space required for the data structure is at most $F(d, k, S, \varepsilon) n$, where $F(d, k, S, \varepsilon) = \mathcal{O}\left(2^{\mathcal{O}(d)} k \Phi(S) \varepsilon^{-1}\right)^k$, where $\Phi(S)$ denotes the spread of the set of vertices and edges of the curves in S.*

Proof. The spread of any collection of sets is at least 1, implying that $(1 + \Phi(S)\varepsilon^{-1}) = \mathcal{O}(\Phi(S)\varepsilon^{-1})$. Thus, the claim is a consequence of Theorem 5.1.4 and Lemma 5.5.3. □

Chapter 6

The k-Shortcut Fréchet Distance

In this chapter we study the k-shortcut Fréchet distance and whether we can design efficient exact and approximate algorithms for its computation.

The main content of this chapter previously appeared as the paper *On Computing the k-Shortcut Fréchet Distance* [CD24] by Jacobus Conradi and Anne Driemel, which was published in the journal *ACM Transactions on Algorithms*. An extended abstract was also published in the *Proceedings of the* 49^{th} *EATCS International Colloquium on Automata, Languages and Programming (ICALP 2022)* [CD22].

6.1 Introduction

The shortcut Fréchet distance was introduced in [DHP12], in which a near-linear time $(3+\varepsilon)$-approximation algorithm for the class of c-packed curves was given. However, they only consider shortcuts that start and end at vertices of the base curve. In the unrestricted setting, where shortcuts are allowed to start and end anywhere along the curve, [BDS14] presented a 3-approximation algorithm for the decision variant—i.e., decide whether $\mathrm{d}_{\mathcal{S}}(P,Q) \leq \Delta$ for given curves P and Q, and value Δ—with running time in $\mathcal{O}(n^3 \log n)$. They complemented this result with a construction showing that solving the decision variant exactly is NP-hard via a reduction from SUBSET-SUM. Prior to our work, exact algorithms for the shortcut Fréchet distance $\mathrm{d}_{\mathcal{S}}(\cdot,\cdot)$ and the parametrized k-shortcut Fréchet distance $\mathrm{d}^k_{\mathcal{S}}(\cdot,\cdot)$ with unrestricted shortcuts had not been studied. Obtaining the exact algorithm was surprisingly simple, once the relevant techniques were combined in the right way.

6.1.1 Results

In Section 6.3, we present an exact algorithm for deciding whether the k-shortcut Fréchet distance is smaller than a given threshold Δ. This algorithm extends to the non-parameterized variant by setting $k = n-1$. Our first main result is the following theorem.

Theorem 6.1.1. *Let T and B be two polygonal curves in the plane with overall complexity n, together with a value $\Delta > 0$. There exists an algorithm with running time in $\mathcal{O}\left(kn^{2k+2}\log n\right)$ and space in $\mathcal{O}\left(kn^{2k+2}\right)$ that decides whether $d^k_{\mathcal{S}}(B,T) \leq \Delta$.*

As the k-shortcut Fréchet distance is directional, that is, $\mathrm{d}^k_{\mathcal{S}}(P,Q)$ is obtained by introducing shortcuts to P only, we will refer to the curve to which we introduce shortcuts

as the *base curve* B, and the other curve is called the *target curve* T. Deciding whether $\mathrm{d}_{\mathcal{S}}^{k}(B,T) \leq \Delta$ can be thought of as modifying the base curve to look (up to Δ) like the target curve.

Recall that the classical algorithm for the decision problem of the Fréchet distance scans the Δ-free space one cell at a time from bottom left to top right, computing the set of points in each cell that is reachable via a monotone path starting at $(0,0)$. We will extend this algorithm by scanning the Δ-free space a total of $k+1$ times. In the initial round we will simply proceed as in the classical algorithm computing the set of points reachable by a monotone path. In the i^{th} round we consider the points reachable in the Δ-free space when we are allowed to modify the curve B by i shortcuts. Shortcuts take the form of 'hops' or 'tunnels' in the Δ-free space which allow reaching new, previously unreachable points. In the i^{th} round we compute the set of points reachable by *exactly* i such tunnels based on the set of points reachable in the previous round. To compute the set of points reachable by a tunnel, we make use of so-called line-stabbing wedges introduced in [GHMS94] as was employed in the approximate decision algorithm for the shortcut Fréchet distance in [BDS14].

Our algorithm has exponential running time. Specifically, the subset of points of some cell reachable by exactly s shortcuts may fragment into roughly n^s components. This fragmentation may propagate to cells in later rounds, resulting in a running time of $\mathcal{O}\left(kn^{2k+2}\log n\right)$. In Section 6.4, we give evidence that this high complexity due to fragmentation is not an artifact of our algorithm, but may be inherent to the problem itself. For this, we assume that the exponential time hypothesis (ETH) holds, which states that 3-SAT in n variables cannot be solved in $2^{o(n)}$ time [IP99]. We obtain the following conditional lower bound:

Theorem 6.1.2. *Unless ETH fails, there is no algorithm for the k-shortcut Fréchet distance decision problem in $\mathbb{R}^d$ for $d \geq 2$ with running time $n^{o(k)}$.*

Our conditional lower bound of Theorem 6.1.2 is obtained via a reduction from a variant of the k-SUM problem, called k-Table-SUM. In this problem one is given k sets of integers $A_1, \ldots, A_k$ and a target values σ and is asked to produce $a_i \in A_i$ such that $\sum_i a_i = \sigma$. We construct two curves B and T for a given k-Table-SUM instance such that $\mathrm{d}_{\mathcal{S}}^{4k+2}(B,T) = 1$ iff the k-Table-SUM instance has a solution. Our construction is based on the NP-hardness reduction presented in [BDS14]. Their construction reduces SUBSET-SUM to the decision of whether the shortcut Fréchet distance between two curves is at most 1. SUBSET-SUM is related to the k-Table-SUM problem in that it asks for a single set A and target value σ whether there is a subset $S \subset A$ such that $\sum_{a \in S} a = \sigma$. Their construction implicitly encodes partial solutions for the SUBSET-SUM instance as reachable intervals on the edges of one of the curves. Via an intricate construction of a 'choice gadget', every fourth shortcut corresponds to the binary choice of whether an item from A should be added to S. This leads to a number of shortcuts linear in the number of items in A. We instead give a new construction for this 'choice gadget' such that every fourth shortcut is instead presented with n choices instead, where the j^{th} option corresponds to selecting the j^{th} item from some A_i.

In light of the above results, it is interesting to consider approximation algorithms and realistic input assumptions for this problem. In Section 6.5, we show that there is an efficient approximation algorithm for the decision problem. If we assume the input curves to be c-packed, we obtain a near-linear time algorithm for constant k:

Theorem 6.1.3. *Let T and B be two c-packed curves in the plane with overall complexity n, together with values $\varepsilon \in (0,1]$ and $\Delta > 0$. There exists an algorithm with running time in $\mathcal{O}\left(kcn\varepsilon^{-5}\log^2\left(n\varepsilon^{-1}\right)\right)$ and space in $\mathcal{O}\left(kcn\varepsilon^{-4}\log^2\left(n\varepsilon^{-1}\right)\right)$ which outputs one of the following: (i) $d^k_{\mathcal{S}}(B,T) \leq (3+\varepsilon)\Delta$ or (ii) $d^k_{\mathcal{S}}(B,T) > \Delta$. In any case, the output is correct.*

In general, any curve of complexity n is c-packed for some $c \leq 2n$. Thus, the theorem also implies a running time of $\mathcal{O}\left(kn^2\varepsilon^{-5}\log^2\left(n\varepsilon^{-1}\right)\right)$ for any two curves in the plane.

The main ingredient in the approximation algorithm is a central property of tunnels originally observed in [DHP12]. They show that for a 3-approximation, it suffices to propagate the reachable points by $(i-1)$ tunnels to a cell in the i^{th} round of the algorithm from only a single point. This completely negates the observed propagation of the fragmentation. To improve the running time, we consider an approximation of the line-stabbing wedge which we obtain via a data structure from [DHP12] answering distance queries of the form $\mathrm{d}_{\mathcal{F}}(\overline{p\,q}, T[s,t])$ for line segments defined by $p, q \in \mathbb{R}^d$ and subcurves of T defined by $0 \leq s \leq t \leq 1$ efficiently. We combine this approximate line-stabbing wedge with the property of c-packed curves that the complexity of the free space diagram of two c-packed curves is only linear in cn when the curves are appropriately simplified. This property was originally observed in [DHW12]. We discussed variants of this result in Section 4.4.3.

6.2 Preliminaries

In the following, we will usually fix two curves T and B for which we want to decide whether $\mathrm{d}^k_{\mathcal{S}}(B,T) \leq \Delta$. We may omit T and B when referring to the Δ-free space, writing $\mathcal{D}_\Delta$ instead of $\mathcal{D}_\Delta(B,T)$. We begin by making the aforementioned concepts of tunnels and the reachable space when introducing s shortcuts more concrete.

Definition 6.2.1 (Reachable Space)**.** We define the (Δ, s)-reachable free space of B and T as

$$\mathcal{R}_{\Delta,s}(B,T) = \{(x_p, y_p) \in [0,1]^2 \mid \mathrm{d}^s_{\mathcal{S}}(T[0,x_p], B[0,y_p]) \leq \Delta\}.$$

We denote the intersection of $\mathcal{R}_{\Delta,s}(B,T)$ with the cell $C_{i,j}$ corresponding to edge i and j of T and B respectively, by $\mathcal{R}^{(i,j)}_{\Delta,s}(B,T) = \mathcal{R}_{\Delta,s}(B,T) \cap C_{i,j}$. We call the intersection $\mathcal{R}^{(i,j)}_{\Delta,s}(B,T) \cap C_{a,b}$ for any $(a,b) \in \{(i-1,j),(i,j-1),(i+1,j),(i,j+1)\}$ a reachability interval of the cell $C_{i,j}$. In particular for $(a,b) \in \{(i-1,j),(i,j-1)\}$ we call them incoming reachability intervals and for $(a,b) \in \{(i+1,j),(i,j+1)\}$ we call them outgoing reachability intervals.

We will write $\mathcal{R}_{\Delta,s}$ and $\mathcal{R}^{(i,j)}_{\Delta,s}$ whenever T and B are fixed. Observe that the reachability intervals for every cell $C_{i,j}$ and s are contained in the boundary set $\partial C_{i,j}$, and each reachability interval is described by a (possibly empty) single interval, since any two points in the reachability interval can be connected via a monotone path that stays inside the Δ-free space.

Definition 6.2.2 (Tunnel)**.** A tunnel $\tau(p,q)$ is a pair of points $p = (x_p, y_p)$ and $q = (x_q, y_q)$ in the parametric space of B and T, with $x_p \leq x_q$ and $y_p \leq y_q$. $\tau(p,q)$ is called feasible if p and q are in $\mathcal{D}_\Delta$. We say that a tunnel is proper, if the endpoints of the shortcut do not lie on the same edge of B. We say a tunnel has a price $\mathrm{prc}(\tau(p,q)) = \mathrm{d}_{\mathcal{F}}(T[x_p,x_q], \overline{B}[y_p,y_q])$. Refer to Figure 6.1.

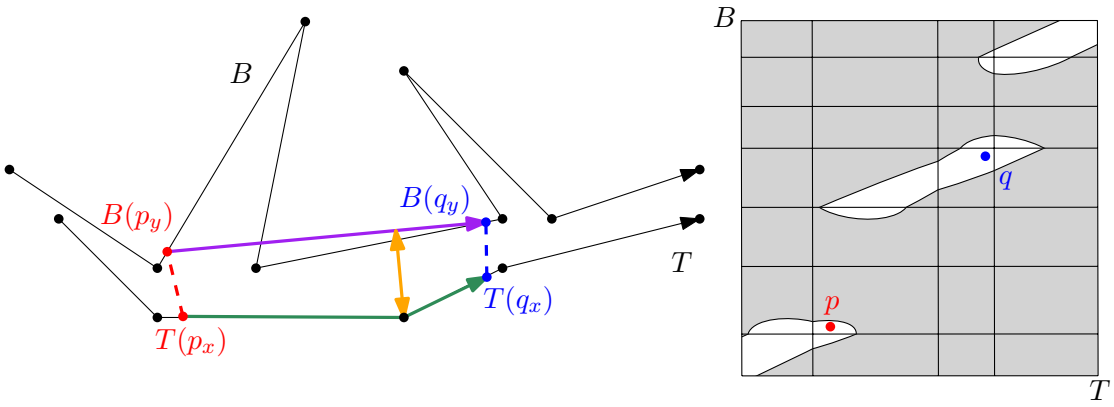


Figure 6.1: Free space diagram for a base curve B and a target curve T. The figure shows a feasible proper tunnel $\tau(p, q)$. The shortcut $\overline{B}[p_y, q_y]$ is shown in purple, and the subcurve $T[p_x, p_y]$ in green. The price of $\tau(p, q)$ is the Fréchet distance of the shortcut and the subcurve.

When considering any k-shortcut curve B' of B and any traversal (f, g) of B' and T with associated cost Δ, then (f, g) induces traversals (f', g') with associated cost at most Δ on every shortcut $\overline{B}[s, t]$ and some corresponding subcurve $T[u, v]$ of T. In particular, $\tau((u, s), (v, t))$ has price at most Δ. Thus there is a correspondence between monotone paths in the Δ-free space augmented by s tunnels and $\mathcal{R}_{\Delta,s}$. The k-shortcut Fréchet distance of T and B is at most Δ if and only if $(1, 1) \in \mathcal{R}_{\Delta,k}(B, T)$. We observe that similar to the classical decision problem of the Fréchet distance, the problem of deciding whether the k-shortcut Fréchet distance is at most Δ reduces to the problem of finding a monotone path augmented with tunnels in the free space diagram starting at $(0, 0)$ and ending at $(1, 1)$.

Observe that any tunnel $\tau(p, q)$ with $p = (x_p, y_p)$ and $q = (x_q, y_q)$ that is not proper induces a traversal of $B[y_p, y_q] = \overline{B}[y_p, y_q]$ and $T[x_p, x_q]$. Thus we can omit the tunnel from any monotone path augmented by it and replace it with a monotone path from p to q in $\mathcal{D}_\Delta$. Therefore it suffices to only consider monotone paths with proper tunnels.

Definition 6.2.3 (Monotone Path with Tunnels)**.** A monotone path with k proper tunnels in the Δ-free space of two curves consists of $k + 1$ monotone (in x and y) paths in the Δ-free space from s_i to t_i for $i \in [k + 1]$, where $s_1 = (0, 0)$, t_i lies to the left and below s_{i+1} for $i \in [k]$, and the tunnels $\tau(t_i, s_{i+1})$ are proper for $i \in [k]$.

Observation 6.2.4. *Let T and B be two polygonal curves. The set $\mathcal{R}_{\Delta,s}(B, T)$ is exactly the set of points $p \in \mathcal{D}_\Delta(B, T)$ such that there exists a monotone path with at most s proper tunnels ending in p, where each tunnel has price at most Δ.*

A cell is reachable via a tunnel $\tau((p_x, p_y), (q_x, q_y))$ only if the tunnel has price at most Δ. By definition, the tunnel has price at most Δ iff $\mathrm{d}_{\mathcal{F}}(T[x_p, x_q], \overline{B}[y_p, y_q]) \leq \Delta$. And $\mathrm{d}_{\mathcal{F}}(T[x_p, x_q], \overline{B}[y_p, y_q]) \leq \Delta$ if $\overline{B(y_p)\, B(y_q)}$ 'stabs through' the ordered set of Δ-disks centered at the vertices of $T[x_p, x_q]$. This is formalized via the notion of an ordered stabber (and line-stabbing wedge) as defined in [GHMS94] (compare Definition 5.3.3):

Definition 6.2.5 (Line-Stabbing Wedge)**.** For a sequence $O_1, \ldots, O_n$ of convex objects, an ordered stabber of the sequence is a line segment $l(x) = (1 - x)s + xt$ from s to t such that points $0 \leq x_1 \leq x_2 \leq \ldots \leq x_n \leq 1$ exist with $p_i = l(x_i) \in O_i$. We call p_i the

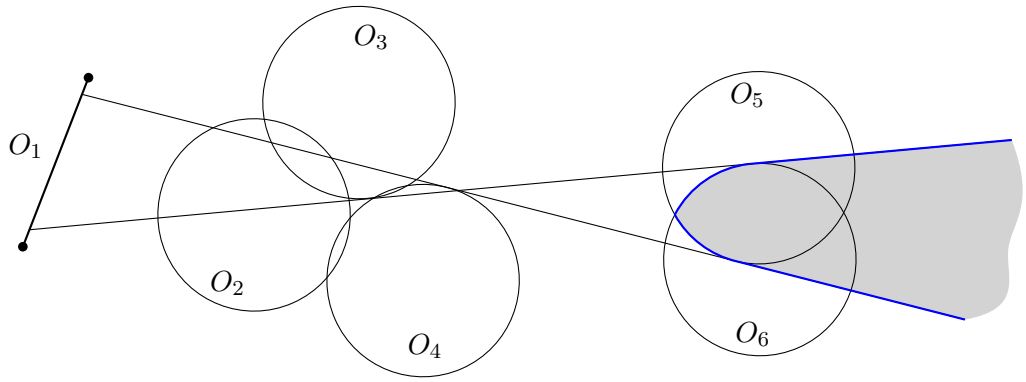


Figure 6.2: Illustration of the line-stabbing wedge for a line segment O_1 and disks $O_2, \ldots, O_6$. The line-stabbing wedge is shown in gray, with its boundary in blue.

realizing points of l. We say that l stabs through $O_1, \ldots, O_n$. We call the set of points t that are endpoints of ordered stabbers of $O_1, \ldots, O_n$ the line-stabbing wedge of this sequence.

[GHMS94] provided an algorithm to compute the line-stabbing wedge for a sequence of n unit-disks and convex polygons of constant size, with running time $\mathcal{O}(n \log n)$. This line-stabbing wedge is described by $\mathcal{O}(n)$ circular arcs, polygonal chains, and two tangents that go to infinity (see Figure 6.2). The following observation is morally the same as Observation 5.3.4.

Observation 6.2.6. *Let B, T and Δ be given. Let $v_1, \ldots, v_n$ be the vertices of T. For any feasible tunnel $\tau(p, q)$ with $p = (x_p, y_p) \in C_{a,b}$ and $q = (x_q, y_q) \in C_{i,j}$, it holds that $\overline{B}[y_p, y_q]$ stabs through the ordered set $\{\mathrm{D}_\Delta(v_{a+1}), \ldots, \mathrm{D}_\Delta(v_i)\}$, iff $\mathrm{prc}(\tau(p, q)) \leq \Delta$.*

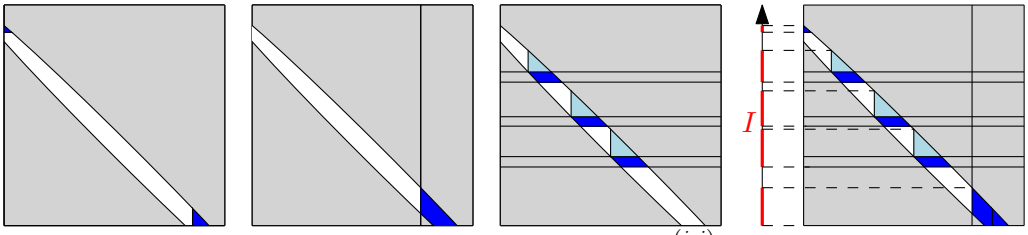
6.3 Exact Decider Algorithm

In this section we describe an exact decider algorithm for the k-shortcut Fréchet distance for two polygonal curves. In Section 6.3.1, we describe the algorithm before analyzing its correctness and running time in Section 6.3.2.

6.3.1 Description of the Algorithm

The input of the algorithm is the parameter k, a value Δ, and the two polygonal curves T and B in the plane with n_1 and n_2 edges respectively. The algorithm iterates over the Δ-free space of B and T in $k+1$ rounds. In round s we compute the set of points that are reachable by a monotone path with s proper tunnels, based on the points that are reachable by a monotone path with $s-1$ proper tunnels computed in the previous round. In each round, we handle the cells of the free space diagram in a row-by-row order, and within each row from left to right. For every cell $C_{i,j}$ we consider three possible ways that a monotone path with proper tunnels can enter.

1. A monotone path can enter the cell $C_{i,j}$ from the neighboring cell $C_{i-1,j}$ to the left or from the neighboring cell $C_{i,j-1}$ below. This does not directly involve a tunnel.



a) $N^s_{i,j}$ b) $V^s_{i,j}$ c) $Q(D^s_{i,j}) \cap \mathcal{D}^{(i,j)}_\delta$ d) $Q(N^s_{i,j} \cup D^s_{i,j} \cup V^s_{i,j}) \cap \mathcal{D}^{(i,j)}_\delta$

Figure 6.3: Example of the composition of the reachable space within a free space cell. The fragmentation of the reachable space in this cell $P^s_{i,j} = Q(N^s_{i,j} \cup D^s_{i,j} \cup V^s_{i,j}) \cap \mathcal{D}^{(i,j)}_\Delta$ results in a large family of intervals I when projecting $P^s_{i,j}$ onto the edge e_i of B, which occurs when computing $P^{s+1}_{k,l}$ for some $i < k$ and $j < l$.

2. A monotone path can reach $C_{i,j}$ with a proper tunnel. We distinguish between vertical and diagonal tunnels (compare [BDS14, DHP12] for a similar distinction).
   (i) The tunnel may start in any cell $C_{a,b}$ with $a < i$ and $b < j$. We call this a diagonal tunnel.
   (ii) The tunnel may start in any cell $C_{i,b}$ for $b < j$. We call this a vertical tunnel.

Note that we do not consider horizontal tunnels starting in a cell $C_{a,j}$ with $a < i$, since we only consider proper tunnels. Using this distinction of diagonal and vertical tunnels, we will describe how to compute the set of points reachable by a monotone path with s proper tunnels, for each cell. We denote the set computed by the algorithm for cell $C_{i,j}$ in round s with $P^s_{i,j}$. The (Δ, s)-reachable space corresponds to the union of these sets over all rounds $\mathcal{R}^{(i,j)}_{\Delta,s} = \bigcup_{0 \le s' \le s} P^{s'}_{i,j}$.

To simplify the description of the algorithm, we use the following set function which receives a set $P \subseteq C_{i,j}$ for some cell $C_{i,j}$ and extends P to all points above and to the right of it:

$$Q(P) = \{(x, y) \in [0, 1]^2 \mid \exists (a, b) \in P \text{ such that } a \le x \text{ and } b \le y\}.$$

We usually intersect this set with $\mathcal{D}^{(i,j)}_\Delta$ to obtain all points that are reachable from a point of P by a monotone path that stays inside the Δ-free space of this cell. Figure 6.3 c) shows an example of the resulting set. Note that the boundary of the resulting set can be described by pieces of the boundary of $\mathcal{D}^{(i,j)}_\Delta$, pieces of the boundary of P, and horizontal and vertical line segments.

Now for the algorithm, initially define $P^0_{1,1} = \mathcal{D}^{(1,1)}_\Delta$, if $(0, 0) \in \mathcal{D}_\Delta$, and otherwise $P^0_{1,1} = \emptyset$. Next, consider some cell $C_{i,j}$ in some round s in which we compute $P^s_{i,j}$. We describe the three subroutines with which we compute $P^s_{i,j}$:

Step 1: Neighboring Cells Since we traverse the cells of the diagram in a lexicographical order, we have already computed the (possibly empty) sets $P^s_{i-1,j}$ and $P^s_{i,j-1}$ by the time we handle cell $C_{i,j}$ in round s. Therefore, we compute the incoming reachability intervals by intersecting $P^s_{i-1,j}$ and $P^s_{i,j-1}$ with $C_{i,j}$. We then apply the function Q to the union of these sets (refer to Figure 6.3 a)) obtaining

$$N^s_{i,j} = Q\left(P^s_{i-1,j} \cap C_{i,j}) \cup (P^s_{i,j-1} \cap C_{i,j})\right) \cap \mathcal{D}^{(i,j)}_\Delta.$$

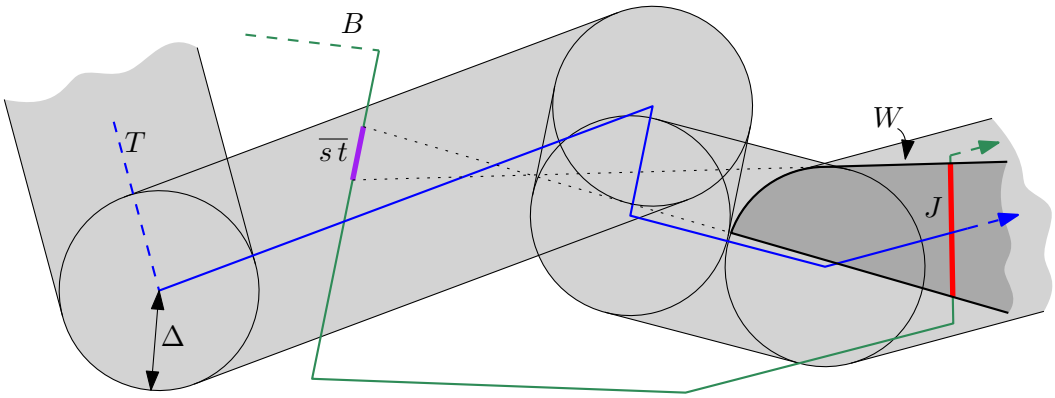


Figure 6.4: Example of the set J (in red) computed by the DIAGONALTUNNEL procedure.

Step 2 (i): Diagonal Tunnels (only for $s \geq 1$) We invoke the following procedure for every $a < i$ and $b < j$ with $P^{s-1}_{a,b}$.

The procedure is given a set of points $P^{s-1}_{a,b}$ in the Δ-free space $\mathcal{D}^{(a,b)}_\Delta$ and computes all points in $\mathcal{D}^{(i,j)}_\Delta$ that are endpoints of tunnels starting in $P^{s-1}_{a,b}$ with price at most Δ. The procedure first projects $P^{s-1}_{a,b}$ onto the edge e_b of the base curve. The resulting set consists of disjoint line segments $I = \{\overline{s_1\,t_1}, \ldots\}$ along e_b (refer to Figure 6.3 d)). The procedure then computes the line-stabbing wedge W through $\overline{s_1\,t_1}$ and disks $\mathrm{D}_\Delta(v_{a+1}), \ldots, \mathrm{D}_\Delta(v_i)$ centered at vertices of T. The wedge W is then intersected with the edge e_j, resulting in a set J on e_j corresponding to a horizontal slab in $C_{i,j}$ (compare Figure 6.3 c) and Figure 6.4). This resulting set is intersected with $\mathcal{D}^{(i,j)}_\Delta$ obtaining all endpoints of feasible shortcuts with price at most Δ starting in $\overline{s_1\,t_1}$. The procedure performs the above steps for every line segment $\overline{s\,t} \in I$ and returns the union of these sets. An illustration of the resulting set can be found in Figure 6.3 c). Repeating this procedure for every $a < i$ and $b < j$ and taking the union over all resulting sets of points results in a subset of $\mathcal{D}^{(i,j)}_\Delta$ which we refer to as $D^s_{i,j}$.

Step 2 (ii): Vertical Tunnels (only for $s \geq 1$) Let p be a point in $\bigcup_{l \leq j-1} P^{s-1}_{i,l}$ with minimal x-coordinate. A feasible vertical tunnel always has price at most Δ. Hence, we compute the intersection of a halfplane that lies to the right of the vertical line at p with the Δ-free space in $C_{i,j}$. We denote this set with $V^s_{i,j}$. Refer to Figure 6.3 b) for an example.

Putting It All Together We compute the set $P^s_{i,j}$ by taking the union of the computed sets and extending this set by using the function Q defined above:

$$P^s_{i,j} = Q(N^s_{i,j} \cup D^s_{i,j} \cup V^s_{i,j}) \cap \mathcal{D}^{(i,j)}_\Delta.$$

After the $(k+1)^{\text{th}}$ round we have computed $P^s_{i,j}$ for every $i \in [n_1]$, $j \in [n_2]$ and $0 \leq s \leq k$. We check if $(1,1)$ is in any $P^s_{n_1,n_2}$ and if so, output that $(1,1)$ is in $\mathcal{R}_{\Delta,s}$, i.e., $\mathrm{d}^k_{\mathcal{S}}(B,T) \leq \Delta$, and $\mathrm{d}^k_{\mathcal{S}}(B,T) > \Delta$ otherwise.

6.3.2 Analysis

We now analyze the described algorithm. We begin by showing that the described routines correctly compute points reachable by diagonal and vertical tunnels before analyzing the correctness and running time of the entire algorithm.

Correctness

We argue that the structure of the sets $P^s_{i,j}$ computed by the algorithm is as claimed. That is, for all i, j, and s, it holds that $\mathcal{R}^{(i,j)}_{\Delta,s} = \bigcup_{0\leq s'\leq s} P^{s'}_{i,j}$.

Lemma 6.3.1. *Let T and B be two polygonal curves with n_1 and n_2 edges respectively. For any $i \in [n_1]$, $j \in [n_2]$ and $s \in [k]$ let $D^s_{i,j}$ be the set of endpoints of diagonal tunnels, as computed in the algorithm described in Section 6.3.1, and let $R = \bigcup_{a=1}^{i-1}\bigcup_{b=1}^{j-1} P^{s-1}_{a,b}$ be the set of reachable points by exactly $s-1$ proper tunnels in the lower-left quadrant of $C_{i,j}$. For any $q \in C_{i,j}$ the tunnel $\tau(p,q)$ has price $\mathrm{prc}(\tau(p,q)) \leq \Delta$ for some $p \in R$ if and only if $q \in D^s_{i,j}$.*

Proof. First let a and b be fixed and look at $P = P^{s-1}_{a,b}$. The diagonal tunnel procedure begins by projecting P onto the edge e_b of B, resulting in I. By the correctness of the procedure computing the line-stabbing wedge presented in [GHMS94], the diagonal tunnel procedure computes among other things the set of points in $\mathbb{R}^2$ that are endpoints of stabbers through I and $\mathrm{D}_\Delta(v_{a+1}), \ldots, \mathrm{D}_\Delta(v_i)$ centered at vertices of T. Intersecting this set with e_j results in all endpoints of stabbers through the ordered set ending on e_j. Call this set J. For every point $B(y_q)$ in J there is at least one point $B(y_p)$ in I, such that $\overline{B}[y_p, y_q]$ stabs through $\{\mathrm{D}_\Delta(v_{a+1}), \ldots, \mathrm{D}_\Delta(v_i)\}$. Hence, by Observation 6.2.6, every point $p \in \mathcal{D}^{(a,b)}_\Delta$ with y-coordinate y_p and every point $q \in \mathcal{D}^{(i,j)}_\Delta$ with y-coordinate y_q form a feasible tunnel $\tau(p,q)$ with price at most Δ.

Conversely, since the line-stabbing algorithm correctly computes all possible endpoints of stabbers starting in I and ending on e_j, Observation 6.2.6 implies that any q such that there exists $p \in R$ with $\mathrm{prc}(\tau(p,q)) \leq \Delta$ also must be in $D^s_{i,j}$. As the algorithm iterates over all cells in the lower-left quadrant of $C_{i,j}$ and in the end defines $D^s_{i,j}$ as the union of above computed sets, the claim follows. □

Lemma 6.3.2. *Let T and B be two polygonal curves with n_1 and n_2 edges, respectively. For any $i \in [n_1]$, $j \in [n_2]$ and $s \in [k]$ let $V^s_{i,j}$ be the points reachable by a vertical tunnel as computed in the algorithm and let $R = \bigcup_{b=1}^{j-1} P^{s-1}_{i,b}$ be the set of reachable points by exactly $s-1$ proper tunnels in the column below $C_{i,j}$. For any $q \in C_{i,j}$ the tunnel $\tau(p,q)$ has price $\mathrm{prc}(\tau(p,q)) \leq \Delta$ for some $p \in R$ if and only if $q \in V^s_{i,j}$.*

Proof. Note that any vertical tunnel costs at most Δ if it is feasible by Observation 2.1.1. Let p be the leftmost point in R. Now assume $\tau(r,q)$ is an arbitrary vertical tunnel with $r \in R$ and $q \in C_{i,j}$. Since a tunnel must be monotone, $x_r \leq x_q$. As p is the leftmost point in R, we have $x_p \leq x_r \leq x_q$. As $V^s_{i,j}$ is constructed by intersecting a vertical halfplane rooted at p with $\mathcal{D}^{(i,j)}_\Delta$, it follows that $q \in V^s_{i,j}$. □

Theorem 6.3.3. *Let T and B be two polygonal curves in the plane with overall complexity n together with a value $\Delta > 0$. Let $P^s_{i,j}$ be the set of reachable points with exactly s proper*

tunnels as computed in the algorithm for all i, j and s. It holds that

$$\bigcup_{s' \leq s} P_{i,j}^{s'} = \mathcal{R}_{\Delta,s}^{(i,j)}(B,T).$$

Thus the algorithm correctly decides whether the k-shortcut Fréchet distance of T and B is at most Δ.

Proof. We show that the reachable space $\mathcal{R}_{\Delta,s}^{(i,j)}$ is correctly computed via induction in i, j and s. Note that $\mathcal{R}_{\Delta,s'}^{(1,1)}$ is computed correctly for all $s' \leq k$ since $\mathcal{D}_{\Delta}^{(1,1)}$ is convex and the algorithm checks whether $(0,0) \in \mathcal{D}_{\Delta}^{(1,1)}$. Thus, if $(0,0) \in \mathcal{D}_{\Delta}^{(i,j)}$, $\mathcal{R}_{\Delta,0}^{(1,1)} = \mathcal{D}_{\Delta}^{(1,1)} = P_{1,1}^{0}$ is computed in the first step, by convexity of $\mathcal{D}_{\Delta}^{(1,1)}$, otherwise it is empty. For $s' > 0$ the set $P_{1,1}^{s'}$ is empty since no cell is below or to the left of it. Hence, $\mathcal{R}_{\Delta,s'}^{(1,1)} = \mathcal{R}_{\Delta,0}^{(1,1)}$ is also computed correctly.

By induction all cells $C_{\leq n_1,<j}$ and $C_{<i,j}$ and in particular $C_{i-1,j}$ and $C_{i,j-1}$ have been handled correctly up to round s and hence the set $P_{i,j}^{s}$ is computed correctly. Let some point $q \in \mathcal{R}_{\Delta,s}^{(i,j)}$ be given. By Observation 6.2.4, the point q corresponds to a monotone path with $s' \leq s$ proper tunnels. There are three possible ways via which this point in the parametric space is reachable. The path reaching q could take s' shortcuts to reach $C_{i-1,j}$ or $C_{i,j-1}$, and enter via a monotone path through the boundary into $C_{i,j}$ at some point $a \in \partial C_{i,j}$. As $C_{i-1,j}$ and $C_{i,j-1}$ has been handled correctly for s', the incoming reachability intervals on the boundary has been computed correctly containing a, thus q is also in $P_{i,j}^{s'}$. Alternatively, the path could enter some cell $C_{i,l}$ with $s'-1$ shortcuts and then take a vertical shortcut into $C_{i,j}$ for some $l < j$ to reach q. Lemma 6.3.2 implies that q is in $P_{i,j}^{s'}$. Lastly, the path could take a diagonal shortcut to the cell $C_{i,j}$ to reach q. Lemma 6.3.1 implies that q is in $P_{i,j}^{s'}$.

Now let $q \in P_{i,j}^{s'}$ for $s' \leq s$. Then q is either in (i) $N_{i,j}^{s'}$, (ii) $V_{i,j}^{s'}$, (iii) $D_{i,j}^{s'}$ or (iv) is reachable by a monotone path from some point q' in one of the three preceding cases. Thus it suffices to analyze the first three cases. For (i), observe that the cells $C_{i,j-1}$ and $C_{i-1,j}$ have been handled correctly up to round s', and thus q must also be in $\mathcal{R}_{\Delta,s'}^{(i,j)}$. For (ii) and (iii), Lemma 6.3.2 and Lemma 6.3.1 imply that q must be in $V_{i,j}^{s'}$ or $\mathcal{R}_{\Delta,s'}^{(i,j)}$ respectively. Thus $\mathcal{R}_{\Delta,s}^{(i,j)}(B,T) = \bigcup_{s' \leq s} P_{i,j}^{s'}$. □

Running Time

Lemma 6.3.4. *Let T and B be two polygonal curves in the plane with overall complexity n, together with a distance threshold $\Delta > 0$. The algorithm described in Section 6.3.1 has running time in $\mathcal{O}(kn^{2k+2} \log n)$ and uses $\mathcal{O}(kn^{2k+2})$ space.*

Proof. The sets $N_{i,j}^{s}$, $D_{i,j}^{s}$ and $V_{i,j}^{s}$ computed by the algorithm are described as unions of intersections of $\mathcal{D}_{\Delta}^{(i,j)}$ with halfplanes. For a fixed $P_{i,j}^{s}$ let $n_{i,j,s}$ be the total number of such operations (unions and intersections) from which $P_{i,j}^{s}$ was obtained. As such, $\mathcal{O}(n_{i,j,s})$ bounds the complexity of this set.

The complexity of $N_{i,j}^{s}$ and $V_{i,j}^{s}$ is constant. The complexity of $D_{i,j}^{s}$ is bounded by the sum of the complexities of all cells to the lower-left:

$$n_{i,j,s} \in \mathcal{O}\left(\sum_{a \in [i-1]} \sum_{b \in [j-1]} n_{a,b,s-1} \right).$$

As $i, j \in [n]$, $s \leq k$, and $n_{a,b,0} \in \mathcal{O}(1)$ for all a and b, it holds that $n_{i,j,s} \in \mathcal{O}(n^{2k})$.

Computing $D^s_{i,j}$ takes $\mathcal{O}(\sum_{a<i} \sum_{b<j} n_{a,b,s-1} \log n + n^2 \log n) = \mathcal{O}(n^{2k} \log n)$ time. This follows from the fact that we compute $\mathcal{O}(n)$ line-stabbing wedges, and for every cell $C_{a,b}$ with $a < i$ and $b < j$ we handle $n_{a,b,s-1}$ line segments based on $P^{s-1}_{a,b}$. Computing $N^s_{i,j}$ takes $\mathcal{O}(n_{i-1,j,s} + n_{i,j-1,s}) = \mathcal{O}(n^{2k})$ time, as we need to compute the reachability intervals from neighboring cells. Computing $V^s_{i,j}$ takes $\mathcal{O}(\sum_{b<j} n_{i,b,s-1}) = \mathcal{O}(n^{2k-1})$ time, as we need to compute the leftmost point among all $P^{s-1}_{i,<j}$. Computing $Q(N^s_{i,j} \cup V^s_{i,j} \cup D^s_{i,j})$ takes linear time in the complexity of $N^s_{i,j} \cup V^s_{i,j} \cup D^s_{i,j}$, i.e., $\mathcal{O}(n^{2k})$. As we do this for every cell in every round, the running time overall is $\mathcal{O}(kn^{2k+2} \log n)$.

Finally, the space required to store $P^s_{i,j}$ is in $\mathcal{O}(n^{2k})$ and hence the space is bounded by $\mathcal{O}(kn^{2k+2})$. □

Lemma 6.3.4 together with Theorem 6.3.3 implies Theorem 6.1.1.

Theorem 6.1.1. *Let T and B be two polygonal curves in the plane with overall complexity n, together with a value $\Delta > 0$. There exists an algorithm with running time in $\mathcal{O}\left(kn^{2k+2} \log n\right)$ and space in $\mathcal{O}\left(kn^{2k+2}\right)$ that decides whether $d^k_{\mathcal{S}}(B, T) \leq \Delta$.*

Since for any k there can be at most $n - 1$ proper tunnels, we obtain the following corollary by setting $k = n - 1$.

Corollary 6.3.5. *Let T and B be two polygonal curves in the plane with overall complexity n, together with a value $\Delta > 0$. There exists an algorithm with running time in $\mathcal{O}\left(n^{2n+1} \log n\right)$ and space in $\mathcal{O}\left(n^{2n+1}\right)$ that decides whether the shortcut Fréchet distance of T and B is at most Δ.*

6.4 Hardness

We next explore conditional lower bounds for the problem of deciding whether the k-shortcut Fréchet Distance is at most Δ. More specifically, we reduce the decision problem for the $(4k + 2)$-shortcut Fréchet distance to the k-Table-SUM variant of the k-SUM problem.

Problem 5 (k-Table-SUM)**.** We are given k lists $S_1, \ldots, S_k$ of n non-negative integers $\{s_{i,1}, \ldots, s_{i,n}\}$ and a non-negative integer σ. Decide whether there are indices $\iota_1, \ldots, \iota_k$ such that $\sum_{i=1}^k s_{i,\iota_i} = \sigma$.

The k-Table-SUM problem cannot be solved in $n^{o(k)}$ time. This is well known and can be shown via a reduction from k-SUM. We provide a proof for the sake of completeness.

Problem 6 (k-SUM)**.** We are given a list S of n non-negative integers $\{s_1, \ldots, s_n\}$ and a non-negative integer σ. Decide whether there are k distinct indices $1 \leq \iota_1 < \ldots < \iota_k \leq n$ such that $\sum_{i=1}^k s_{\iota_i} = \sigma$.

Theorem 6.4.1 (Folklore)**.** *Assuming the exponential time hypothesis, for any fixed $k > 0$ the k-Table-SUM problem cannot be solved in $n^{o(k)}$ time.*

Proof. The *exponential time hypothesis* states that the well-known 3-SAT problem in n variables cannot be solved in $2^{o(n)}$ time [IP99]. Assuming the exponential time hypothesis, [PW10] showed that k-SUM cannot be solved in $n^{o(k)}$ time.

We reduce a k-SUM instance to a k-Table-SUM instance. We begin by randomly partitioning the original integer list into k non-empty parts. With probability at least $k!/k^k > e^{-k}$, any given solution is then split, with one item in each of the k lists. This can be derandomized by computing a k-perfect family of hash functions, introduced in [SS90]. A deterministic construction for a suitable k-perfect family of hash functions can be found in [AYZ95] resulting in a family of size $2^{O(k)} \log n$. Thus overall we can solve one k-SUM instance by solving $2^{O(k)} \log n$ instances of the k-Table-SUM problem. This in turn implies that there is at least one k-Table-SUM instance which cannot be solved in $n^{o(k)}/(2^{O(k)} \log n) = n^{o(k)}/2^{O(k)}$ time. There exists a constant c, such that

$$n^{o(k)}/2^{O(k)} \geq n^{o(k)}/c^k = n^{o(k)}/n^{k/\log_c n} \geq n^{o(k)-k/\log_c n} = n^{o(k)},$$

where the last equation holds for any fixed k, and any $n \geq n_0$ if we choose n_0 large enough, concluding the proof. □

Without loss of generality we assume that each table has a minimum entry of value 0. This is equivalent to the above stated k-Table-SUM problem by subtracting the minimum value of each list from every value of that list as well as the sum of all minimum values from the target value σ. We further assume that all k lists are sorted.

6.4.1 General Idea

A k-Table-SUM instance consists of k lists of integers and a target value and asks whether the target value can be rewritten as a sum of values, one from each list. Based on such an instance, we construct a $(4k+2)$-shortcut Fréchet distance instance consisting of the target curve T and the base curve B with the property that they have a distance of 1 if and only if the underlying instance has a solution.

The target curve T lies on a horizontal line going to the right. The set of points in $\mathbb{R}^2$ which have a distance of at most 1 to any point of the target curve we call the *hippodrome*. The base curve consists of several horizontal edges going to the left on the boundary of the hippodrome. All other edges of the base curve lie outside the hippodrome. Any shortcut curve of B that has Fréchet distance at most 1 to T we call *feasible*. Trivially, any feasible shortcut curve must lie completely inside the hippodrome. Since any edge of the base curve that is inside the hippodrome lies on the boundary of the hippodrome and is oriented in the opposite direction of the base curve, no feasible shortcut curve consists of any subcurve of the target curve. Hence, every shortcut on a feasible shortcut curve has to start where the previous shortcut ended.

To restrict the set of feasible shortcut curves even further, we place so called *twists* on the target curve. A twist can only be traversed by a shortcut by going through precisely one point. We call this point the *focal point* or *projection center*. For a simplified structural view of the curves refer to Figure 6.5. These twists are constructed by going a distance of 2 to the left, before continuing rightwards (c.f. 'zig-zags' in Section 5.4.1). We do not place any edges of the base curve too close to twists, so that a shortcut must be taken to traverse every twist.

Intuitively, we can think of the horizontal edges of the base curve as mirrors that disperse incoming light in all directions and focal points as a wall with a hole, similar to that of a pinhole camera. A shortcut curve can be thought of as the path of a photon bouncing from mirror to mirror, always passing through a focal point. A feasible shortcut curve exists if and only if it is possible to send a photon from the beginning of the base curve to the very end.

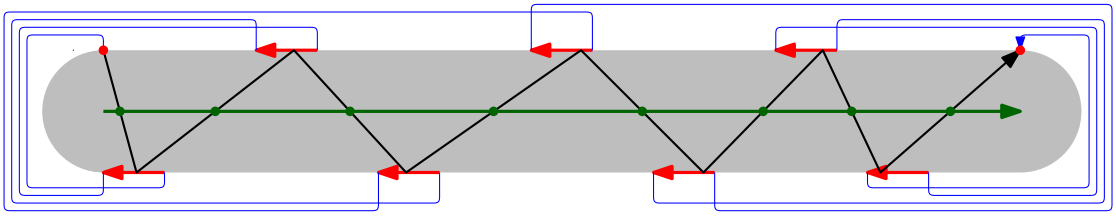


Figure 6.5: Simplified global layout of the target curve and its focal points in green, and the base curve consisting of red mirror edges and blue connector edges. A feasible shortcut curve is drawn in black, hippodrome in gray.

We keep track of the partial sum encoded by any feasible shortcut curve by tracking the position on each mirror edge from which such a photon bounces off of, i.e., where the feasible shortcut curve starts a new shortcut. By the careful placement of mirror edges we ensure that the relative position on each mirror edge where such a photon bounces off of is similar to the relative position of where the photon arrives on the next mirror edge. Our construction presents a shortcut curve with choices of diverging paths, with each diverging path affecting the relative offset along the later mirror edges differently. We place multiple edges, one for each item in a list S_i of the k-Table-SUM instance, at distances between $\frac{1}{2}$ and 1 of the base curve. The shortcut curve has to choose between these edges and this choice encodes which element of S_i is taken.

These can be thought of as semi-transparent mirrors, since a shortcut curve can either end a shortcut on such a mirror edge or pass through it ending on a different mirror edge, akin to a photon either bouncing off of a semi-transparent mirror or traveling through it. Bouncing off of such a semi-transparent mirror corresponds to taking the corresponding item from a list in the k-Table-SUM instance. Since the distance from these edges to the target curve can be less than 1, it can happen that a feasible shortcut curve traverses the edge before taking the next shortcut. Hence, the relative position along an edge no longer encodes precise values but approximates the partial sums. We introduce a scaling in the horizontal direction to contain this error. A second problem that occurs is that edges may overlap in the vertical direction such that photons may visit multiple edges. We address this issue similarly, by scaling.

6.4.2 Construction

In this section we describe the construction of the curves T and B given a k-Table-SUM instance.

First, we describe the overall layout of the instance. The instance consists of $k+2$ basic blocks, which we call gadgets. It consists of an initialization gadget g_0, k encoding-gadgets $g_1, \ldots, g_k$ that encode the individual lists of the k-Table-SUM instance and a terminal gadget g_{k+1} used to verify that the relative offset encoded by a feasible shortcut curve is the same as the target value σ. Each gadget g_i consist of two curves T_i and B_i, which we concatenate to get T and B in the end. We denote by H_y the horizontal line at y in $\mathbb{R}^2$ and by $H_{>y}$ all points above H_y. Similarly for $\geq, \leq$, and $<$. And finally $H_{\geq a}^{<b} = H_{\geq a} \cap H_{<b}$. The target curve T lies in H_0.

The base curve has leftwards oriented horizontal edges in $H_{\geq 1/2}^{\leq 1}$ and $H_{\geq -1}^{\leq -1/2}$ which we call *mirror edges*. All other edges of B_i that connect these mirror edges we call *connector*

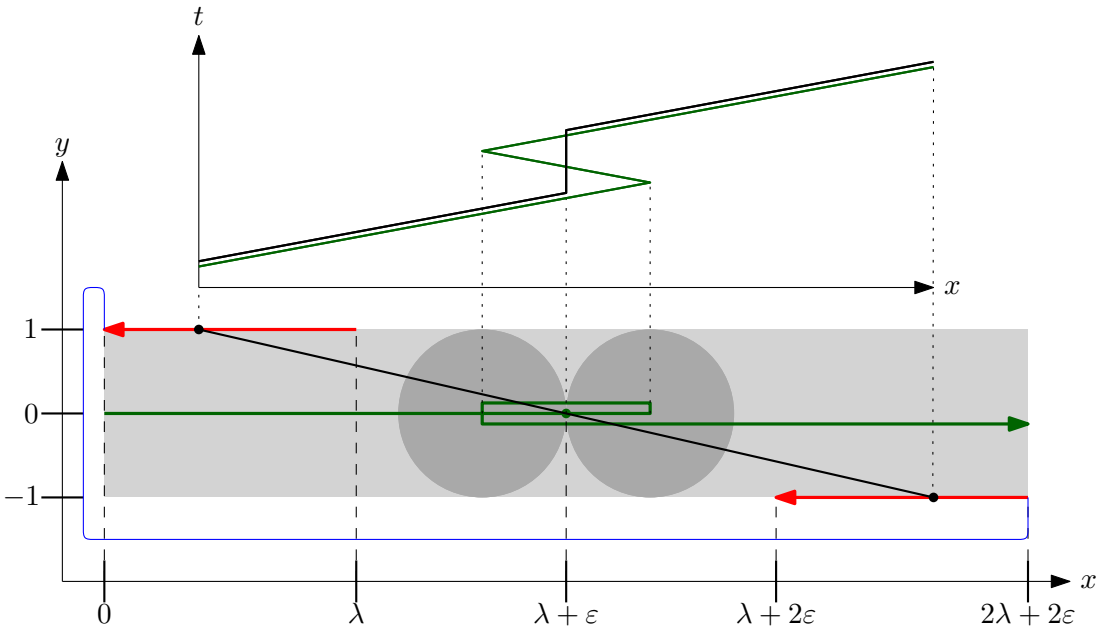


Figure 6.6: Traversal of a shortcut through a twist. The target curve is distorted, to emphasize the structure of the twist.

edges. The connector edges mostly lie outside of the hippodrome. The placement for connector edges that lie outside of the hippodrome is irrelevant. We have to carefully look at any exception, since we want any feasible shortcut curve to only interact with the mirror edges. Since all points lie on a small set of horizontal lines, we may occasionally denote the x-coordinate of a point and the point itself with the same variable but in different fonts. For example the point $\mathbf{x}_j^i$ has x-coordinate x_j^i.

The edges of the target curves T_i are, with the exception of twists, oriented in positive x-direction. A twist centered at the focal point $(p, 0)$ is a subcurve defined by the points $(p-1, 0), (p+1, 0)$, and $(p-1, 0)$ connected by straight lines. Around each focal point we introduce a buffer rectangle of length $2\varepsilon = 5$ and height 3, where we let ε be a global constant for the construction. The base curve never intersects these buffer zones, which is important for the twists to restrict the feasible shortcut curves as intended.

The instance has two more global parameters. The first parameter $\gamma \geq 1$ is a global scaling factor in y-direction, which ensures that feasible shortcut curves never enter connector edges. Furthermore, it ensure that the approximate encoding of two different partial sums stays disjoint. The parameter γ will be in $\mathcal{O}(k)$. Lastly, β is a spacing parameter ensuring that edges are far enough apart from one another.

Before presenting the precise construction, we first provide justification for the correctness of the twists; see Figure 6.6 for details. Assume we have two mirror edges of length λ, one placed from $(\lambda, 1)$ to $(0, 1)$, the other from $(2\lambda + 2\varepsilon, -1)$ to $(\lambda + 2\varepsilon, -1)$, which are connected by connector edges. We have a twist centered at $(\lambda + \varepsilon, 0)$ on an otherwise rightwards oriented target curve. Assume furthermore that we have a partial feasible shortcut curve, which reaches some point $(p, 1)$ on the first mirror edge. Since the distance to the target curve is precisely 1, any reparametrization with a distance at most 1 for the shortcut Fréchet distance matches the point $(p, 1)$ of B to the point $(p, 0)$ of T. Since the target curve is oriented in the opposite direction to the mirror edge, the only

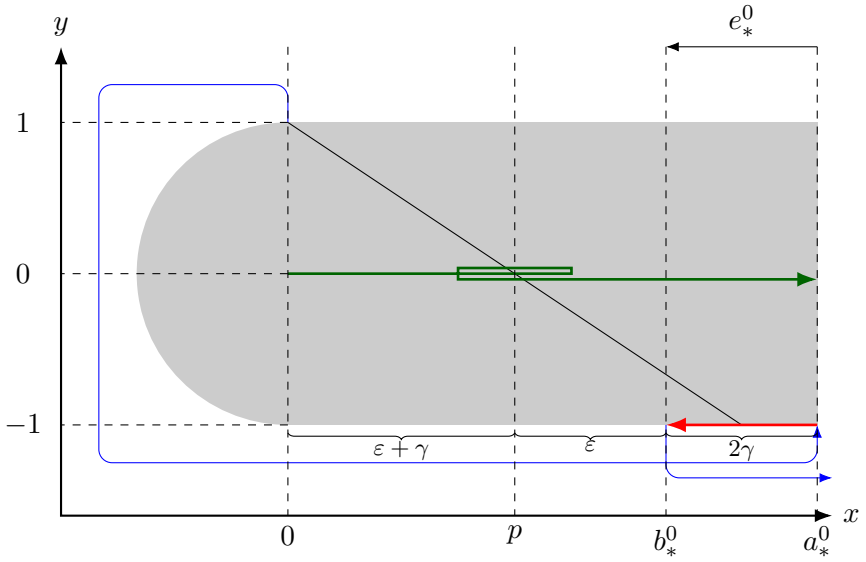


Figure 6.7: Construction of the Initialization gadget. The first forced shortcut is drawn in black. Mirror edges are red, connector edges are blue, and the target curve is green.

way to continue the feasible shortcut curve is by a shortcut to the right. It cannot jump to any point on the first mirror edge, since all points who come after $(p, 1)$ (w.r.t. the curve B) lie left of $(p, 1)$. The shortcut has to traverse the buffer zone of the twist. And since there are no edges of the base curve in the buffer zone, the shortcut has to traverse it completely. To analyze all shortcuts at a distance of at most 1, we place two auxiliary disks centered at $(\lambda + \varepsilon \pm 1, 0)$ of radius 1. Any feasible shortcut curve traversing the buffer zone must traverse both of these disks, since otherwise no reparametrization can pair to the points $(\lambda + \varepsilon \pm 1, 0)$ at distance at most 1, which are part of the target curve.

Since the twist first goes to $(\lambda + \varepsilon + 1, 0)$ and then to $(\lambda + \varepsilon - 1, 0)$, any feasible shortcut curve must also traverse the disks in this order. The first disk lies to the right of the second disk, and we try to traverse these disks from the left. The only possible way to traverse them with a straight line is through the intersection of the disks. And the only point in the intersection is exactly the focal point. So any shortcut of a feasible shortcut curve that traverses the buffer zone of a twist must traverse its focal point. A possible partial traversal is given in the upper plot in Figure 6.6. Note that it is in t-x-space, corresponding to how the two points paired by the reparametrizations traverse the curves in the x-direction.

Initialization Gadget g_0

For the construction refer to Figure 6.7. Both curves T_0 and B_0 start at x-coordinate 0 placing the start point for the base curve at $(0, 1)$, and the start point for the target curve at $(0, 0)$. The target curve goes rightwards, up to the first twist centered at $(\varepsilon+\gamma, 0)$ and continue rightwards after that. The base curve immediately leaves the hippodrome to the left and connects to the first mirror edge from $\mathbf{a}_*^0 = (3\gamma + 2\varepsilon, -1)$ to $\mathbf{b}_*^0 = (\gamma + 2\varepsilon, -1)$.

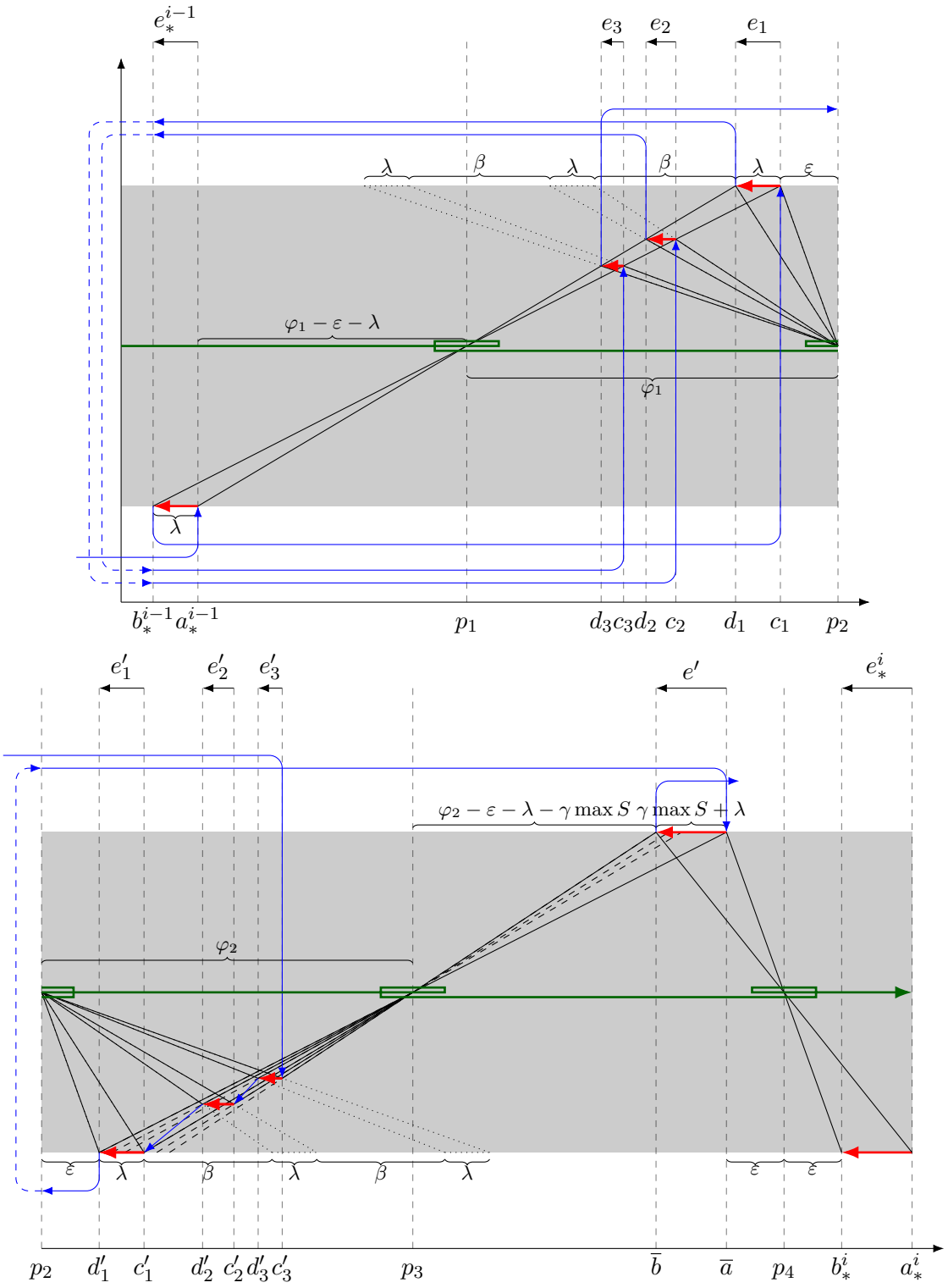


Figure 6.8: Construction of the encoding gadget. Mirror edges are red, connector edges blue and the target curve is green. Projection cones are black.

Step 1:	$\mathbf{p}_1 = \mathbf{a}_*^{i-1} + (\varphi_1 - \varepsilon - \lambda, 1)$	
	$\mathbf{d}_1 = L(\mathbf{a}_*^{i-1}, \mathbf{p}_1) \cap H_1$	for $L(x, y)$ the supporting line of $\overline{x\,y}$
	$\mathbf{c}_1 = L(\mathbf{b}_*^{i-1}, \mathbf{p}_1) \cap H_1$	
Step 2:	$\mathbf{p}_2 = \mathbf{p}_1 + (\varphi_1, 0)$	
	$\mathbf{d}_j = L(\mathbf{a}_*^{i-1}, \mathbf{p}_1) \cap L(\mathbf{d}_1 - \mathbf{s}_j, \mathbf{p}_2)$	with $\mathbf{s}_j = ((j-1)(\beta + \lambda), 0)$
	$\mathbf{c}_j = L(\mathbf{b}, \mathbf{p}_1) \cap L(\mathbf{c}_1 - s_j, \mathbf{p}_1)$	
Step 3:	$\mathbf{p}_3 = \mathbf{p}_2 + (\varphi_2, 0)$	
	$\mathbf{c}'_1 = L(\mathbf{d}_1, \mathbf{p}_2) \cap H_{-1}$	
	$\mathbf{d}'_1 = L(\mathbf{c}_1, \mathbf{p}_2) \cap H_{-1}$	
	$\overline{\mathbf{b}} = (L(\mathbf{c}'_1, \mathbf{p}_3) \cap H_1) - (\gamma \max S_i, 0)$	
	$\overline{\mathbf{a}} = L(\mathbf{d}'_1, \mathbf{p}_3) \cap H_1$	
	$\mathbf{c}'_j = L(\mathbf{d}_j, \mathbf{p}_2) \cap L(\overline{\mathbf{b}}_j, \mathbf{p}_3)$	with $\overline{\mathbf{b}}_j = \overline{\mathbf{b}} + \gamma((\max S_i) - s_{i,j}, 0)$
	$\mathbf{d}'_j = L(\mathbf{c}_j, \mathbf{p}_2) \cap L(\overline{\mathbf{a}}_j, \mathbf{p}_3)$	with $\overline{\mathbf{a}}_j = \overline{\mathbf{a}} - \gamma(s_{i,j}, 0)$
Step 4:	$\mathbf{p}_4 = \mathbf{p}_3 + (\varphi_2, 0)$	
	$\mathbf{b}_*^i = L(\overline{\mathbf{a}}, \mathbf{p}_4) \cap H_{-1}$	
	$\mathbf{a}_*^i = L(\overline{\mathbf{b}}, \mathbf{p}_4) \cap H_{-1}$	

Table 6.1: Precise construction of the i^{th} encoding gadget. Index i is omitted in most cases.

Encoding Gadget g_i

The overall structure of a gadget g_i for some $i \in [k]$ is depicted in Figure 6.8. This gadget encodes the i^{th} table $S_i = \{s_1^i, \ldots, s_n^i\}$ of the k-Table-SUM instance. Table 6.1 shows the construction of the precise values. As for the parameters, λ^i is the length of the entry edge, determined by the previous gadget g_{i-1}, and β is the global spacing parameter. We do not specify φ_1^i and φ_2^i explicitly but show that choosing them to be in $\Theta(\text{poly}(n, \lambda^i, \beta, \varepsilon, \max S_i))$ ensures in a correct reduction. Excluding the entry edge, the base curve B_i consists of $2n + 2$ mirror edges and $\mathcal{O}(n)$ connector edges. For $j \in [n]$ the first n mirror edges e_j^i are defined by $\mathbf{c}_j^i$ and $\mathbf{d}_j^i$, and the second n mirror edges $e_j'^i$ are defined by $\mathbf{c}_j'^i$ and $\mathbf{d}_j'^i$. The last two mirror edges are defined by $\bar{\mathbf{a}}^i$ and $\bar{\mathbf{b}}^i$, and $\mathbf{a}_*^i$ and $\mathbf{b}_*^i$. All of these mirror edges lie in either $H_{\geq 1/2}^{\leq 1}$ or $H_{\geq -1}^{\leq -1/2}$ by construction. The target curve T_i has four twists centered at $\mathbf{p}_1^i, \ldots, \mathbf{p}_4^i$. Since the index i does not change other than for the entry and exit edge, we omit these indices in the construction of this gadget.

The intuition behind the construction is as follows: The first two steps place the first projection center at a distance from the entry edge, such that n copies of the entry edge fit into its *projection cone*. The projection cone refers to the cone formed by projecting the edge through a projection center. The edges must satisfy further constraints, namely that all of the edges lie in $H_{\geq 1/2}^{\leq 1}$ and they have sufficient distance in x-direction.

These n edges correspond to the choice, which item from list S_i should be added to the partial solution encoded by a shortcut curve. Step 3 places an edge e' from $\bar{\mathbf{a}}$ to $\bar{\mathbf{b}}$, where all the diverging paths have to meet, and then places n copies of the entry edge in the n disjoint projection cones such that their projections onto e' have a relative offset according to the values in the list. Step 4 defines the entry edge to the next gadget. The edges in Step 3 and 4 are used to recombine the diverging paths making sure that the offset between the paths corresponds to the value of the items in the list S_i.

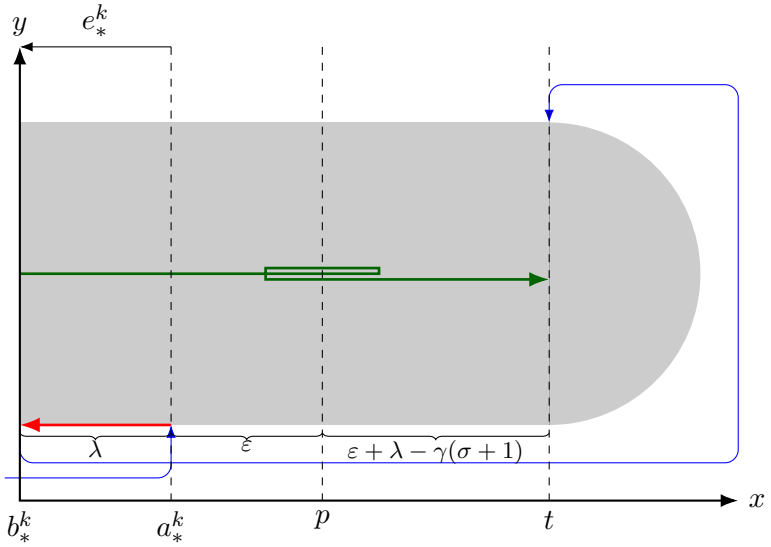


Figure 6.9: Construction of the Terminal-Gadget. Mirror edges are red, connector edges are blue and the target curve is green.

A shortcut curve traversing this gadget looks as follows: A shortcut curve reaches some point in the entry edge e_*^{i-1}. From here it takes a shortcut to some e_j^i. The next shortcuts are forced to land on $e_j'^i$, then e'^i and finally e_*^i.

Terminal Gadget g_{k+1}

The terminal gadget g_{k+1} is the dual to the initialization gadget (refer to Figure 6.9). The entry edge from $(b_*^k + \lambda^k, -1)$ to $(b_*^k, -1)$ is defined by the previous gadget. The target curve T_{k+1} has a single twist at $(b_*^k + \lambda + \varepsilon, 0)$ and ends at $(b_*^k + 2\lambda + 2\varepsilon - \gamma(\sigma+1), 0)$. The base curve B_{k+1} connects the entry edge to $(b_*^k + 2\lambda + 2\varepsilon - \gamma(\sigma+1), 1)$ from outside the hippodrome. The final vertex $B(1)$ of the base curve is placed such that a shortcut from the entry edge e_*^k has to start precisely at x-coordinate $b_*^k + \gamma(\sigma + 1)$ in order to pass through the projection center $(b_*^k + \lambda + \varepsilon, 0)$ and end at $B(1)$, where σ is the target value from the k-Table-SUM instance.

6.4.3 Correctness for One-Touch Shortcut Curves

We now argue that this construction is correct. That is, there exists a feasible shortcut curve with $(4k + 2)$ shortcuts if and only if the original k-Table-SUM instance has a solution. We begin by showing this for a subset of shortcut curves we call *one-touch*. For general shortcut curves this will be shown in Section 6.4.5. These one-touch shortcut curves consist of only shortcuts and never take subcurves of the base curve B. In the following section we often have to argue with distances that get preserved when getting projected through a projection center. This argument is captured in the following observation.

Observation 6.4.2. *If an edge lies on some $H_{-\beta}$ with length λ, and some point p on H_0 is given, we can then project the edge through p onto some H_α of length λ'. This forms two congruent triangles such that $\lambda' = \frac{\alpha\lambda}{\beta}$. See e_*^{i-1} and e_3 in Figure 6.8 for an example.*

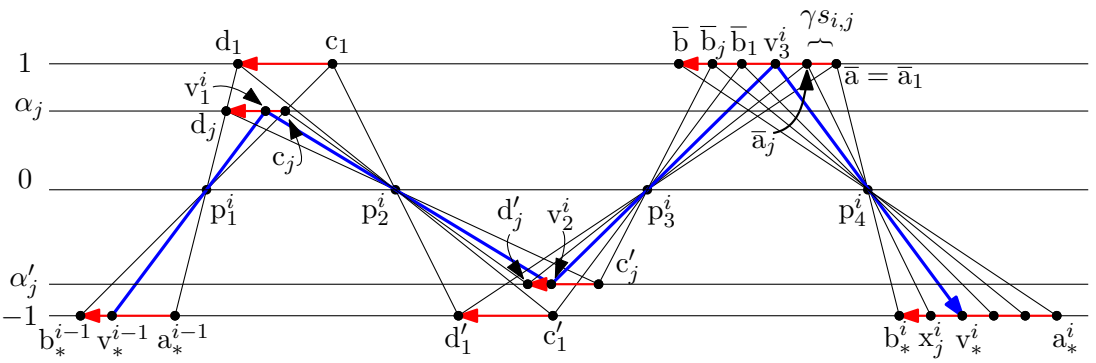


Figure 6.10: The path of a shortcut curve through the gadget g_i in the case, where $s_{i,j}$ is selected from the i^{th} list (Lemma 6.4.5). Most top indices i are omitted. For presentation purposes, the mirror edges have horizontal overlap. In the construction they do not.

Definition 6.4.3 (One-Touch Encoding)**.** Let $I = \iota_1, \ldots, \iota_k$ be an index set of a k-Table-SUM instance. We construct a one-touch shortcut curve B_I of the base curve incrementally. The first two vertices on the initial gadget are defined as follows. We choose the first vertex of the base curve $B(0)$ for $\mathbf{v}_0^0$, then we project it through the first projection center $\mathbf{p}^0$ onto e_*^0 to obtain $\mathbf{v}_*^0$. Now for $i \in [k]$ we project $\mathbf{v}_*^{i-1}$ through $\mathbf{p}_1^i$ to land on $e_{\iota_i}^i$ to obtain $\mathbf{v}_1^i$. We continue by projecting $\mathbf{v}_l^i$ through $\mathbf{p}_l + 1$ onto B_i to obtain $\mathbf{v}_{l+1}^i$ for $1 \leq l \leq 3$ (refer to Figure 6.10). Since these projections are all forced, no choices have to be made. Let $\mathbf{v}_*^i = \mathbf{v}_4^i$. We continue this construction throughout all gadgets in order of i. Finally, we choose $B(1)$ as the last vertex of our shortcut curve.

Lemma 6.4.4. *For any $i \in [k]$ and $j \in [n]$ let $\mathbf{x}_j^i$ be the leftmost point on e_*^i reachable by projections starting on edge e_j^i. Then $x_j^i - b_*^i = \gamma s_{i,j}$.*

Proof. This follows directly from the construction (refer to Figure 6.10 and Table 6.1) and repeated application of Observation 6.4.2. The value x_j is determined by the projection of $\mathbf{c}_j$ through $\mathbf{p}_2$, which is $\mathbf{d}'_j$. Projecting this through $\mathbf{p}_3$ lands on $\bar{\mathbf{a}}_j$ which by another projection through $\mathbf{p}_4$ lands on $\mathbf{x}_j$. The offset between $\bar{\mathbf{a}}_j$ and $\bar{\mathbf{a}}$ is precisely the offset between $\mathbf{x}_j$ and $\mathbf{b}'$. And this offset is by construction $\gamma s_{i,j}$. □

Lemma 6.4.5. *Given a shortcut curve B_I, which is a one-touch encoding, let $\mathbf{v}_*^i$ be the vertex of B_I on the entry edges e_*^i of gadgets g_i for all $0 \leq i \leq k$. Then $||\mathbf{v}_*^i - \mathbf{b}_*^i|| = \gamma(\sigma_i + 1)$, where σ_i is the i^{th} partial sum of the values s_{i,ι_i} with ι_i in the index set I encoded by B_I.*

Proof. We prove this via induction. For $i = 1$ this is correct by construction of the initialization gadget. Refer for the following argument to Figure 6.10 and Observation 6.4.2. For all choices of j we have $||\mathbf{v}_*^{i-1} - \mathbf{b}_*^{i-1}|| = ||\mathbf{v}_*^i - \mathbf{x}_j^i||$. This follows immediately from following the projections:

$$||\mathbf{v}_*^{i-1} - \mathbf{b}_*^{i-1}|| = \alpha_j||\mathbf{v}_1^i - \mathbf{c}_j|| = \alpha'_j||\mathbf{v}_2^i - \mathbf{d}'_j|| = ||\bar{\mathbf{v}}_3^i - \bar{\mathbf{a}}_j|| = ||\mathbf{v}_*^i - \mathbf{x}_j^i||.$$

Together with Lemma 6.4.4 we have

$$||\mathbf{v}_*^i - \mathbf{b}_*^i|| = ||\mathbf{v}_*^i - \mathbf{x}_j^i|| + ||\mathbf{x}_j^i - \mathbf{b}_*^i|| = \gamma(\sigma_{i-1} + 1) + \gamma s_{i,j} = \gamma(\sigma_i + 1).$$ □

Lemma 6.4.6. *Let $o_j^i = (j-1)(\lambda^{i-1}+\beta)$. Then $\mathbf{c}_j^i$, $\mathbf{d}_j^i$, $\mathbf{c'}_j^i$ and $\mathbf{d'}_j^i$ are given by*

$$\mathbf{c}_j^i = \mathbf{p}_1^i + \left(\frac{(\varphi_1^i-\varepsilon)\varphi_1^i}{\varphi_1^i+o_j^i}, \frac{\varphi_1^i}{\varphi_1^i+o_j^i}\right),$$
$$\mathbf{d}_j^i = \mathbf{p}_1^i + \left(\frac{(\varphi_1^i-\varepsilon-\lambda)\varphi_1^i}{\varphi_1^i+o_j^i}, \frac{\varphi_1^i}{\varphi_1^i+o_j^i}\right),$$
$$\mathbf{c'}_j^i = \mathbf{p}_2^i + \left(\frac{(\varepsilon+o_j^i+\lambda)\varphi_2^i}{\varphi_2^i-\gamma s_{i,j}+o_j^i}, \frac{-\varphi_2^i}{\varphi_2^i-\gamma s_{i,j}+o_j^i}\right), \text{ and}$$
$$\mathbf{d'}_j^i = \mathbf{p}_2^i + \left(\frac{(\varepsilon+o_j^i)\varphi_2^i}{\varphi_2^i-\gamma s_{i,j}+o_j^i}, \frac{-\varphi_2^i}{\varphi_2^i-\gamma s_{i,j}+o_j^i}\right).$$

Proof. We will only show this for $\mathbf{d}_j^i$, as the computations for the other points are very similar. We translate the instance such that $\mathbf{p}_1^i$ coincides with the origin. Then $\mathbf{d}_j^i$ is defined as the intersection of a line l_1 from $(0,0)$ to $(\varphi_1^i-\varepsilon-\lambda^{i-1},1)$ and l_2 from $(\varphi_1^i-\varepsilon-\lambda^{i-1}-(j-1)(\lambda^{i-1}+\beta),1)$ to $(\varphi_1^i,0)$. Thus the x-coordinate of the intersection point satisifies

$$\frac{d_j^i}{\varphi_1^i-\varepsilon-\lambda^{i-1}} = 1 - \frac{d_j^i-(\varphi_1^i-\varepsilon-j\lambda^{i-1}-(j-1)\beta)}{\varepsilon+j\lambda^{i-1}+(j-1)\beta}$$

and hence

$$d_j^i = \frac{(\varphi_1^i-\varepsilon-\lambda^{i-1})\varphi_1^i}{\varphi_1^i+(j-1)\lambda^{i-1}+(j-1)\beta}.$$

Since l_1 has slope $\left(\varphi_1^i-\varepsilon-\lambda^{i-1}\right)^{-1}$ we have

$$\mathbf{d}_j^i = \frac{1}{\varphi_1^i+(j-1)(\lambda^{i-1}+\beta)}\left((\varphi_1^i-\varepsilon-\lambda^{i-1})\varphi_1^i, \varphi_1^i\right). \qquad \square$$

Lemma 6.4.7. *Assume φ_1^i and φ_2^i are given such that $\varphi_1^i \geq 3\varepsilon+\lambda^{i-1}$ and $\varphi_2^i \geq 2\varepsilon+\gamma s_{i,j}+(j-1)(\lambda^{i-1}+\beta)+2\lambda^{i-1}$ holds for every $i\in[k]$ and $j\in[n]$. Then the constructed base curve never enters any buffer zone centered at a projection center.*

Proof. We will not explicitly discuss the connector edges outside the hippodrome, since they can easily be placed such that they do not enter buffer zones. We first consider the encoding gadget g_i. In this proof we omit the top index $i-1$ from λ^{i-1} and top index i from all other variables, as we only look at a single gadget at a time. For the buffer zones centered at $\mathbf{p}_2$ and $\mathbf{p}_4$ for $i\in[k]$ the claim is implied by construction.

For the buffer zone centered at $\mathbf{p}_1$ the closest edge in x-direction inside the hippodrome is by construction e_n. And for this edge, $\mathbf{d}_n$ is the closest point. Lemma 6.4.6 and the fact that $\varphi_1 \geq 2\varepsilon+\lambda+\varepsilon$ holds, imply that $d_n \geq p_1 + \frac{\varphi_1-\lambda-\varepsilon}{2} > p_1+\varepsilon$ holds. This implies the claim for the projection center $\mathbf{p}_1$ as well.

For the buffer zone centered at $\mathbf{p}_3$ the closest edge in x-direction inside the hippodrome is by construction e'_n. And for this edge, $\mathbf{c}'_n$ is the closest point. Lemma 6.4.6 together with the fact that $\varphi_2 \geq 2\varepsilon+s_n+o_j+2\lambda$ holds, imply that we get $c'_n \leq p_2+\varphi_2/2$ and thus $p_3 \geq c'_n+\varphi_2/2 \geq c'_n+\varepsilon$. The last two buffer zones left to analyze are those centered at the projection center in the initialization and end gadget. For these the claim follows immediately from construction. $\square$

Lemma 6.4.8 (4-monotonicity [BDS14])**.** *Any feasible shortcut curve is rightwards 4-monotone. That is, if x_1 and x_2 are the x-coordinates of two points that appear on the shortcut curve in that order, then $x_2 + 4 \geq x_1$. Furthermore, it lies inside or on the boundary of the hippodrome.*

Proof. Any point on the feasible shortcut curve has to lie within distance 1 to some point of the target curve, thus the curve cannot leave the hippodrome. As for the monotonicity, assume for the sake of contradiction that there exist two such points with $x_2 + 4 < x_1$. Let $\hat{x}_1$ be the x-coordinate of the point on the target curve matched to x_1 and let $\hat{x}_2$ be the one for x_2. By the Fréchet matching it follows that $\hat{x}_2 - 1 + 4 < \hat{x}_1 + 1$. This would imply that the target curve is not 2-monotone which contradicts the way we constructed it. □

Lemma 6.4.9. *For every $\lambda > 0$, $\beta \geq 5$, $\varepsilon > 0$ and integer $n > 0$ there are values $\varphi_1^i, \varphi_2^i \in \Theta(\mathrm{poly}(\varepsilon, \lambda, (\beta-4)^{-1}, \beta, \gamma \max S_i, n))$ for every $i \in [n]$ such that any two mirror edges of the gadget g_i are at least 4 apart and all mirror edges lie inside the hippodrome and in either $H_{\geq 1/2}^{\leq 1}$ or $H_{\geq -1}^{\leq -1/2}$.*

Proof. We first consider the encoding gadget g_i. For this we omit the top index $i-1$ from λ^{i-1} and top index i from all other variables as we only look at a single gadget at a time. Recall from Lemma 6.4.6

$$\mathbf{d}_j = \frac{1}{\varphi_1 + (j-1)(\lambda+\beta)}\Big((\varphi_1 - \varepsilon - \lambda)\varphi_1, \varphi_1\Big)$$

and

$$\mathbf{c}_j = \frac{1}{\varphi_1 + (j-1)(\lambda+\beta)}\Big((\varphi_1 - \varepsilon)\varphi_1, \varphi_1\Big).$$

Hence for any $\varphi_1 \geq (n-1)(\lambda+\beta)$ the y-coordinate of any such edge lies in $[1/2, 1]$. Next we show that $c_{j+1} + 4 < d_j$ holds, implying that the edges $e_1, \ldots, e_n$ have a pairwise distance of at least 4. The expression $c_{j+1} + 4 < d_j$ is equivalent to

$$\lambda\varphi_1 j(\lambda+\beta) + 4\Big(\varphi_1 + j(\lambda+\beta)\Big)\Big(\varphi_1 + (j-1)(\lambda+\beta)\Big) + (\lambda+\beta)\varphi_1\varepsilon < \beta\varphi_1^2.$$

As both sides are second degree polynomials in φ_1 and the second order coefficient on the left hand side is 4 whereas on the right hand side it is β (recall $\beta \geq 5$), there is a value $\Phi_1 \in \mathcal{O}(\mathrm{poly}(\lambda, (\beta-4)^{-1}, \beta, \varepsilon, n))$ such that for every $\varphi_1 > \Phi_1$ the expression $c_{j+1} + 4 < d_j$ holds for all $j \in [n-1]$.

Let us next look at the edges e'_j. Similarly recall from Lemma 6.4.6 that

$$\mathbf{c}'_j = \frac{1}{\varphi_2 - \gamma s_j + (j-1)(\lambda+\beta)}\Big((\varepsilon + j\lambda + (j-1)\beta)\varphi_2, -\varphi_2\Big)$$

and

$$\mathbf{d}'_j = \frac{1}{\varphi_2 - \gamma s_j + (j-1)(\lambda+\beta)}\Big((\varepsilon + (j-1)\lambda + (j-1)\beta)\varphi_2, -\varphi_2\Big).$$

Hence there is a $\Phi_2 \in \mathcal{O}(\mathrm{poly}(\max_i s_i, n, \lambda, \beta))$ such that for any $\varphi_2 > \Phi_2$ the y-coordinate of any such edge lies in $[-1, -1/2]$. Lastly, the expression $c'_j + 4 < d_{j+1}$ is equivalent to

$$\begin{aligned}
&\Big(\varepsilon + j\lambda + (j-1)\beta\Big)\Big(\gamma(s_j - s_{j+1}) + \lambda + \beta\Big)\varphi_2 \\
&\quad + 4\Big(\varphi_2 - \gamma s_j + (j-1)(\lambda+\beta)\Big)\Big(\varphi_2 - \gamma s_{j+1} + j(\lambda+\beta)\Big) \\
&< \beta\varphi_2\Big(\varphi_2 - \gamma s_j + (j-1)(\lambda+\beta)\Big).
\end{aligned}$$

We again have two second order polynomials in φ_2 on both sides, with the second order coefficient on the left hand side being 4 and on the right hand side being $\beta \geq 5$. Hence there is a $\Phi_2' \in \mathcal{O}(\text{poly}(\max_j s_j, n, \lambda, (\beta-4)^{-1}, \beta, \varepsilon))$ such that all $\varphi_2 > \Phi_2'$ satisfy this equation.

Thus we can chose $\varphi_1, \varphi_2 \in \Theta(\text{poly}(\varepsilon, \lambda, (\beta-4)^{-1}, \beta, \max_j s_j, n))$ such that $\varphi_1 > \max((n-1)(\lambda+\beta), \Phi_1)$ and $\varphi_2 > \max(\Phi_2, \Phi_2')$. Thus all edges $e_1, \ldots, e_n$ have a pairwise distance of at least 4, and all edges $e_1', \ldots, e_n'$ have a pairwise distance of at least 4. Further they all lie within the hippodrome and lie in either $H_{\geq 1/2}^{\leq 1}$ or $H_{\geq -1}^{\leq -1/2}$. As all other mirror edges are separated by buffer zones and their position is trivially inside the hippodrome and at distance at least $1/2$ to B, this concludes the proof. □

Observation 6.4.10. *If $\beta \geq 5$ and $\varepsilon > 2$ holds, we can choose values φ_1^i and φ_2^i in $\Theta(\text{poly}(\varepsilon, \lambda^{i-1}, (\beta-4)^{-1}, \beta, \gamma \max_j s_{i,j}, n))$ for all $i \in [k]$ such that the conditions of both Lemma 6.4.7 and Lemma 6.4.9 hold.*

Lemma 6.4.11. *If $\varepsilon > 2$, then a feasible shortcut curve passes through every buffer zone of the target curve via its projection center and furthermore it does so from left to right.*

Proof. Any feasible shortcut curve has to start at $B(0)$ and end at $B(1)$, and all of its vertices must lie in the hippodrome or on its boundary. By Lemma 6.4.7, the base curve does not enter any of the buffer zones and therefore the feasible shortcut curve has to pass through the buffer zone by using a shortcut. If we choose the width of a buffer zone $2\varepsilon > 4$, then the only way to do this while matching to the two associated vertices of the target curve in their respective order is to go through the intersection of their unit disks. The intersection lies at the center of the buffer zone, as we saw in the above. □

Lemma 6.4.12. *If $\varepsilon > 2$, $\beta \geq 5$ and all φ_1^i and φ_2^i are chosen according to Observation 6.4.10, then a feasible shortcut curve that is one-touch visits exactly one of the edges e_j^i and exactly one of the edges $e_j'^i$ for some $j \in [n]$ in every gadget g_i for $i \in [k]$. Furthermore, it visits all edges e_*^i for $0 \leq i \leq k$.*

Proof. By Lemma 6.4.8, any feasible shortcut curve is 4-monotone. Furthermore, it starts at $B(0)$ and ends at $B(1)$. By Lemma 6.4.11, it goes through all projection centers of the target curve from left to right. We first want to argue that it visits at least one mirror edge between two projection centers, i.e., that it cannot 'skip' such a mirror edge by matching to two twists in one shortcut. Such a shortcut would have to lie on H_0, since it has to go through the two corresponding projection centers lying on H_0. By construction the only possible endpoints of such a shortcut lie on the connector edges that connect to mirror edges. Assume such a shortcut could be taken by a shortcut curve starting from $B(0)$. Then there must be a connector edge which intersects a line from a point on a mirror edge through the projection center. In particular since the curve has to go through all projection centers, one or more of the following must be true for some $i \in [k]$:

- there exists a line through $\mathbf{p}_1^i$ intersecting a mirror edge e_*^{i-1} and a connector edge of e_j^i,
- there exists a line through $\mathbf{p}_2^i$ intersecting a mirror edge e_j^i and a connector edge of $e_l'^i$ for some l, or
- there exists a line through $\mathbf{p}_3^i$ intersecting a mirror edge $e_j'^i$ and a connector edge of $e_l'^i$ for some $l > j$.

However, this was prevented by the careful placement of these connector edges.

It remains to prove that for each $i \in [k]$ the shortcut curve cannot visit more than one e_j^i and cannot visit more than one $e_j'^i$ and therefore visits exactly one mirror edge between two projection centers. The shortcut curve has to lie inside or on the boundary of the hippodrome and is 4-monotone (Lemma 6.4.8). At the same time, we constructed the gadget such that the mirror edges between two consecutive projection centers have distance at least 4 to one another by Lemma 6.4.9, choosing φ_1^i and φ_2^i in the process. Furthermore, inside the projection cone from $e_j'^i$ to $\mathbf{p}_3^i$ all mirror edges come before (as parametrized by the base curve) $e_j'^i$, implying the claim. □

To summarize, we may think of a one-touch shortcut curve as a one-touch encoding, as any one-touch shortcut curve visits exactly one of the edges e_j^i. Let $e_{\iota_i}^i$ be this edge for gadget g_i. This defines the one-touch encoding $I = \{\iota_1, \ldots, \iota_k\}$.

Putting the above lemmas together implies the correctness of the reduction for shortcut curves that are one-touch.

Lemma 6.4.13. *If $\varepsilon > 2$ and $\beta \geq 5$ and all φ_1^i and φ_2^i are chosen according to Observation 6.4.10, then for any feasible one-touch shortcut curve $B_\diamond$ it holds that the index set I encoded by $B_\diamond$ sums to σ. Furthermore, for any index set I that solves the k-Table-SUM instance there is a feasible one-touch shortcut curve that encodes it.*

Proof. Lemma 6.4.11 and Lemma 6.4.12 imply that $B_\diamond$ must be a one-touch encoding as defined in Definition 6.4.3. By Lemma 6.4.5, the penultimate vertex of $B_\diamond$ is the point on the edge e_*^k which is at distance $\gamma(\sigma_\diamond + 1)$ to $\mathbf{b}_*^k$ where $\sigma_\diamond$ is the sum encoded by the values selected by $B_\diamond$. The last vertex of $B_\diamond$ is equal to $B(1)$, which we placed in distance $\gamma(\sigma + 1)$ to the projection of $\mathbf{b}_*^k$ through $\mathbf{p}_1^{k+1}$. Thus the last shortcut of $B_\diamond$ passes through the last projection center of the target curve if and only if $\sigma_\diamond = \sigma$. It follows that if $\sigma_\diamond \neq \sigma$, then $B_\diamond$ cannot be feasible.

For the second part of the claim we construct a one-touch encoding as defined in Definition 6.4.3. By the above analysis it will be feasible if the selected values sum to σ, since the curve visits every edge of B in at most one point and in between uses shortcuts which pass through every buffer zone from left to right and via the buffer zones projection center. □

6.4.4 Size of Coordinates

Lemma 6.4.14. *The curves can be constructed in $\mathcal{O}(kn)$ time. Furthermore, if we choose $\varepsilon = \frac{5}{2}$ and $\beta = 5$ and φ_1^i and φ_2^i according to Observation 6.4.10, then the coordinates used are in $\mathcal{O}(\mathrm{poly}(k, n, \gamma \sum_i \max S_i))$.*

Proof. From the construction we know that $p_4^i - p_4^{i-1} = 2\varphi_1^i + 2\varphi_2^i$. Further by Observation 6.4.10 we know that we can choose φ_1^i and φ_2^i to be in $\Theta(\mathrm{poly}(\varepsilon, n, \lambda^{i-1}, \gamma \max S_i, (\beta - 4)^{-1}, \beta)) = \Theta(\mathrm{poly}(n, \lambda^{i-1}, \gamma \max S_i))$ as $\varepsilon = \frac{5}{2}$ and $\beta = 5$.

The length of the entire instance is given by the combined lengths of all of the gadgets. Thus the length is given by

$$\varepsilon + \gamma + \left(\sum_{i=1}^{n} \left(2\varphi_1^i + 2\varphi_2^i \right) \right) + \varepsilon + \lambda^n + \left(\varepsilon + \lambda^n - \gamma(\sigma + 1) \right).$$

This is in $\mathcal{O}(\mathrm{poly}(k, n, \gamma \sum_i \max S_i))$, as $\lambda^0 = 2\gamma$ and $\lambda^i = \lambda^{i-1} + \gamma \max S_i$.

As for the complexity, each of the constructed gadgets uses $\mathcal{O}(n)$ vertices, since we need to place $\mathcal{O}(n)$ mirror and connector edges. Because we construct $k+2$ gadgets, the overall number of vertices used is in $\mathcal{O}(kn)$. The curves T and B can be constructed using a single iteration from left to right, therefore the overall construction takes $\mathcal{O}(kn)$ time. □

6.4.5 Correctness for General Shortcut Curves

When we consider general feasible shortcut curves that might not necessarily be one-touch, they might follow a mirror edge for a short while instead of immediately taking the next shortcut. This results in a small error when comparing the shortcut curve with a one-touch curve encoding the same index set. We now want to contain this incremental error with the control parameter γ.

Lemma 6.4.15. *Choose $\varepsilon > 2$ and $\beta \geq 5$ and all φ_1^i and φ_2^i according to Lemma 6.4.9. If we choose $\gamma > 16k + 5$, then given a feasible shortcut curve $B_\diamond$, it has at least one shortcut end on every exit edge e_*^i, and for every $i \in [k]$ there is exactly one $\iota_i \in [n]$ such that the curve $B_\diamond$ has a shortcut end on edge e_j^i encoding an index set $I = \{\iota_1, \ldots, \iota_k\}$. Further, let v_i be any point of $B_\diamond$ on the exit edge e_*^i of the gadget g_i and let σ_i be the i^{th} partial sum of values encoded by I. Then*

$$b_*^i + \gamma(\sigma_i + 1) - \xi_i \leq v_i \leq b_*^i + \gamma(\sigma_i + 1) + \xi_i$$

holds where $\xi_i = 16i+5$ is an upper bound of the maximum error possible for any shortcut curve traversing up to gadget g_i.

Proof. We prove this claim by induction on i. For $i = 0$ the claim follows by construction of the initialization gadget: As $B_\diamond$ has to start at $B(0)$ and $B_\diamond$ has to lie completely in the hippodrome $B_\diamond$ has to take a shortcut, and since $B_\diamond$ has to pass a twist it must traverse its projection center. The only point where this shortcut can end is on the entry edge of g_1. By construction, this point is at a distance of γ from b_*^0. Since the edge is oriented leftwards, $B_\diamond$ can only follow it in that direction. However, by Lemma 6.4.8, $B_\diamond$ is rightwards 4-monotone. It follows that

$$b_*^0 + \gamma - 4 \leq v_1 \leq b_*^0 + \gamma.$$

Since $\xi_0 = 5 > 4$ and $\sigma_0 = 0$ this implies the claim for $i = 0$.

For $i > 0$ the curve $B_\diamond$ entering gadget g_i from edge e_*^{i-1} has to pass the first twist and must do so through the projection center. By induction

$$b_*^{i-1} + \gamma(\sigma_{i-1} + 1) - \xi_{i-1} \leq v_{i-1} \leq b_*^{i-1} + \gamma(\sigma_{i-1} + 1) + \xi_{i-1}.$$

Since $\gamma > \xi_i = \xi_{i-1} + 16$ it follows that the distance of v_i to the endpoints of the edge is

$$\gamma(\sigma_{i-1} + 1) - \xi_{i-1} \geq \gamma - \xi_{i-1} > 16$$

and

$$\gamma(\sigma_{i-1} + 1) + \xi_{i-1} \leq (\lambda^{i-1} - \gamma) + \xi_{i-1} \leq \lambda^{i-1} - 16,$$

thus v_{i-1} lies at a distance greater than 4 from the endpoints of the entry edge of gadget g_i. Therefore the only edges on which a shortcut through the projection center p_1^i can end are e_j. Let ι be the edge on which this shortcut ends.

Denote by $o_{\max} = \gamma(\sigma_{i-1}+1)+\xi_{i-1}$ and $o_{\min} = \gamma(\sigma_{i-1}+1)-\xi_{i-1}$ the maximal and minimal offset v_i may have from b_*^{i-1}. Furthermore, let α_ι be the y-coordinate of the edge e_ι^i, and similarly α'_ι for the edge $e_\iota'^i$. We will again omit the top index of i since it is fixed for the gadget g_i from now on. Then the interval of x-coordinates where the shortcut may end on e_ι is

$$[c_\iota - \alpha_\iota o_{\max}\,,\ c_\iota - \alpha_\iota o_{\min}].$$

The length of the edge e_ι is $\alpha_\iota \lambda^{i-1}$. Thus the endpoint lies inside the edge. Now $B_\diamond$ may follow this edge as well. Again it can do so only leftwards. As the curve is rightwards 4-monotone it may do so a distance of at most 4. As $\alpha_\iota \geq \frac{1}{2}$ and $o_{\max} > \lambda + 16$, the shortcut curve cannot leave this edge by following it. Thus all possible points for $B_\diamond$ are determined by the interval

$$[c_\iota - \alpha_\iota o_{\max} - 4\,,\ c_\iota - \alpha_\iota o_{\min}].$$

Hence the shortcut curve must leave this edge via a shortcut through $\mathbf{p}_2$. It then may again follow the edge up to a distance of 4 to the left resulting in the interval

$$\left[d'_\iota + \alpha'_\iota o_{\min} - 4\,,\ d'_\iota + \alpha'_\iota\left(o_{\max} + \frac{4}{\alpha_\iota}\right)\right].$$

Repeated application for the next two edges results in the interval for the edge e'

$$\left[\bar{a} - \gamma s_{i,\iota} - \left(o_{\max} + \frac{4}{\alpha_\iota}\right) - 4\,,\ \bar{a} - \gamma s_{i,\iota} - \left(o_{\min} - \frac{4}{\alpha'_\iota}\right)\right].$$

Note that $\bar{a}_\iota = \bar{a} - \gamma s_{i,\iota}$ by construction. And for e_* it lands in the interval

$$\left[b_* + o_{\min} + \gamma s_{i,\iota} - \frac{4}{\alpha'_\iota} - 4\,,\ b_* + o_{\max} + \gamma s_{i,\iota} + \frac{4}{\alpha_\iota} + 4\right].$$

Since $\alpha_\iota \geq \frac{1}{2}$ and $\alpha'_\iota \geq \frac{1}{2}$ holds, we get for the item $s_{i,\iota}$ taken by the shortcut curve

$$b_*^i + \gamma(\sigma_i+1) - \xi_i = b_*^i + \gamma s_{i,\iota} + \gamma(\sigma_{i-1}+1) - \xi_{i-1} - 16 \leq v_i$$

as well as

$$v_i \leq b_*^i + \gamma s_{i,\iota} + \gamma(\sigma_{i-1}+1) + \xi_{i-1} + 16 = b_*^i + \gamma(\sigma_i+1) + \xi_i,$$

implying the claim. □

Theorem 6.1.2. *Unless ETH fails, there is no algorithm for the k-shortcut Fréchet distance decision problem in $\mathbb{R}^d$ for $d \geq 2$ with running time $n^{o(k)}$.*

Proof. Let some k-Table-SUM instance be given. Let $\varepsilon = \frac{5}{2}$, $\beta = 5$ and $\gamma = 16(k+1)+5$. Choose φ_1^i and φ_2^i according to Lemma 6.4.9. Let $B_\diamond$ be any feasible shortcut curve of the constructed instance for the k-Table-SUM instance. Since $B_\diamond$ is feasible, it must visit the exit edge of the last gadget g_k at distance $\gamma(\sigma+1)$ to b_*^k, since this is the only point that connects to $B(1)$ via a shortcut. Let $v_k = b_*^k + \gamma(\sigma+1)$ be the x-coordinate of this visiting point and let $\sigma_\diamond$ be the sum of the values encoded by $B_\diamond$. Lemma 6.4.15 implies that

$$b_*^k + \gamma(\sigma_\diamond+1) - \xi_k \leq v_i = b_*^k + \gamma(\sigma_\diamond+1) \leq b_*^k + \gamma(\sigma_\diamond+1) + \xi_k,$$

since $\gamma = 16(k+1)+5 > \xi_k$. Therefore,

$$\sigma_\diamond - \frac{\xi_k}{\gamma} \leq \sigma \leq \sigma_\diamond + \frac{\xi_k}{\gamma}.$$

Since $\gamma > \xi_k$ it follows that $\sigma_\diamond$ must be σ since both are integers. Hence any feasible shortcut curve solves the k-Table-SUM instance, implying the claim. □

6.5 Approximate Decision Algorithms

In light of the previous section, we now describe a $(3+\varepsilon)$-approximation algorithm for the decision problem of the k-shortcut Fréchet distance of two polygonal curves in the plane. The algorithm has near-quadratic running time in n. In Section 6.5.3, we show that the algorithm can be modified to have running time near-linear in n for c-packed curves.

6.5.1 Description of the Algorithm

We describe how to modify the algorithm of Section 6.3 to circumvent the exponential complexity of the reachable space and obtain a polynomial-time approximation algorithm.

Let two polygonal curves T and B be given, together with a distance threshold Δ and approximation parameter ε. As before, the algorithm (see Algorithm 9) iterates over the cells of the free space diagram and computes sets $N^s_{i,j}$, $V^s_{i,j}$, and $D^s_{i,j}$ for each cell $C_{i,j}$. The main difference now is that, instead of computing the exact set of points that can be reached by a diagonal tunnel, we want to use an approximation for this set. For this, we give an approximate diagonal tunnel procedure (refer to Algorithm 10). This procedure is called with the rightmost point $r^{s-1}_{i-1,j-1}$ in $\bigcup_{a<i;b<j} P^{s-1}_{a,b}$, ε and distance parameter 3Δ. Crucially, the set resulting from one call to the procedure has constant complexity and is sufficient to approximate the set $D^s_{i,j}$. We then compute $P^s_{i,j} = Q(N^s_{i,j} \cup D^s_{i,j} \cup V^s_{i,j}) \cap \mathcal{D}^{(i,j)}_\Delta$, as in Section 6.3. From this we compute (i) the leftmost point $l^s_{i,j}$ in $\bigcup_{b\le j} P^s_{i,b}$ based on $P^s_{i,j}$ and $l^s_{i,j-1}$, (ii) the rightmost point $r^s_{i,j}$ in $\bigcup_{a\le i;b\le j} P^s_{a,b}$ based on $P^s_{i,j}$, $r^s_{i-1,j}$ and $r^s_{i,j-1}$, and (iii) the outgoing reachability intervals of $P^s_{i,j}$. We store these variables for use in the next round. Finally, after k rounds, we check if $(1,1)$ is contained in the computed set of reachable points.

Our approximate diagonal tunnel procedure makes use of a data structure introduced in [DHP12], which is summarized in the following lemma. This data structure is built once for T in the beginning and is then available throughout the algorithm.

Lemma 6.5.1 (Distance oracle [DHP12])**.** *Given a polygonal curve Z with n vertices in $\mathbb{R}^d$ and $\varepsilon > 0$, one can build a data structure $\mathcal{F}_\varepsilon$ in $\mathcal{O}\left(\chi^2 n \log^2 n\right)$ time that uses $\mathcal{O}\left(n\chi^2\right)$ space such that given a query segment $\overline{p\,q}$ and any two points u and v on the curve, one can $(1+\varepsilon)$-approximate $d_\mathcal{F}(\overline{p\,q}, Z[u,v])$ in $\mathcal{O}\left(\varepsilon^{-2}\log n \log\log n\right)$ time, where $\chi = \varepsilon^{-d}\log\left(\varepsilon^{-1}\right)$.*

Definition 6.5.2. Define the scaled integer grid as $\mathbb{G}_\Delta = \left\{(\Delta x, \Delta y) \mid (x,y) \in \mathbb{Z}^2\right\}$.

Approximate Diagonal Tunnel Procedure The procedure (see Algorithm 10) is provided with parameters ε, Δ, some $r' = (r_T, r_B)$ in cell $C_{a,b}$ and the edge e_j that is associated with a cell $C_{i,j}$ as well as the data structure $\mathcal{F}_\varepsilon$ presented in Lemma 6.5.1. We want to compute a set of stabbers starting at $r = B(r_B)$ that contains every stabber through the disks $\mathrm{D}_\Delta(v_{a+1}), \ldots, \mathrm{D}_\Delta(v_i)$, and is contained in the set of all stabbers through disks of radius $(1+\varepsilon)^2\Delta$ centered at the same vertices. We approximate this set of stabbers as follows.

We iterate over all grid points t in the disk $\mathbb{G}_{\frac{\Delta\varepsilon}{\sqrt{2}}} \cap \mathrm{D}_{(1+\varepsilon)\Delta}(v_i)$, and make queries to the data structure $\mathcal{F}_\varepsilon$ to determine if the Fréchet distance of the query segment $\overline{r\,t}$ to

Algorithm 9 Approximate Decider

procedure ApproximateDecider(curve T, curve B, $\Delta > 0$, $0 < \varepsilon \leq 1$)
  if $||T(0) - B(0)|| > \Delta$ or $||T(1) - B(1)|| > \Delta$ **then**
    return '$\mathrm{d}_{\mathcal{S}}^{k}(T', B') > \Delta$'
  Let $\mathcal{F}_{\varepsilon}$ be the data structure of Lemma 6.5.1 built on T with ε
  Let $\mathcal{A}^s$ and $\bar{\mathcal{A}}^s$ be arrays of size n_1 for each $0 \leq s \leq k$
  Let g_r^s and g_l^s be two-dimensional arrays of size $n_1 \times n_2$ for each $0 \leq s \leq k$
  for $s = 0, \ldots, k$ **do**
    for $j = 1, \ldots, n_2$ **do**
      Copy array $\mathcal{A}^s$ into $\bar{\mathcal{A}}^s$
      for $i = 1, \ldots, n_1$ **do**
        Compute $\mathcal{D}_{\Delta}^{(i,j)}$
        if $i = 1, j = 1$ and $s = 0$ **then**
          $P_{i,j}^s = \mathcal{D}_{\Delta'}^{(i,j)}$
        else
          //Compute set of points directly reachable from neighboring cells
          Let $I_v = \emptyset$ and $I_h = \emptyset$
          if $j > 1$ **then** I_v is the incoming reachability interval from $\bar{\mathcal{A}}^s[i]$
          if $i > 1$ **then** I_h is the incoming reachability interval from $\mathcal{A}^s[i-1]$
          Let $N_{i,j}^s = (Q(I_v) \cup Q(I_h)) \cap \mathcal{D}_{\Delta}^{(i,j)}$
          if $s > 0$ **then**
            //Approximate set of points reachable by diagonal tunnel
            Let $r = g_r^{s-1}[i-1, j-1]$ be the rightmost point in $P_{<i,<j}^{s-1}$
            Let $D_{i,j}^s =$ apxDiagonalTunnel$(r, (i,j), \varepsilon, \mathcal{F}_{\varepsilon}, 3\Delta)$
            //Compute set of points reachable by vertical tunnel
            Let $l = g_l^{s-1}[i, j-1]$ be the leftmost point in $P_{i,<j}^{s-1}$.
            Let $V_{i,j}^s =$ verticalTunnel$(l, (i,j), \Delta)$
          else
            Let $D_{i,j}^s = \emptyset$ and $V_{i,j}^s = \emptyset$
          $P_{i,j}^s = Q(N_{i,j}^s \cup D_{i,j}^s \cup V_{i,j}^s) \cap \mathcal{D}_{\Delta}^{(i,j)}$
        if $P_{i,j}^s \neq \emptyset$ **then**
          Store rightmost point among $P_{i,j}^s$, $g_r^s[i-1, j]$ and $g_r^s[i, j-1]$ in $g_r^s[i,j]$
          Store leftmost point among $P_{i,j}^s$ and $g_l^s[i, j-1]$ in $g_l^s[i,j]$
          Compute outgoing reachability intervals using $P_{i,j}^s$
          Store outgoing reachability intervals in $\mathcal{A}^s[i]$.
  if $(1,1) \in \mathcal{A}^s[n_1]$ **then**
    return '$\mathrm{d}_{\mathcal{S}}^{k}(T', B') \leq 3(1+\varepsilon)^2\Delta$' with $s \leq k$ shortcuts
  else
    return '$\mathrm{d}_{\mathcal{S}}^{k}(T', B') > \Delta$' with at most k shortcuts

Algorithm 10 Approximate Diagonal Tunnel

procedure APXDIAGONALTUNNEL((r_T, r_B), (i, j), ε, $\mathcal{F}_\varepsilon$, Δ)
  Let $r = B(r_B)$
  for $t \in \left(\mathbb{G}_{\frac{\Delta\varepsilon}{\sqrt{2}}} \cap \mathrm{D}_{3(1+\varepsilon)\Delta}(v_i) \right)$ **do**
    Query $\mathcal{F}_\varepsilon$ for the distance $\mathrm{d}_{\mathcal{F}}(\overline{r\,t}, T[r_T, v_i])$ and store the answer in Δ'
    if $\Delta' \leq (1+\varepsilon)^2\Delta$ **then**
      Mark t as eligible
  Compute the convex hull H of eligible points
  if $r \in H$ **then**
    return $C = \mathcal{D}_\Delta^{(i,j)}$
  else
    Let U be the cone with apex r formed by tangents t_1 and t_2 from r to H
    Let $p_i \in H$ be a supporting point of the tangent t_i for $i \in \{1, 2\}$
    Let L be the subchain of ∂H with endpoints p_1 and p_2 which is facing r
    Let $H' \subset U$ be the set bounded by L and the rays supported by t_1 and t_2 facing away from r
    Let C' be the set of points on edge e_j in H'
    return the set C of points in $\mathcal{D}_\Delta^{(i,j)}$ corresponding to C'

the subcurve of T from $T(r_T)$ to v_i is sufficiently small. We mark t if the approximate distance returned by the data structure is at most $(1+\varepsilon)^2\Delta$. We then compute the convex hull H of all marked grid points, and the two tangents t_1 and t_2 of H through $B(r_B)$. The true set of endpoints of stabbers is approximated by the set H' of points that lie inside and 'behind' the convex hull H, from the perspective of r. Figure 6.11 illustrates this. We then intersect H' with the edge e_j resulting in a single horizontal slab in $C_{i,j}$. This resulting set is then intersected with $\mathcal{D}_\Delta^{(i,j)}$ and returned.

6.5.2 Analysis of the Approximation Algorithm

We now analyze the described algorithm, namely the ApproximateDecider in Algorithm 9 procedure.

We argue that the structure of $P_{i,j}^s$ as approximated by the ApproximateDecider procedure is as claimed. Namely for all i, j and s it holds that

$$\mathcal{R}_{\Delta,s}^{(i,j)} \subset \bigcup_{0 \leq s' \leq s} P_{i,j}^{s'} \subset \mathcal{R}_{3(1+\varepsilon)^2\Delta,s}^{(i,j)}.$$

We again consider any monotone path with s proper tunnels ending in some cell and show the set inclusion by induction. To prove correctness, we make use of the following lemma from [DHP12]. Intuitively, the lemma states that if a feasible tunnel $\tau(r, q)$ costs more than 3Δ, then any feasible tunnel $\tau(p, q)$ with $x_p \leq x_r$ costs more than Δ.

Lemma 6.5.3 (monotonicity of tunnels [DHP12])**.** *Given a value $\Delta > 0$ and two curves T_1 and T_2 such that T_2 is a subcurve of T_1, and given two line segments $\bar{B}_1$ and $\bar{B}_2$ such that $\mathrm{d}_{\mathcal{F}}(T_1, \bar{B}_1) \leq \Delta$ and the start and end point of T_2 are within distance Δ to the start and end point of $\bar{B}_2$ respectively, then $\mathrm{d}_{\mathcal{F}}(T_2, \bar{B}_2) \leq 3\Delta$.*

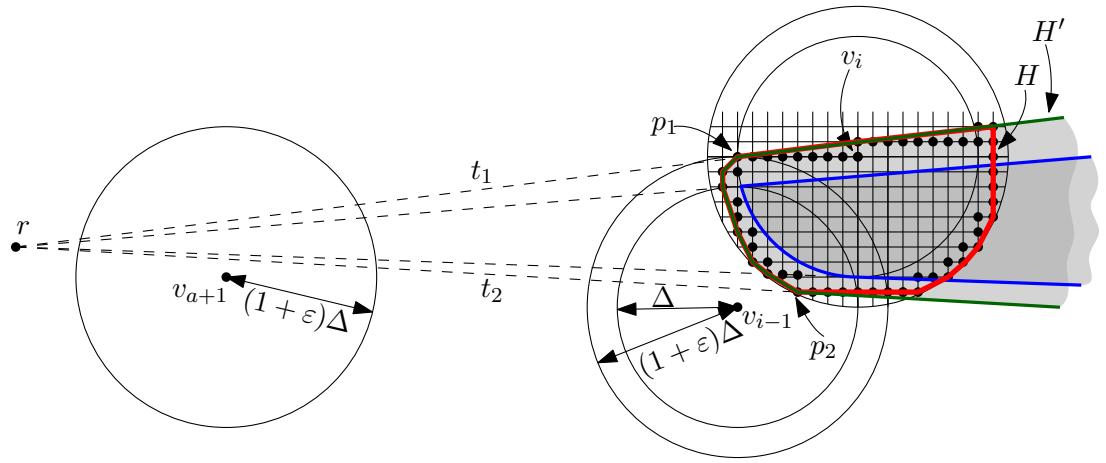


Figure 6.11: Illustration to the approximate diagonal tunnel procedure. The true line-stabbing wedge for disks with radius Δ is shown in blue. The convex hull of eligible grid points is shown in red. The approximate line stabbing wedge is shown in green.

In the following, we denote with $\{\mathrm{D}_\Delta(v_i)\}_i$ a finite sequence of disks $\{\mathrm{D}_\Delta(v_1), \ldots\}$.

Lemma 6.5.4. *Let $a, b_1, b_2 \in \mathbb{R}^d$ together with a sequence of vertices $v_1, \ldots, v_n$ be given. If $\overline{a\,b_1}$ stabs through disks $\{\mathrm{D}_\Delta(v_i)\}_i$, and $\overline{a\,b_2}$ stabs through $\{\mathrm{D}_\Delta(v_i)\}_i$, then for any $t \in [0,1]$ the line segment $\overline{a\,b(t)}$ stabs through $\{\mathrm{D}_\Delta(v_i)\}_i$, where $b(t) = b_1 + t(b_2 - b_1)$.*

Proof. Refer to Figure 6.12. Consider the triangle with sides $(b_1 - a)$, $(b_2 - a)$, and $(b_1 - b_2)$, where the first two sides correspond to the original stabbers and the last side to $b(t)$. Note that any line segment $\overline{a\,b(t)}$ lies completely within this triangle with $(b_1 - a)$ on the one and $(b_2 - a)$ on the other side. Hence, for every i and realizing points p_i of $\overline{a\,b_1}$ and q_i of $\overline{a\,b_2}$, p_i lies on the one and q_i on the other side of $\overline{a\,b(t)}$. Since $\mathrm{D}_\Delta(v_i)$ is convex and p_i and q_i are inside this disk, the intersection of $\overline{p_i\,q_i}$ and $\overline{a\,b(t)}$ is inside the disk as well. Let r_i denote this intersection point. The points $\{r_i\}_i$ are realizing points for $\overline{a\,b(t)}$. This follows from the fact that $\{p_i\}_i$ and $\{q_i\}_i$ are ordered along their respective line segments, and thus $\overline{p_i\,q_i}$ never crosses another $\overline{p_j\,q_j}$. Thus for $i < j$, r_i appears before r_j along $\overline{a\,b(t)}$, implying the claim. □

Lemma 6.5.5. *Let $a_1, a_2, b_1, b_2 \in \mathbb{R}^2$ together with a sequence of vertices $v_1, \ldots, v_n$ be given. If $\overline{a_1\,b_1}$ stabs through $\{\mathrm{D}_\Delta(v_i)\}_i$, and $\|a_1 - b_1\| \leq \Delta'$ and $\|a_2 - b_2\| \leq \Delta'$, then $\overline{a_1\,b_1}$ stabs through $\{\mathrm{D}_{\Delta+\Delta'}(v_i)\}_i$.*

Proof. By Observation 2.1.1, $\mathrm{d}_{\mathcal{F}}(\overline{a_1\,b_1}, \overline{a_2\,b_2}) \leq \Delta'$ via the reparameterization (f, g) with $f(t) = t$ and similarly $g(t) = t$. As $p = \overline{a_1\,b_1}$ stabs through $\{\mathrm{D}_\Delta(v_i)\}_i$, there exist realizing points $p_i = p(t_i)$ for some ordered values $\{t_i\}_i$, with p_i lying in the Δ-disk centered at v_i. Then

$$\|q(t_i) - v_i\| \leq \|q(t_i) - p(t_i)\| + \|p(t_i) - v_i\| \leq \Delta' + \Delta.$$ □

Lemma 6.5.6. *Given $r \in \mathbb{R}^2$, $C_{i,j}$, ε and Δ like in the* apxDiagonalTunnel *procedure. Denote by S_Δ the set of endpoints of all Δ-stabbers (that is, stabbers through $\mathrm{D}_\Delta(v_m)$ for $a + 1 \leq m \leq i$) on the edge e_j starting at r and let C' be the point set computed in line 15 of the procedure. Then*

$$S_\Delta \subseteq C' \subseteq S_{(1+\varepsilon)^2\Delta}.$$

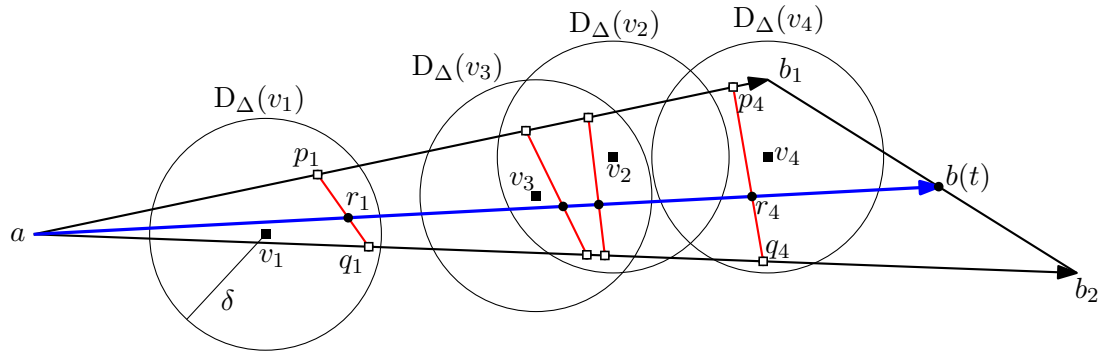


Figure 6.12: Linear interpolation between two Δ-stabbers starting in a. Illustrations to the proof of Lemma 6.5.4. In blue $\overline{a\,b(t)}$, and in red $\overline{p_i\,q_i}$ is illustrated. Their intersections form the realizing points r_i of $\overline{a\,b(t)}$.

Proof. Let $y \in C'$. Then $q = B(y) \in H'$ where H' is set of points computed by the algorithm. Denote the intersection of $\overline{r\,q}$ and the boundary of H' by h. The point h is then a linear combination of at most two grid points whose stabbers from r have been marked as eligible, i.e., who are $(1+\varepsilon)^2\Delta$-stabber. Hence, Lemma 6.5.4 implies that $\overline{r\,q}$ is also a $(1+\varepsilon)^2\Delta$-stabber, implying $C' \subseteq S_{(1+\varepsilon)^2\Delta}$.

Now let $y \in S_\Delta$ be an arbitrary value such that for $q = e_j(y)$ the edge $\overline{r\,q}$ is a Δ-stabber. Let t be the last realizing point of $\overline{r\,q}$. The line segment $\overline{r\,t}$ is a Δ-stabber and t lies in $\mathrm{D}_\Delta(v_i)$. We claim that t lies in H. Consider the set $G = \mathbb{G}_{\frac{\Delta\varepsilon}{\sqrt{2}}} \cap \mathrm{D}_{\varepsilon\Delta}(t)$. By the properties of the grid, t lies within the convex hull of G. Moreover, $G \subset \mathrm{D}_{(1+\varepsilon)\Delta}(v_i)$. Lemma 6.5.5 implies that $\overline{r\,t'}$ is a $((1+\varepsilon)\Delta)$-stabber for any $t' \in G$. This in turn implies that for the first point s' of $\overline{r\,t'}$ inside $\mathrm{D}_{\Delta(1+\varepsilon)}(v_a)$, $\overline{s'\,t'}$ is a $((1+\varepsilon)\Delta)$-stabber, hence, t' would have been marked as an eligible endpoint as the distance computed via $\mathcal{F}_\varepsilon$ is at most $(1+\varepsilon)^2\Delta$ (by Lemma 6.5.1). Since H is the convex hull of eligible points, it follows that $t \in \mathrm{conv}(G) \subset H$. Therefore $q \in H'$ and thus $y \in C'$. □

Lemma 6.5.7. *For any $i \in [n_1]$, $j \in [n_2]$, and $s \in [k]$, let $D^s_{i,j}$ be the endpoints of diagonal tunnels as computed in the* ApproximateDecider *procedure, and let $R = \bigcup_{a=1}^{i-1}\bigcup_{b=1}^{j-1} P^{s-1}_{a,b}$ be the set of reachable points by exactly $s-1$ proper tunnels in the lower-left quadrant of the cell $C_{i,j}$. It holds that*

(i) there exists a point $p \in R$ such that for any $q \in C_{i,j}$ if the diagonal tunnel $\tau(p,q)$ has price $\mathrm{prc}(\tau(p,q)) \leq 3\Delta$ then $q \in D^s_{i,j}$. If $q \in D^s_{i,j}$, then $\mathrm{prc}(\tau(p,q)) \leq 3(1+\varepsilon)^2\Delta$, and

(ii) there exists no other $b \in C_{i,j} \setminus D^s_{i,j}$ that is the endpoint of a diagonal tunnel from R with price at most Δ.

Proof. The first part follows from Lemma 6.5.6 together with the process described by the algorithm: The point p is simply the rightmost point in R, which is maintained in g'_r (by an induction argument) at the time, where $C_{i,j}$ is processed. Let $p = (x_p, y_p)$ lie in cell $C_{a,b}$. We call the apxDiagonalTunnel procedure with p and the vertices $v_{a+1}, \ldots, v_i$ between the a^{th} and i^{th} edge of the target curve. It returns points $q = (x_q, y_q)$ inside the Δ-free space such that $\overline{B}[y_p, y_q]$ stabs through the sequence $\mathrm{D}_{3(1+\varepsilon)^2\Delta}(v_{a+1}), \ldots, \mathrm{D}_{3(1+\varepsilon)^2\Delta}(v_i)$. Since p and q are in the Δ-free space of

B and T, $\|T(x_p) - B(y_p)\| \leq \Delta$ and $\|T(x_q) - B(y_q)\| \leq \Delta$, which together imply $\mathrm{d}_{\mathcal{F}}(\overline{B}[y_p, y_q], T[x_p, x_q]) \leq 3(1+\varepsilon)^2\Delta$.

Assume for the sake of contradiction of the second part that such a point b does exist and the start point of the shortcut is $s \in R$. Then by Lemma 6.5.3, all tunnels $\tau(r, b)$ with $x_s < x_r$ have price at most 3Δ. In particular $\mathrm{prc}(\tau(p, b)) \leq 3\Delta$, but then b would have been in $D^s_{i,j}$ already. □

Lemma 6.5.8. *Given two polygonal curves T and B in the plane as well as parameters $\varepsilon > 0$ and $\Delta > 0$, the* ApproximateDecider *computes a decision of either $d^k_{\mathcal{S}}(B, T) > \Delta$ or $d^k_{\mathcal{S}}(B, T) \leq 3(1+\varepsilon)^2\Delta$.*

Proof. We show that

$$\mathcal{R}_{\Delta,s}(B, T) \subset \bigcup_{s' \leq s} P^{s'} \subset \mathcal{R}_{3(1+\varepsilon)^2\Delta,s}(B, T),$$

for $s \leq k$.

This proof is by induction on the order of handled cells. We show the inclusions from the theorem for each cell, i.e.,

$$\mathcal{R}^{(i,j)}_{\Delta,s}(B, T) \subset \bigcup_{s' \leq s} P^{s'}_{i,j} \subset \mathcal{R}^{(i,j)}_{3(1+\varepsilon)^2\Delta,s}(B, T).$$

Assume that $(0, 0) \in \mathcal{D}^{(1,1)}_{\Delta}$, as otherwise the algorithm would have returned a correct decision in line 3. For $i = j = 1$, we have that $P^0_{i,j} = \mathcal{D}^{(1,1)}_{\Delta}$ which is correct by convexity of $\mathcal{D}^{(1,1)}_{\Delta}$. For all other s we have that $P^s_{1,1} = \emptyset$. This follows from the fact that there are no points in the column below or in the lower-left quadrant of $C_{1,1}$. Thus, for $i = j = 1$, we have $\bigcup_{s' \leq s} P^{s'}_{i,j} = \mathcal{R}^{(i,j)}_{\Delta,s} \subset \mathcal{R}^{(i,j)}_{3(1+\varepsilon)^2\Delta,s}$.

Consider the algorithm handling some cell $C_{i,j}$. By induction, all cells $C_{\leq n_1, <j}$ and $C_{<i,j}$ and in particular $C_{i-1,j}$ and $C_{i,j-1}$ have been handled correctly up to s. Hence, their reachability intervals and left- and rightmost points have been computed correctly and are stored in their respective arrays. We need to show that $\mathcal{R}^{(i,j)}_{\Delta,s} \subset \bigcup_{s' \leq s} P^{s'}_{i,j}$. Thus let $q \in \mathcal{R}^{(i,j)}_{\Delta,s}$ be the endpoint of a monotone path from $(0, 0)$ walking monotonously through $\mathcal{D}^{(i,j)}_{\Delta}$ using $s' \leq s$ proper tunnels of cost Δ. There are three possibilities of how the path could have entered $C_{i,j}$.

The path could have taken s' shortcuts to enter a neighboring cell and then walked into $C_{i,j}$ through its boundary at some point a. Since $C_{i-1,j}$ and $C_{i,j-1}$ have been handled correctly, a is in the computed reachability interval of the neighboring cell. Since the path must be monotone, q lies in the closed halfplane fixed at the lower or left end of the reachability interval in the respective directions, thus q is also in $P^{s'}_{i,j}$. Alternatively the path could have entered some cell $C_{i,l}$ with $s' - 1$ shortcuts and then took a horizontal shortcut into $C_{i,j}$ for some $j < l$. By Lemma 6.3.2 together with the induction hypothesis for $P_{i,<j}$ we have that q is in $P^{s'}_{i,j}$. Similarly, if the path took a diagonal shortcut, we can apply Lemma 6.5.7 together with the induction hypothesis for $P_{<i,<j}$, showing that q is in $P^{s'}_{i,j}$, implying the left inclusion $\mathcal{R}^{(i,j)}_{\Delta,s} \subset \bigcup_{s' \leq s} P^{s'}_{i,j}$.

Now let $q \in P^{s'}_{i,j}$ for some $s' \leq s$. Then either (i) q is in $N^s_{i,j}$, (ii) q is in $V^s_{i,j}$, (iii) q is in $D^s_{i,j}$, or (iv) q is in the upper right quadrant of some point p, where p satisfies (i), (ii) or (iii). Thus it suffices to analyze the first three cases. For (i), observe that

$P^{s'}_{i-1,j}$ and $P^{s'}_{i,j-1}$ have been computed correctly up to round s', and thus q must also be in $\mathcal{R}^{(i,j)}_{3(1+\varepsilon)^2\Delta,s}(B,T)$. For (ii), observe that $P^{s'}_{i,<j}$ have been computed correctly and the leftmost point is stored correctly in $\bar{g}_l^{s'-1}$, hence Lemma 6.3.2 implies that q must also be in $\mathcal{R}^{(i,j)}_{3(1+\varepsilon)^2\Delta,s}(B,T)$. Finally, for (iii), observe that $P^{s'}_{<i,<j}$ have been computed correctly. Hence, the rightmost point in the lower-left quadrant of $C_{i,j}$ that was reachable by $s'-1$ shortcuts is correctly stored in $\bar{g}_r^{s'-1}$. By Lemma 6.5.7, the apxDiagonalTunnel computes endpoints of shortcuts that are contained within the set of shortcuts with price at most $3(1+\varepsilon)^2\Delta$. Thus q is in $\mathcal{R}^{(i,j)}_{3(1+\varepsilon)^2\Delta,s}(B,T)$. Hence $P^{s'}_{i,j}$ is computed correctly and all relevant left- and rightmost points as well as reachability intervals are stored correctly. Hence, $\mathcal{R}_{\Delta,s}(T',B') \subset \bigcup_{s'\leq s} P^{s'} \subset \mathcal{R}_{3(1+\varepsilon)^2\Delta,s}(T',B')$. Finally, the algorithm output corresponds to whether $(1,1)$ is in $P^{\leq k}$ proving the claim. □

Theorem 6.5.9. *Let T and B be two polygonal curves in the plane with overall complexity n, together with values $\varepsilon \in (0,1]$ and $\Delta > 0$. There exists an algorithm with running time in $\mathcal{O}\left(kn^2\varepsilon^{-5}\log^2\left(n\varepsilon^{-1}\right)\right)$ and space in $\mathcal{O}\left(kn^2\varepsilon^{-4}\log^2\left(\varepsilon^{-1}\right)\right)$ which outputs one of the following: (i) $d^k_{\mathcal{S}}(B,T) \leq (3+\varepsilon)\Delta$ or (ii) $d^k_{\mathcal{S}}(B,T) > \Delta$. In any case, the output is correct.*

Proof. We claim that the ApproximateDecider procedure (after rescaling $\varepsilon \leftarrow \varepsilon/9$) fulfills these requirements.

For the precomputation we construct the data structure from Lemma 6.5.1. This precomputation takes $\mathcal{O}\left(\varepsilon^{-4}\log^2\left(\varepsilon^{-1}\right)n\log^2(n)\right)$ time. We iterate over all $\mathcal{O}(n^2)$ cells k times. The computation for each of these $\mathcal{O}(kn^2)$ steps is dominated by a call to the apxDiagonalTunnel procedure. This procedure iterates over $\mathcal{O}\left(\varepsilon^{-2}\right)$ gridpoints, thus queries the data structure $\mathcal{O}\left(\varepsilon^{-2}\right)$ times where each query takes $\mathcal{O}\left(\varepsilon^{-2}\log n\log\log n\right)$ time. Finally, we construct a convex hull and intersect it with a line. This can be done in $\mathcal{O}\left(\varepsilon^{-2}\log\varepsilon^{-1}\right)$ time as we construct the convex hull of $\mathcal{O}\left(\varepsilon^{-2}\right)$ points. Thus the overall running time of the apxDiagonalTunnel procedure is $\mathcal{O}\left(\varepsilon^{-4}\log n\log\log n\right)$. Thus the overall running time of the ApproximateDecider procedure is

$$\begin{aligned}
&\mathcal{O}\left(\varepsilon^{-4}\log^2\left(\varepsilon^{-1}\right)n\log^2(n) + kn^2\varepsilon^{-1}\left(\varepsilon^{-4}\log n\log\log n\right)\right)\\
=\ &\mathcal{O}\left(\varepsilon^{-5}n\log^2(n) + kn^2\varepsilon^{-5}\log n\log\log n\right)\\
=\ &\mathcal{O}\left(kn^2\varepsilon^{-5}\log^2\left(n\varepsilon^{-1}\right)\right).
\end{aligned}$$

The space bound is a consequence of the space needed for the approximate distance data structure. All other data structures necessary for the algorithm use $\mathcal{O}(kn^2)$ or $\mathcal{O}\left(\varepsilon^{-2}\right)$ space. Hence, the space is in $\mathcal{O}\left(kn^2 + n\varepsilon^{-4}\log^2\left(\varepsilon^{-1}\right)\right)$. The correctness of the output is guaranteed by Lemma 6.5.8. Lastly, notice that due to the rescaling of $\varepsilon \leftarrow \varepsilon/9$ we have that $3(1+\varepsilon/9)^2\Delta < (3+\varepsilon)\Delta$. □

6.5.3 Modified Algorithm for c-Packed Curves

In the case that the input curves are c-packed, for some constant c, we can modify the algorithm and achieve near-linear running time in n. For this, we follow the approach in [DHW12] to first simplify the curves. Recall that Theorem 4.4.5 (on page 63) implies that Algorithm 3 (on page 62) computes a $(0,\varepsilon)$-maximal simplification of a polygonal curve with complexity n in $\mathcal{O}(n)$ time. Further, recall that by Lemma 4.4.10, there are at most $\mathcal{O}(cn/\varepsilon^2)$ non-empty cells in a suitable free space of a $(2\Delta,\varepsilon\Delta)$-maximal

simplification and $(0, \varepsilon\Delta)$-maximal simplification of a c-packed curve. This analysis goes back to the following lemma due to [DHW12].

Lemma 6.5.10 ([DHW12, Lemma 4.4])**.** *For any two c-packed curves B and T in $\mathbb{R}^d$ of total complexity n, and two parameters $\varepsilon \in (0, 1]$ and $\Delta > 0$ and $(0, \varepsilon\Delta)$-maximal simplifications B^* and T^* of B and T respectively, there are at most $\mathcal{O}(cn\varepsilon^{-1})$ cells in the free space diagram with non-empty Δ-free space.*

As for a $(0, \varepsilon\Delta)$-maximal simplification B^* of B we have that $\mathrm{d}_{\mathcal{F}}(B, B^*) \leq \varepsilon\Delta$ and implying the following lemma.

Lemma 6.5.11 ([DHW12])**.** *Given a simplification parameter μ and two polygonal curves B and T, let B^* and T^* denote $(0, \mu)$-maximal simplifications of B and T. For all $k \in \mathbb{N}$ it holds that*

$$d_{\mathcal{S}}^k(B^*, T^*) - 2\mu \leq d_{\mathcal{S}}^k(B, T) \leq d_{\mathcal{S}}^k(B^*, T^*) + 2\mu.$$

Modifications

The three major modifications we apply to Algorithm 9, in order the achieve near-linear running time, are the following.

First, we compute $(0, \varepsilon\Delta)$-maximal simplifications of both input curves T and B, such that $\mathcal{D}_\Delta(B', T')$ only has $\mathcal{O}(cn\varepsilon^{-1})$ non-empty cells by Lemma 6.5.10.

Secondly, instead of iterating over all cells, we only want to iterate over these non-empty cells. We solve this with an output-sensitive algorithm for computing the intersections of edges and the boundary of Δ-neighborhoods of these edges. For this we first compute the $\mathcal{O}(n)$ boundaries of neighborhoods of edges, whose geometric shape we refer to as a capsule in $\mathcal{O}(n)$ time. We then compute the intersections between all edges and capsules of B and T with a slight modification (to handle capsules) of the classical sweep line algorithm presented in [BO79]. From these intersections, we can then reconstruct which cells have non-empty Δ-free space.

Lastly, in order to store and retrieve the left- and rightmost points in a column below and in the lower-left quadrant of a cell, we use two dimensional range trees described in [dBCvKO08]. Both storing and retrieving takes logarithmic time, but now we are able to retrieve these points, while only storing and updating these points whenever we are in a non-empty cell.

Intersection Finder

In this section we describe the aforementioned slight modification of the sweep line algorithm presented in [BO79] to compute which cells of the Δ-free space are non-empty.

Lemma 6.5.12. *Given two polygonal curves B and T in $\mathbb{R}^2$, a parameter $\Delta \geq 0$ and let B^* and T^* be $(0, \varepsilon\Delta)$-maximal simplifications of B and T. One can find all $\mathcal{O}(\frac{cn}{\varepsilon})$ cells in the free space diagram that have non-empty Δ-free space in $\mathcal{O}(\frac{cn}{\varepsilon}\log(\frac{cn}{\varepsilon}))$ time.*

Proof. Without loss of generality it suffices to find all edges of the curve B^* that enter and exit a Δ-neighborhood of any edge of T^*, since any edge that is completely contained in this neighborhood lies between two edges entering and leaving the neighborhood. In

Algorithm 11 Intersection Finder

```
1: procedure IntersectionFinder((0, εΔ)-maximal simplifications X* and Y*, Δ)
2:     Insert every start- and end-point of edges and every start- and end-points of arcs
       of Δ-capsules of edges of X* and Y* like in Figure 6.13 into a priority queue E
3:     Sort E by firstly the x-coordinate and secondly by the y-coordinate
4:     Let A be a self-balancing empty binary tree and I an empty array
5:     while E ≠ ∅ do
6:         pop the head object off E into x
7:         if x is the first vertex to be inserted of an object o then
8:             sorted insert o into A by its current y-coordinate
9:             compute the intersection point of x and its at most two neighbors
10:            sorted insert this point by its x-coordinate into E
11:        if x is the second vertex to be inserted of an object o then
12:            let l and r be the two neighbors of o
13:            remove o from A
14:            compute the intersection point of l and r updating E
15:        if x corresponds to an intersection between o and o' then
16:            insert the intersection to I
17:            swap o and o' in A
18:            compute the intersection point of new neighbors updating E
19:    Return I
```

the special case that the start or end vertex of B^* lies in such a neighborhood it is easily checked by looking whether the first (resp. last) such edge is entering or leaving the neighborhood. Entering and exiting such a neighborhood is the same as intersecting its boundary. Thus we can modify for example the classical sweep-line algorithm to find all intersections in a set of edges as introduced in [BO79] (refer to Algorithm 11).

We sweep along the x-axis and, in an array of size $\mathcal{O}(n)$, keep track of all objects that cross the sweeping line. Every time a new object enters the array it checks with its at most two neighbors how far the sweeping line would have to sweep to get to the intersection point of the new object. If an intersection occurs at some time in the future, we add this event to the event queue of the sweeping line. If the sweeping line is at an intersection event, it swaps the two objects in question and updates all new $\mathcal{O}(1)$ neighbors. We can modify this easily to work with capsules (the geometric shape of the Δ-neighborhood of an edge) by introducing two sections of the capsule into the array instead of a single line, as can be seen in Figure 6.13. Intersections with its neighbors can still be checked and updated in $\mathcal{O}(1)$. The algorithm runs in $\mathcal{O}((n+k)\log(n+k))$ time for k intersecting objects.

By Lemma 6.5.10 there are $\mathcal{O}(\frac{cn}{\varepsilon})$ cells in the 2Δ-free space of B^* and T^*, B^* and B^*, and T^* and T^* each. Hence the number of intersections of the described objects is in $\mathcal{O}(\frac{cn}{\varepsilon})$, implying the claim. □

Analysis of the Algorithm

We now turn to analyzing the modified algorithm as described above.

Theorem 6.1.3. *Let T and B be two c-packed curves in the plane with overall complexity n, together with values $\varepsilon \in (0,1]$ and $\Delta > 0$. There exists an algorithm with running*

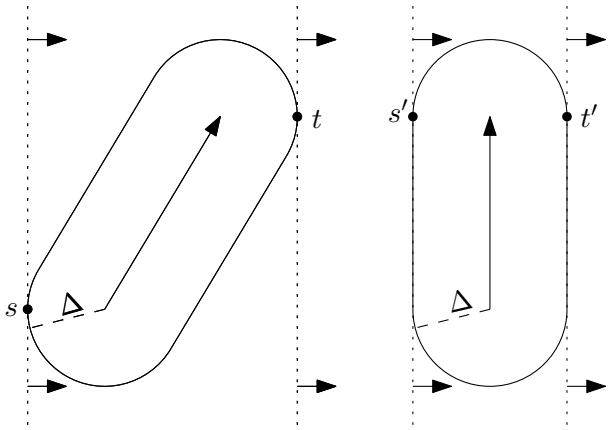


Figure 6.13: Arc definition for the intersection procedure.

time in $\mathcal{O}\left(kcn\varepsilon^{-5}\log^2\left(n\varepsilon^{-1}\right)\right)$ *and space in* $\mathcal{O}\left(kcn\varepsilon^{-4}\log^2\left(n\varepsilon^{-1}\right)\right)$ *which outputs one of the following: (i)* $d_{\mathcal{S}}^k(B,T) \leq (3+\varepsilon)\Delta$ *or (ii)* $d_{\mathcal{S}}^k(B,T) > \Delta$*. In any case, the output is correct.*

Proof. The algorithm first computes $(0,\varepsilon\Delta)$-maximal simplifications T^* and B^* of T and B, and all cells of the free space diagram with non-empty Δ-free space. It then invokes the algorithm ApproximateDecider from Algorithm 9 replacing the two loops iterating over all cells to instead loop over the identified non-empty cells in lexicographic order. Lemma 6.5.8 guarantees a correct decision of either $d_{\mathcal{S}}^k(B^*,T^*) > \Delta$ or $d_{\mathcal{S}}^k(B^*,T^*) \leq 3(1+\varepsilon)^2\Delta$. By Lemma 6.5.11, this decision implies a correct decision of either $d_{\mathcal{S}}^k(B,T) > \Delta - 2\varepsilon\Delta = (1-2\varepsilon)\Delta$ or $d_{\mathcal{S}}^k(B^*,T^*) \leq 3(1+\varepsilon)^2\Delta + 2\varepsilon\Delta = (3(1+\varepsilon)^2+2\varepsilon)\Delta$.

Rescaling $\Delta \leftarrow \Delta/(1-\varepsilon/10)$ and $\varepsilon \leftarrow \varepsilon/20$ does not affect the asymptotic running time, but turns the decision output by the algorithm into a correct decision of either $d_{\mathcal{S}}^k(B,T) > \Delta$ or $d_{\mathcal{S}}^k(B^*,T^*) \leq (3+\varepsilon)\Delta$, as

$$\left(1-\frac{2\varepsilon}{20}\right)\frac{\Delta}{1+\varepsilon/10} = \Delta,$$

and as $\varepsilon \leq 1$ we have that

$$\frac{3\left(1+\frac{\varepsilon}{20}\right)^2 + 2\frac{\varepsilon}{20}}{1-\frac{\varepsilon}{10}}\Delta \leq (3+\varepsilon)\Delta.$$

Now for the running time observe that the non-trivial steps of the algorithm are: (i) Precomputation on the curves, (ii) finding all non-empty cells, (iii) iterating over these cells, (iv) the apxDiagonalTunnel procedure, and (v) storing and restoring the rightmost point in the lower-left quadrant of any cell. By Theorem 4.4.5, we can compute the $(0,\varepsilon\Delta)$-simplifications in $\mathcal{O}(n)$ total time.

For the precomputation we initialize the data structure described in Lemma 6.5.1. This precomputation takes $\mathcal{O}\left(\varepsilon^{-4}\log^2\left(\varepsilon^{-1}\right)n\log^2(n)\right)$ time. By Lemma 6.5.12, finding all intersections can be done in $\mathcal{O}\left(cn\varepsilon^{-1}\log\left(cn\varepsilon^{-1}\right)\right)$ time. Sorting these intersections in $\mathcal{O}\left(cn\varepsilon^{-1}\log\left(cn\varepsilon^{-1}\right)\right)$ time lexicographically by the indices of the corresponding cell allows us to iterate over the cells with non-empty Δ-free space as described in the algorithm. In Section 6.5.1 we described the apxDiagonalTunnel procedure. As described in the proof of Theorem 6.5.9, the overall running time of the apxDiagonalTunnel

procedure is $\mathcal{O}\left(\varepsilon^{-4}\log n\log\log n\right)$. We call this procedure $\mathcal{O}\left(kcn\varepsilon^{-1}\right)$ times, k times for each non-empty cell.

To store and retrieve the rightmost point in the lower-left quadrant we can use two two-dimensional range trees as described in [dBCvKO08]. We build these trees with $\mathcal{O}\left(cn\varepsilon^{-1}\right)$ points at the end of each outer loop storing all right- and leftmost points for the next iteration in $\mathcal{O}\left(cn\varepsilon^{-1}\log\left(cn\varepsilon^{-1}\right)\right)$ time. As we do this k times, this results in an overall running time of $\mathcal{O}\left(kcn\varepsilon^{-1}\log\left(cn\varepsilon^{-1}\right)\right)$, where the space used is $\mathcal{O}\left(cn\varepsilon^{-1}\log\left(cn\varepsilon^{-1}\right)\right)$.

Thus the overall running time is

$$\begin{aligned}
&\mathcal{O}\left(cn\varepsilon^{-1}\log\left(cn\varepsilon^{-1}\right)+\varepsilon^{-4}\log^2\left(\varepsilon^{-1}\right)n\log^2(n)+kcn\varepsilon^{-1}\left(\varepsilon^{-4}\log n\log\log n\right)\right)\\
=\ &\mathcal{O}\left(cn\varepsilon^{-1}\log\left(n\varepsilon^{-1}\right)+\varepsilon^{-5}n\log^2(n)+kcn\varepsilon^{-5}\log n\log\log n\right)\\
=\ &\mathcal{O}\left(kcn\varepsilon^{-5}\log^2\left(n\varepsilon^{-1}\right)\right).
\end{aligned}$$

The space follows directly from the space needed for the approximate distance data structure. All other data structures necessary for the algorithm use $\mathcal{O}(cn/\varepsilon\log(n))$ or $\mathcal{O}\left(\varepsilon^{-2}\right)$ space. Hence, the space is $\mathcal{O}\left(cn/\varepsilon\log(n)+n\varepsilon^{-4}\log^2\left(\varepsilon^{-1}\right)\right)$. □

The non-parametrized Shortcut Fréchet Distance

The algorithm presented and analyzed in Theorem 6.1.3 can be modified to also yield results for the non-parametrized shortcut Fréchet distance $\mathrm{d}_{\mathcal{S}}(\cdot,\cdot)$. The only two differences to the algorithm from Theorem 6.1.3 are that we do not store the subset of the (Δ,s)-reachable space reachable by *exactly* s proper shortcuts in every cell, but rather the union over all s, and instead of storing all left- and rightmost points at the *end* of each round in its own range tree we need to store them all in the same range tree and make them accessible immediately. The first is a straightforward modification by simply not keeping track of the number of proper shortcuts s instead passing over the Δ-free space only once. For the latter, we may use standard techniques such as the Bentley-Saxe method [BS80], increasing the query time by a logarithmic factor.

Theorem 6.5.13. *Let T and B be two c-packed polygonal curves in the plane with overall complexity n, together with values $0<\varepsilon\leq 1$ and $\Delta>0$. There exists an algorithm with running time in $\mathcal{O}\left(cn\varepsilon^{-5}\log^2\left(n\varepsilon^{-1}\right)\right)$ and space in $\mathcal{O}\left(cn\varepsilon^{-4}\log^2\left(\varepsilon^{-1}\right)\right)$ which outputs one of the following: (i) $d_{\mathcal{S}}(B,T)\leq(3+\varepsilon)\Delta$ or (ii) $d_{\mathcal{S}}(B,T)>\Delta$. In any case, the output is correct.*

Proof. This is an immediate consequence of the proof of Theorem 6.1.3 and the described modifications as well as the techniques from [BS80]. □

Chapter 7

Conclusion

In this thesis we explored algorithmic approaches for identifying and leveraging patterns in spatio-temporal data. We developed clustering algorithms for two distinct problem settings: detecting global patterns across a set of curves, and identifying local patterns of variable length within individual or small numbers of long curves. Both settings produce representative curves that summarize recurring behavior. Additionally, we proposed a data structure for classifying previously unseen curves based on a set of known representatives. Lastly, we examined a variant of the Fréchet distance aimed at filtering noise or irrelevant substructures from input curves.

In the following we discuss open problems and further research directions with respect to the individual problems discussed.

7.1 ε-Coresets for (k, ℓ)-Median under p-DTW

In Chapter 3 we discussed the (k, ℓ)-median problem under p-DTW. Our contributions involve investigating the VC dimension of range spaces characterized by arbitrarily small perturbations of DTW distances. While our results hold for a relaxed variant of the range spaces in question, they establish a robust link between numerous sampling results dependent on the VC dimension and DTW distances. Indeed, our first algorithmic contribution is the construction of coresets for (k, ℓ)-median through the sensitivity sampling framework presented in [FL11]. Apart from the VC dimension, the crux of adapting the sensitivity sampling framework to our (non-metric) setting was to use a known weak version of the triangle inequality satisfied by DTW. This inequality prompted us to further explore approximation algorithms by approximating DTW with a metric. By reducing to the metric case and plugging in our coresets, we designed an algorithm for the (k, ℓ)-median problem with running time linear in the number of the input sequences, and an approximation factor predominantly determined by our generalized iterated triangle inequality.

Although our primary motivation lies in constructing coresets, there are additional direct consequences through sampling bounds that establish a connection between the sample size and the VC dimension. For instance, suppose that we have a large set of time series following some unknown distribution, and we want to estimate the probability that a new time series falls within a given DTW ball b. Suppose that we also allow for small perturbations of the distances, i.e., we only want to guarantee that the estimated probability is realized by ***some*** small perturbations of the distances. This probability can be approximated within a constant additive error, by considering a random sample

of size depending solely on the VC dimension and the probability of success (over the random sampling) and measuring its intersection with b (see e.g., Theorem 3.3.6). Such an estimation can be used for example in anomaly detection, where one aims to detect time series with a small chance of occurring, or in time series segmentation, where diverse patterns may emerge throughout the series.

Future Direction 1 (Size-Independent ε-Coresets for (k, ℓ)-Median under p-DTW)**.** In our work, we constructed ε-coresets for the (k, ℓ)-median problem under p-DTW using sensitivity sampling. The resulting coreset size depends logarithmically on the number of input curves.

In recent work, [BCJ+22] showed how to construct ε-coresets with size independent of the input size for k-median-type problems in metric spaces, including the continuous Fréchet distance. Their method relies heavily on triangle inequality properties, which does not hold for p-DTW. This raises the question: Can we construct size-independent ε-coresets for (k, ℓ)-median under p-DTW? It is unclear whether such coresets can exist in the non-metric setting of DTW, and if so, whether new techniques beyond those in [BCJ+22] are required.

Future Direction 2 (Constant Factor Approximation Algorithm for (k, ℓ)-Median under p-DTW)**.** Based on our ε-coresets for (k, ℓ)-median under p-DTW, we obtained a linear-time $(\mathcal{O}(\ell m), 1)$-approximation algorithm. This relies on the polynomial-time $(\mathcal{O}(\ell m), 1)$-approximation algorithm we introduced. In fact, coresets can reduce the running time of any polynomial time algorithm to only have linear dependency in the number of curves.

However, no constant-factor (i.e., $(\mathcal{O}(1), 1)$) approximation algorithm for this problem is currently known. This leads to the open question: Does there exist a polynomial-time $(\mathcal{O}(1), 1)$-approximation algorithm for (k, ℓ)-median under p-DTW? Addressing this likely requires novel algorithmic ideas due to the non-metric nature of DTW.

7.2 Subtrajectory Covering and Coverage Maximization

In Chapter 4, we investigated the problem of clustering subtrajectories under the Fréchet distance, with a focus on approximation algorithms for the Subtrajectory Covering (SC) and Subtrajectory Coverage Maximization (SCM) problems. Rather than reducing the size of the smallest known candidate set from [vdHvdHO25], we analyzed structural aspects of the underlying greedy approach. Our main contributions include the introduction of the sweep-sequence and proxy coverage structures, which enabled the first deterministic cubic-time $(\mathcal{O}(\log n), O(1))$-approximation algorithm.

For the case of c-packed curves, we are able to improve the running time of the approximation algorithm for the SC problem to depend roughly quadratically on the input complexity. We observed further that a quartic version of the algorithm appears practical with real-world data, which suggests that the dependence in n for c-packed curves is not tight.

Future Direction 3 (Improved Algorithms for SC and SCM)**.** Our $(\mathcal{O}(\log n), \mathcal{O}(1))$-approximation algorithm for Subtrajectory Covering runs in cubic time, and roughly quadratic time for c-packed curves. However, the cubic running time—particularly with fractional exponents in the general case—suggests room for improvement. This raises several questions: Can we design an $(\mathcal{O}(\log n), \mathcal{O}(1))$-approximation algorithm for SC

with truly quadratic dependency on n? Furthermore, for c-packed curves, can the dependence on n be reduced to linear? The former question likely requires new algorithmic techniques beyond those explored here, while the latter may even be shown with existing techniques for c-packed curves [DHW12].

Future Direction 4 ((Conditional) Lower Bounds for SC and SCM)**.** Complementing the algorithmic side, it is natural to ask about computational hardness. [ABCD23] showed that computing exact solutions for SC or SCM is NP-hard. The situation for approximation and restricted versions appears more opaque.

A key question is: Can we establish approximation hardness for SC or SCM similar to that of set cover (e.g., hardness within $o(\log n)$)? How about if we consider the restricted version where, given a reference curve S and a curve P, one must compute the subcurve $\pi \subseteq S$ of complexity ℓ that maximizes $\lambda\left(\mathrm{Cov}_P(\pi, 4\Delta)\right)$, as is the case in our algorithm, where S is a maximal simplification of P? For a related problem, [GW22] presented a conditional lower bound stating that determining the 'largest' cluster takes at least *cubic* (in n) time. Our approximation algorithm based on the proxy coverage avoids such lower bounds by sidestepping exact computation of the optimal subcurve. A better understanding of these lower bounds could clarify whether our algorithms are close to optimal.

Future Direction 5 (Practicality of Cubic Time Algorithm)**.** Our empirical results suggest that the algorithm from [ABCD23], as well as our extension to centers of non-constant complexity, capture meaningful structure in trajectory data. However, our implementation does not yet incorporate all known optimizations (e.g., [vdHvdHO25]), nor our structural insights into the proxy coverage from Section 4.6–4.8.

This leads to the following question: To what extent can these improvements reduce the running time in practice, or are we approaching an algorithm that is theoretically efficient but suffers from large hidden constants? Understanding this trade-off would help bridge the gap between theory and application for the SC and SCM problems.

7.3 $(1+\varepsilon)$-ANN under the Continuous Fréchet Distance

In Chapter 5, we addressed approximate nearest neighbor (ANN) search for curves under the Fréchet distance, despite its inherently high complexity. We showed that while $\mathbb{X}^{d,k}$ has unbounded doubling dimension, there are spaces which are arbitrarily close to $\mathbb{X}^{d,k}$ with bounded doubling dimension which can be used for efficient ANN queries. Our method constructs ANN data structures in these neighboring spaces and extends naturally to special cases such as c-packed curves, yielding improved query times. This construction opens up questions about fully combinatorial ANN data structures and tighter dimensional bounds for restricted curve families.

Future Direction 6 (Combinatorial ANN Data Structure With Fast Queries)**.** Our approach to ANN for Fréchet distance in arbitrary dimension leverages an embedding into a space with bounded doubling dimension. However, this incurs a dependency on numerical parameters like the spread, either in preprocessing or query time. [CH23a] showed how to avoid such dependencies in a purely combinatorial structure, but their query time still depends on $\sqrt{n}$.

This motivates the question: Can we design a purely combinatorial ANN data structure for Fréchet distance with logarithmic query time (e.g., roughly $\mathcal{O}(\log n)$)? Such a

structure would eliminate numerical dependencies and could offer substantial improvements in practical settings.

Future Direction 7 (Doubling Dimension)**.** We presented both upper and lower bounds on the doubling dimension of the space of (μ, ε)-curves in $\mathbb{X}_\Lambda^{d,k}$. In particular, for c-packed curves, the gap between these bounds is small. However, neither of the bounds are tight.

Thus, a natural question is: Can the gap in the doubling dimension bound be further narrowed, or even closed? It seems plausible that the intrinsic dimension of the space is at least kd when $k = m$ and $d \in \mathcal{O}(\log nm)$. Any improvement in this direction would give new theoretical understanding of the structure of curve spaces.

7.4 Computing the k-Shortcut Fréchet Distance

In Chapter 6 we presented multiple results on the computability of the k-shortcut Fréchet distance. We gave an exact decision algorithm for the k-shortcut Fréchet distance and the non-parametrized Shortcut Fréchet distance with exponential running time. We complemented this result with a conditional lower bound showing that computing a decision of whether $\mathrm{d}_{\mathcal{S}}^k(P, Q) \leq \Delta$ requires $n^{o(k)}$ time for specific P, Q, and Δ. The main insight for the exact case being that the reachable space inside a cell may fragment into $n^{o(k)}$ pieces which we propagate individually to later cells. Circumventing this fragmentation, we presented a $(3+\varepsilon)$-approximate decision algorithm which exploited a critical insight given in [BDS14] which states that it suffices to propagate only a single point per cell. This results in an approximation algorithm with a running time roughly in $\mathcal{O}(kn^2)$, and $\mathcal{O}(n^2)$ for the non-parametrized version. In the special case of c-packed curves in the plane the running time improves to roughly $\mathcal{O}(kcn)$ and $\mathcal{O}(cn)$ respectively.

Future Direction 8 (Exact Computation)**.** We gave an exact decision algorithm for the k-shortcut Fréchet distance and its non-parametrized variant. Following the classical Fréchet distance setting, one might try to compute the distance by searching over a finite set of events, at which the decision algorithm for the k-shortcut Fréchet distance changes combinatorially.

However, unlike the classical case, where such events include point-to-point and point-to-edge distances, no such characterization is known for the shortcut variants. Such a set of candidate events ought to include the smallest distance Δ^* such that for a sequence of cells in the parametric space, tunnels from each cell to its subsequent cell of price Δ^* exist, such that each tunnel starts where the previous ends. This most likely depends on specific geometric configurations and it is not entirely clear how to compute such values Δ^*. Is it possible to identify a suitable set of events for the k-shortcut Fréchet distance or the non-parametrized Shortcut Fréchet distance to enable exact computation nonetheless? Without such a set, it remains unclear whether a fully combinatorial algorithm exists.

Future Direction 9 (Approximate Computation)**.** Similar to the exact case, our approximate decision algorithm could be extended to compute the distance if a suitable candidate set of events is known. For approximations, it suffices to search over a set containing at least one value that gives a constant-factor approximation.

This raises the question: Can we construct such a candidate set efficiently—ideally in $\widetilde{\mathcal{O}}(n^2)$ time—that enables approximate computation of the shortcut Fréchet distance? A similar logic to the exact case applies making it hard to geometrically describe and compute such a set.

Bibliography

[AB99] Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.

[ABCD23] Hugo Akitaya, Frederik Brüning, Erin Chambers, and Anne Driemel. Subtrajectory Clustering: Finding Set Covers for Set Systems of Subcurves. *Computing in Geometry and Topology*, 2(1):1:1–1:48, 2023.

[ABRU19] Hugo Akitaya, Maike Buchin, Leonie Ryvkin, and Jérôme Urhausen. The k-Fréchet Distance: How to Walk Your Dog While Teleporting. In *30th International Symposium on Algorithms and Computation*, volume 149, pages 50:1–50:15, 2019.

[ABS10] Marcel R. Ackermann, Johannes Blömer, and Christian Sohler. Clustering for Metric and Nonmetric Distance Measures. *ACM Transactions on Algorithms*, 6(4):59:1–59:26, 2010.

[AD18] Peyman Afshani and Anne Driemel. On the complexity of range searching among curves. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 898–917, 2018.

[AFK+15] Rinat Ben Avraham, Omrit Filtser, Haim Kaplan, Matthew J. Katz, and Micha Sharir. The Discrete and Semicontinuous Fréchet Distance with Shortcuts via Approximate Distance Counting and Selection. *ACM Transactions on Algorithms*, 11(4):29:1–29:29, 2015.

[AFM+18] Pankaj K. Agarwal, Kyle Fox, Kamesh Munagala, Abhinandan Nath, Jiangwei Pan, and Erin Taylor. Subtrajectory Clustering: Models and Algorithms. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 75–87, 2018.

[AG95] Helmut Alt and Michael Godau. Computing the Fréchet Distance between Two Polygonal Curves. *International Journal of Computational Geometry & Applications*, 5:75–91, 1995.

[AGK+04] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local Search Heuristics for k-Median and Facility Location Problems. *SIAM Journal on Computing*, 33(3):544–562, 2004.

[AHPK+06] Boris Aronov, Sariel Har-Peled, Christian Knauer, Yusu Wang, and Carola Wenk. Fréchet Distance for Curves, Revisited. In *Algorithms - ESA 2006*, pages 52–63, 2006.

[AYZ95] Noga Alon, Raphael Yuster, and Uri Zwick. Color-Coding. *Journal of the ACM*, 42(4):844–856, 1995.

[BBD+17] Kevin Buchin, Maike Buchin, David Duran, Brittany Terese Fasy, Roel Jacobs, Vera Sacristan, Rodrigo I. Silveira, Frank Staals, and Carola Wenk. Clustering Trajectories for Map Construction. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 14:1–14:10, 2017.

[BBG+11] Kevin Buchin, Maike Buchin, Joachim Gudmundsson, Maarten Löffler, and Jun Luo. Detecting Commuting Patterns by Clustering Subtrajectories. *International Journal of Computational Geometry & Applications*, 21(3):253–282, 2011.

[BBG+20] Kevin Buchin, Maike Buchin, Joachim Gudmundsson, Jorren Hendriks, Erfan Hosseini Sereshgi, Vera Sacristán, Rodrigo I. Silveira, Jorrick Sleijster, Frank Staals, and Carola Wenk. Improved Map Construction using Subtrajectory Clustering. In *LocalRec'20: Proceedings of the 4th ACM SIGSPATIAL Workshop on Location-Based Recommendations, Geosocial Networks, and Geoadvertising*, pages 5:1–5:4, 2020.

[BBK+20] 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 *SIGSPATIAL: 28th International Conference on Advances in Geographic Information Systems*, pages 99–110, 2020.

[BBMM17] Kevin Buchin, Maike Buchin, Wouter Meulemans, and Wolfgang Mulzer. Four Soviets Walk the Dog: Improved Bounds for Computing the Fréchet Distance. *Discrete & Computational Geometry*, 58(1):180–216, 2017.

[BBW09] Kevin Buchin, Maike Buchin, and Yusu Wang. Exact Algorithms for Partial Curve Matching via the Fréchet Distance. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 645–654, 2009.

[BCD22a] Frederik Brüning, Jacobus Conradi, and Anne Driemel. Faster Approximate Covering of Subcurves Under the Fréchet Distance. In *30th Annual European Symposium on Algorithms (ESA 2022)*, pages 28:1–28:16, 2022.

[BCD22b] Frederik Brüning, Jacobus Conradi, and Anne Driemel. Faster Approximate Covering of Subcurves Under the Fréchet Distance. *arXiv preprint arXiv:2204.09949*, 2022.

[BCD+25] Kevin Buchin, Jacobus Conradi, Lindsey Deryckere, Mart Hagedoorn, and Carolin Rehs. Tight Lower Bound for Approximating (k, ℓ)-Center Clustering under the Fréchet Distance. Presented at the *21st Spanish Meeting on Computational Geometry (EGC25)*, 2025.

[BCJ+22] Vladimir Braverman, Vincent Cohen-Addad, Shaofeng H.-C. Jiang, Robert Krauthgamer, Chris Schwiegelshohn, Mads Bech Toftrup, and Xuan Wu. The Power of Uniform Sampling for Coresets. In *63rd IEEE*

Annual Symposium on Foundations of Computer Science, pages 462–473, 2022.

[BCL+10] Prosenjit Bose, Sébastien Collette, Stefan Langerman, Anil Maheshwari, Pat Morin, and Michiel Smid. Sigma-local graphs. *Journal of Discrete Algorithms*, 8(1):15–23, 2010.

[BD23] Frederik Brüning and Anne Driemel. Simplified and Improved Bounds on the VC-Dimension for Elastic Distance Measures. *arXiv preprint arXiv:2308.05998*, 2023.

[BDG+19] 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 Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2922–2938, 2019.

[BDNP22] Karl Bringmann, Anne Driemel, André Nusser, and Ioannis Psarros. Tight Bounds for Approximate Near Neighbor Searching for Time Series under the Fréchet Distance. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 517–550, 2022.

[BDR23] Maike Buchin, Anne Driemel, and Dennis Rohde. Approximating (k, ℓ)-Median Clustering for Polygonal Curves. *ACM Transactions on Algorithms*, 19(1):4:1–4:32, 2023.

[BDS14] Maike Buchin, Anne Driemel, and Bettina Speckmann. Computing the Fréchet Distance with Shortcuts is NP-Hard. In *Proceedings of the Thirtieth Annual Symposium on Computational Geometry*, pages 367–376, 2014.

[BDS20] Kevin Buchin, Anne Driemel, and Martijn Struijs. On the Hardness of Computing an Average Curve. In *17th Scandinavian Symposium and Workshops on Algorithm Theory*, pages 19:1–19:19, 2020.

[BDvdLN19] Kevin Buchin, Anne Driemel, Natasja van de L'Isle, and André Nusser. klcluster: Center-based Clustering of Trajectories. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 496–499, 2019.

[BDvG+22] Maike Buchin, Anne Driemel, Koen van Greevenbroek, Ioannis Psarros, and Dennis Rohde. Approximating Length-Restricted Means Under Dynamic Time Warping. In *International Workshop on Approximation and Online Algorithms*, pages 225–253, 2022.

[BEHW89] Anselm Blumer, A. Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, 1989.

[BFL+16] Vladimir Braverman, Dan Feldman, Harry Lang, Adiel Statman, and Samson Zhou. New Frameworks for Offline and Streaming Coreset Constructions. *arXiv preprint arXiv:1612.00889*, 2016.

[BKK20] Maike Buchin, Bernhard Kilgus, and Andrea Kölzsch. Group Diagrams for Representing Trajectories. *International Journal of Geographical Information Science*, 34(12):2401–2433, 2020.

[BL97] Jeffrey S Beis and David G Lowe. Shape Indexing Using Approximate Nearest-Neighbour Search in High-Dimensional Spaces. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1000–1006. IEEE, 1997.

[BM16] Karl Bringmann and Wolfgang Mulzer. Approximability of the discrete Fréchet distance. *Journal of Computational Geometry*, 7(2):46–76, 2016.

[BO79] J. L. Bentley and T. A. Ottmann. Algorithms for Reporting and Counting Geometric Intersections. *IEEE Transactions on computers*, 28(9):643–647, 1979.

[BOS19] 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 Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2887–2901, 2019.

[BR22] Maike Buchin and Dennis Rohde. Coresets for (k, ℓ)-Median Clustering Under the Fréchet Distance. In *Algorithms and Discrete Applied Mathematics - 8th International Conference, CALDAM, Puducherry, India, February 10-12, Proceedings*, pages 167–180, 2022.

[Bri14] Karl Bringmann. Why Walking the Dog Takes Time: Frechet Distance Has No Strongly Subquadratic Algorithms Unless SETH Fails. In *55th IEEE Annual Symposium on Foundations of Computer Science*, pages 661–670, 2014.

[BS80] Jon Louis Bentley and James B Saxe. Decomposable searching problems I. Static-to-dynamic transformation. *Journal of Algorithms*, 1(4):301–358, 1980.

[BS06] Eckart Bindewald and Bruce A Shapiro. RNA secondary structure prediction from sequence alignments using a network of k-nearest neighbor classifiers. *RNA*, 12(3):342–352, 2006.

[CD22] Jacobus Conradi and Anne Driemel. On computing the k-shortcut fréchet distance. In *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, pages 46:1–46:20, 2022.

[CD23] Jacobus Conradi and Anne Driemel. Finding Complex Patterns in Trajectory Data via Geometric Set Cover. *arXiv preprint arXiv:2308.14865*, 2023.

[CD24] Jacobus Conradi and Anne Driemel. On Computing the k-Shortcut Fréchet Distance. *ACM Transactions on Algorithms*, 20(4):1–37, 2024.

[CD25a] Jacobus Conradi and Anne Driemel. Subtrajectory Clustering and Coverage Maximization in Cubic Time, or Better. In *33rd Annual European Symposium on Algorithms (ESA 2025)*, pages 12:1–12:18, 2025.

[CD25b] Jacobus Conradi and Anne Driemel. Subtrajectory Clustering and Coverage Maximization in Cubic Time, or Better. *arXiv preprint arXiv:2504.17381*, 2025.

[CDK24] Jacobus Conradi, Anne Driemel, and Benedikt Kolbe. $(1+\varepsilon)$-ANN Data Structure for Curves via Subspaces of Bounded Doubling Dimension. *Computing in Geometry and Topology*, 3(2):6:1–6:22, 2024.

[CH23a] Siu-Wing Cheng and Haoqiang Huang. Approximate Nearest Neighbor for Polygonal Curves Under Fréchet Distance. In *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*, pages 40:1–40:18, 2023.

[CH23b] Siu-Wing Cheng and Haoqiang Huang. Curve Simplification and Clustering under Fréchet Distance. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1414–1432, 2023.

[Che09] Ke Chen. On Coresets for k-Median and k-Means Clustering in Metric and Euclidean Spaces and Their Applications. *SIAM Journal on Computing*, 39(3):923–947, 2009.

[CKPR23] Jacobus Conradi, Benedikt Kolbe, Ioannis Psarros, and Dennis Rohde. Fast Approximations and Coresets for (k,ℓ)-Median under Dynamic Time Warping. *arXiv preprint arXiv:2312.09838*, 2023.

[CKPR24] Jacobus Conradi, Benedikt Kolbe, Ioannis Psarros, and Dennis Rohde. Fast Approximations and Coresets for (k,ℓ)-Median Under Dynamic Time Warping. In *40th International Symposium on Computational Geometry (SoCG 2024)*, pages 42:1–42:17, 2024.

[CS08] Teunis Cloete and Cornie Scheffer. Benchmarking of a full-body inertial motion capture system for clinical gait analysis. In *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4579–4582, 2008.

[dBCG13] Mark de Berg, Atlas F. Cook IV, and Joachim Gudmundsson. Fast Fréchet queries. *Computational Geometry*, 46(6):747–755, 2013.

[dBCvKO08] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 2008.

[DGM+14] Jean-Lou De Carufel, Amin Gheibi, Anil Maheshwari, Jörg-Rüdiger Sack, and Christian Scheffer. Similarity of polygonal curves in the presence of outliers. *Computational Geometry*, 47(5):625–641, 2014.

[DHP12] Anne Driemel and Sariel Har-Peled. Jaywalking Your Dog: Computing the Fréchet Distance with Shortcuts. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 318–337, 2012.

[DHW12] Anne Driemel, Sariel Har-Peled, and Carola Wenk. Approximating the Fréchet Distance for Realistic Curvesin Near Linear Time. *Discrete & Computational Geometry*, 48(1):94–127, 2012.

[DKS16] Anne Driemel, Amer Krivošija, and Christian Sohler. Clustering time series under the Fréchet distance. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 766–785, 2016.

[DNPP21] Anne Driemel, André Nusser, Jeff M Phillips, and Ioannis Psarros. The VC dimension of metric balls under Fréchet and Hausdorff distances. *Discrete & Computational Geometry*, 66(4):1351–1381, 2021.

[Eri05] Jeff Erickson. Local Polyhedra and Geometric Graphs. *Computational Geometry*, 31(1):101–125, 2005. Special Issue on the 19th Annual Symposium on Computational Geometry - SoCG 2003.

[FFK23] Arnold Filtser, Omrit Filtser, and Matthew J. Katz. Approximate Nearest Neighbor for Curves: Simple, Efficient, and Deterministic. *Algorithmica*, 85(5):1490–1519, 2023.

[FJ84] Greg N. Frederickson and Donald B. Johnson. Generalized Selection and Ranking: Sorted Matrices. *SIAM Journal on Computing*, 13(1):14–30, 1984.

[FL11] Dan Feldman and Michael Langberg. A Unified Framework for Approximating and Clustering Data. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, pages 569–578, 2011.

[FSS20] Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning Big Data Into Tiny Data: Constant-Size Coresets for k-Means, PCA, and Projective Clustering. *SIAM Journal on Computing*, 49(3):601–657, 2020.

[GHMS94] Leonidas Guibas, John Hershberger, Joseph Mitchell, and Jack Snoeyink. Approximating Polygons and Subdivisions with Minimum-Link Paths. *International Journal of Computational Geometry & Applications*, 3:151–162, 1994.

[GKL03] Anupam Gupta, Robert Krauthgamer, and James R Lee. Bounded geometries, fractals, and low-distortion embeddings. In *44th Annual IEEE Symposium on Foundations of Computer Science*, pages 534–543, 2003.

[GSW20] Joachim Gudmundsson, Yuan Sha, and Sampson Wong. Approximating the Packedness of Polygonal Curves. In *31st International Symposium on Algorithms and Computation*, volume 181, pages 9:1–9:15, 2020.

[GV14] Joachim Gudmundsson and Nacho Valladares. A GPU Approach to Subtrajectory Clustering Using the Fréchet Distance. *IEEE Transactions on Parallel and Distributed Systems*, 26(4):924–937, 2014.

[GvKS04] Joachim Gudmundsson, Marc van Kreveld, and Bettina Speckmann. Efficient Detection of Motion Patterns in Spatio-Temporal Data Sets. In *Proceedings of the 12th Annual ACM International Workshop on Geographic Information Systems*, pages 250–257, 2004.

[GW22] Joachim Gudmundsson and Sampson Wong. Cubic upper and lower bounds for subtrajectory clustering under the continuous Fréchet distance. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 173–189, 2022.

[HPIM12] Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani. Approximate Nearest Neighbor: Towards Removing the Curse of Dimensionality. *Theory of Computing*, 8(14):321–350, 2012.

[HPM06] Sariel Har-Peled and Manor Mendel. Fast Construction of Nets in Low-Dimensional Metrics and Their Applications. *SIAM Journal on Computing*, 35(5):1148–1184, 2006.

[HPS11] Sariel Har-Peled and Micha Sharir. Relative (p, ϵ)-Approximations in Geometry. *Discrete & Computational Geometry*, 45(3):462–496, 2011.

[Ind99] Piotr Indyk. Sublinear Time Algorithms for Metric Space Problems. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, pages 428–434, 1999.

[IP99] Russell Impagliazzo and Ramamohan Paturi. The Complexity of k-SAT. In *Proceedings of the Fourteenth Annual IEEE Conference on Computational Complexity*, pages 237–240, 1999.

[JJO11] Youngseon Jeong, Myong Kee Jeong, and Olufemi A. Omitaomu. Weighted dynamic time warping for time series classification. *Pattern Recognition*, 44(9):2231–2240, 2011.

[Joh73] David S Johnson. Approximation Algorithms for Combinatorial Problems. In *Proceedings of the Fifth Annual ACM Symposium on Theory of Computing*, pages 38–49, 1973.

[Kat16] Rohit J. Kate. Using dynamic time warping distances as features for improved time series classification. *Data Mining and Knowledge Discovery*, 30(2):283–312, 2016.

[KG14] Andreas Krause and Daniel Golovin. Submodular Function Maximization. *Tractability: Practical Approaches to Hard Problems*, 3(71-104):3, 2014.

[Kir83] David Kirkpatrick. Optimal Search in Planar Subdivisions. *SIAM Journal on Computing*, 12(1):28–35, 1983.

[KR02] David R. Karger and Matthias Ruhl. Finding Nearest Neighbors in Growth-Restricted Metrics. In *Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing*, pages 741–750, 2002.

[KS13] Aman Kataria and MD Singh. A Review of Data Classification Using K-Nearest Neighbour Algorithm. *International Journal of Emerging Technology and Advanced Engineering*, 3(6):354–360, 2013.

[KSS04] Amit Kumar, Yogish Sabharwal, and Sandeep Sen. A Simple Linear Time $(1 + \epsilon)$-Approximation Algorithm for k-Means Clustering in Any Dimensions. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 454–462, 2004.

[KSSF22] Vojtěch Kubelka, Brett K Sandercock, Tamás Székely, and Robert P Freckleton. Animal migration to northern latitudes: environmental changes and increasing threats. *Trends in Ecology & Evolution*, 37(1):30–41, 2022.

[KVW+17] Björn Krüger, Anna Vögele, Tobias Willig, Angela Yao, Reinhard Klein, and Andreas Weber. Efficient Unsupervised Temporal Segmentation of Motion Data. *IEEE Transactions on Multimedia*, 19(4):797–812, 2017.

[LC19] Rick Lumpkin and Luca Centurioni. Global Drifter Program quality-controlled 6-hour interpolated data from ocean surface drifting buoys. NOAA National Centers for Environmental Information, 2019.

[Lee91] Yuchun Lee. Handwritten Digit Recognition Using K Nearest-Neighbor, Radial-Basis Function, and Backpropagation Neural Networks. *Neural Computation*, 3(3):440–449, 1991.

[Lem09] Daniel Lemire. Faster Retrieval with a Two-Pass Dynamic-Time-Warping Lower Bound. *Pattern Recognition*, 42(9):2169–2180, 2009.

[Lov75] László Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13(4):383–390, 1975.

[LS10] Michael Langberg and Leonard J. Schulman. Universal ϵ-approximators for Integrals. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 598–607, 2010.

[LYW+24] Anqi Liang, Bin Yao, Bo Wang, Yinpei Liu, Zhida Chen, Jiong Xie, and Feifei Li. Sub-trajectory clustering with deep reinforcement learning. *The VLDB Journal*, 33(3):685–702, 2024.

[MGAM20] Jana Mattheus, Hans Grobler, and Adnan M. Abu-Mahfouz. A Review of Motion Segmentation: Approaches and Major Challenges. In *2020 2nd International Multidisciplinary Information Technology and Engineering Conference (IMITEC)*, pages 1–8, 2020.

[Mir23] Majid Mirzanezhad. On Approximate Near-Neighbors Search under the (Continuous) Fréchet Distance in Higher Dimensions. *Information Processing Letters*, page 106405, 2023.

[MoC07] C MoCap. Carnegie Mellon University Graphics Lab Motion Capture Database, 2007.

[MP69] Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. The MIT Press, 1969.

[MSSW18] Alexander Munteanu, Chris Schwiegelshohn, Christian Sohler, and David P. Woodruff. On Coresets for Logistic Regression. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems, NeurIPS*, pages 6562–6571, 2018.

[NWF78] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming*, 14(1):265–294, 1978.

[OBS94] Atsuyuki Okabe, Barry Boots, and Kokichi Sugihara. Nearest neighbourhood operations with generalized Voronoi diagrams: a review. *International Journal of Geographical Information Systems*, 8(1):43–71, 1994.

[Par13] Harold R. Parks. The Volume of the Unit n-Ball. *Mathematics Magazine*, 86(4):270–274, 2013.

[PFW+14] François Petitjean, Germain Forestier, Geoffrey I Webb, Ann E Nicholson, Yanping Chen, and Eamonn Keogh. Dynamic Time Warping Averaging of Time Series Allows Faster and More Accurate Classification. In *2014 IEEE International Conference on Data Mining*, pages 470–479, 2014.

[PFW+16] François Petitjean, Germain Forestier, Geoffrey I. Webb, Ann E. Nicholson, Yanping Chen, and Eamonn J. Keogh. Faster and more accurate classification of time series by exploiting a novel dynamic time warping averaging algorithm. *Knowledge and Information Systems*, 47(1):1–26, 2016.

[PW10] Mihai Pătraşcu and Ryan Williams. On the Possibility of Faster SAT Algorithms. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1065–1075, 2010.

[RAPJK+06] U Rajendra Acharya, K Paul Joseph, Natarajan Kannathal, Choo Min Lim, and Jasjit S Suri. Heart rate variability: a review. *Medical and Biological Engineering and Computing*, 44:1031–1051, 2006.

[RCM+13] Thanawin Rakthanmanon, Bilson J. L. Campana, Abdullah Mueen, Gustavo E. A. P. A. Batista, M. Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn J. Keogh. Addressing Big Data Time Series: Mining Trillions of Time Series Subsequences Under Dynamic Time Warping. *ACM Transactions on Knowledge Discovery from Data*, 7(3):10:1–10:31, 2013.

[SH75] Michael Ian Shamos and Dan Hoey. Closest-point problems. In *16th Annual Symposium on Foundations of Computer Science (sfcs 1975)*, pages 151–162, 1975.

[SS90] Jeanette P. Schmidt and Alan Siegel. The Spatial Complexity of Oblivious k-Probe Hash Functions. *SIAM Journal on Computing*, 19(5):775–786, 1990.

[SS22] Donald R. Sheehy and Siddharth S. Sheth. Nearly-Doubling Spaces of Persistence Diagrams. In *38th International Symposium on Computational Geometry (SoCG 2022)*, volume 224, pages 60:1–60:15, 2022.

[Ste74] Sherman K Stein. Two combinatorial covering theorems. *Journal of Combinatorial Theory, Series A*, 16(3):391–397, 1974.

[STS12] Mai Shouman, Tim Turner, and Rob Stocker. Applying k-Nearest Neighbour in Diagnosing Heart Disease Patients. *International Journal of Information and Education Technology*, 2(3):220–223, 2012.

[Tal04] Kunal Talwar. Bypassing the embedding: algorithms for low dimensional metrics. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*, pages 281–290, 2004.

[TLNH19] Tuan Minh Tran, Xuan-May Thi Le, Hien T. Nguyen, and Van-Nam Huynh. A novel non-parametric method for time series classification based on k-Nearest Neighbors and Dynamic Time Warping Barycenter Averaging. *Engineering Applications of Artificial Intelligence*, 78:173–185, 2019.

[vdHvdHO24] Ivor van der Hoog, Thijs van der Horst, and Tim Ophelders. Faster, Deterministic and Space Efficient Subtrajectory Clustering. *arXiv preprint arXiv:2402.13117*, 2024.

[vdHvdHO25] Ivor van der Hoog, Thijs van der Horst, and Tim Ophelders. Faster, Deterministic and Space Efficient Subtrajectory Clustering. In *52nd International Colloquium on Automata, Languages, and Programming (ICALP 2025)*, pages 133:1–133:18, 2025.

[WFH+16] Laura J Wilson, Christopher J Fulton, Andrew McC Hogg, Karen E Joyce, Ben TM Radford, and Ceridwen I Fraser. Climate-driven changes to ocean circulation and their inferred impacts on marine dispersal patterns. *Global Ecology and Biogeography*, 25(8):923–939, 2016.

[XW05] Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

[ZAFG21] Olga Zatsarynna, Yazan Abu Farha, and Juergen Gall. Multi-Modal Temporal Convolutional Network for Anticipating Actions in Egocentric Videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2249–2258, 2021.

[ZDlTH08] Feng Zhou, Fernando De la Torre, and Jessica K. Hodgins. Aligned Cluster Analysis for temporal segmentation of human motion. In *2008 8th IEEE International Conference on Automatic Face & Gesture Recognition*, pages 1–7, 2008.

[ZDlTH12] Feng Zhou, Fernando De la Torre, and Jessica K Hodgins. Hierarchical Aligned Cluster Analysis for Temporal Clustering of Human Motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(3):582–596, 2012.