# Subtrajectory Clustering, Curve Averaging and the Complexity of Underlying Range Spaces

**DISSERTATION**

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

**Frederik Sebastian Brüning**

aus

Langenfeld, Deutschland

Bonn 2024

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

| | |
|---|---|
| Gutachterin/Betreuerin: | Prof. Dr. Anne Driemel |
| Gutachter: | Prof. Dr. Heiko Röglin |

Tag der Promotion: 19.11.2024
Erscheinungsjahr: 2024

## Abstract

In this thesis, we study the clustering of spatial data with a focus on trajectory data. Trajectory data appears in various applications. These range from the recorded positions of moving objects (e.g. animals, humans, vehicles) to the change of measurements over time (e.g. biomarkers, electricity demand, temperature, sea level). A trajectory is usually modeled as a polygonal curve that is derived from the data by linear interpolation between consecutive observations. A clustering area that we are particularly interested in is subtrajectory clustering which consists of finding reoccurring patterns in trajectory data. We model subtrajectory clustering as a set cover problem and measure similarity based on the Fréchet distance. Given a polygonal curve with $n$ vertices, the goal is to find the smallest set of center curves of complexity $\ell$ such that each point on the input curve is part of a subcurve that has Fréchet distance of at most a given $\Delta$ to at least one of the center curves. We design bicriterial-approximation algorithms for this NP-hard problem. If there exists a solution of size $k$, then our algorithms find solutions of size $O(k\ell \log(k) \log(\ell))$ that solve the problem under distance $O(\Delta)$. The expected running time and space requirement of our algorithms is polynomial in $k$, $\ell$, $n$, $\frac{1}{\Delta}$ and the arclength of the input curve. Our approach uses a variation of the multiplicative weight update method on a simplified version of the problem.

The second clustering problem that we study is curve averaging: the problem of optimizing the center curve for a fixed set of curves. In particular, we study a widely used heuristic for curve averaging under the dynamic time warping (DTW) distance called the DTW Barycenter Averaging (DBA) algorithm. The algorithm is very similar to the popular $k$-means algorithm. Given an initial center curve, it alternates between assignment and update steps until convergence. We study the number of iterations that DBA performs until it converges to a local optimal solution. We assume that DBA is given $n$ polygonal curves of $m$ points in $\mathbb{R}^d$ and a parameter $k$ that specifies the length of the average curve to be computed. We conduct experiments that support the general view that DBA converges fast in practice. In contrast, we show that in the worst-case the number of iterations can be exponential in $k$. This gap between practical performance and worst-case analysis suggests that the worst-case behaviour is likely degenerate. To analyze the number of iterations on non-degenerate input, we further study DBA in the model of smoothed analysis. This model is based on bounding the expected number of iterations in the worst-case under random perturbation of the input. We achieve a bound that is polynomial in $k$, $n$ and $d$, and for constant $\frac{n}{d}$ is also polynomial in $m$.

Additionally, we study the complexity of range spaces underlying clustering problems where ranges are balls that are implicitly given by a center and a radius $\Delta$ and include all elements that are at a distance of at most $\Delta$ to the center. As distance measures, we consider Hausdorff distance, Fréchet distance and DTW. As centers and elements of the ground set, we consider polygonal curves in $\mathbb{R}^d$ and polygonal regions in $\mathbb{R}^2$. To measure the complexity, we bound the VC-dimension and the shattering dimension of the resulting range spaces. Our approach is based on splitting range queries for the considered range spaces into simple predicates that can be determined by sign values of polynomials. This enables us to bound the VC-dimension and shattering dimension based on the number of cells in the arrangement of zero sets of these polynomials.

## Acknowledgments

I would like to take this opportunity to thank all the people who have accompanied me on my scientific journey over the past years. Without your help, this thesis would not have been possible.

First and foremost, I thank my advisor Anne Driemel for her guidance and support throughout the years. She always had time to discuss research with me and contributed invaluable ideas and approaches to tackle the problems at hand. She encouraged me to attend workshops, conferences and other research visits, which allowed me to discuss my academic work with various members of the scientific community.

I also thank my co-authors Hugo Akitaya, Erin Chambers, Jacobus Conradi, Anne Driemel, Alperen Ergür and Heiko Röglin for their contributions and the fruitful discussions. In this context, I would also like to thank the reviewers of the preliminary versions of our papers for their helpful feedback.

Furthermore, I would like to thank Anne Driemel, Heiko Röglin, Reinhard Klein and Jan-Henrik Haunert for agreeing to be part of the doctoral committee. Thank you for your time and expertise.

Moreover, I am very grateful to my colleagues of department V. They not only created a pleasant work environment but also left me with numerous unforgettable memories from board game nights, bouldering sessions and other leisure activities. I especially thank my officemate Anurag Murty Naredla, who was always open to discussing research with me and would passionately explain to me any elegant technique he read about.

Finally, I would like to thank my family and friends for their love and encouragement and for always having my back. Special thanks go to Negar Shariat and Lubia Polo for their patience and unconditional support.

# Contents

# Chapter 1

# Introduction

The ongoing technological evolution enables the collection of large quantities of spatial data in various areas. This can be geospatial data from satellites, cell phone, ocean drifters, laser scanners, demographics archives and more [20, 93, 94] or other types of spatial data, e.g. from medical imaging [5, 111], atomic and molecular physics [134], computer simulations [105] or video games [54]. Analysis of spatial data can help to understand and address challenges of modern society like climate change [88, 113], epidemics [61], disease diagnosis [5, 111] or forecasting of natural disasters [135]. The overwhelming flood of data, however, often makes it difficult to extract relevant information from the data in a reasonable time (example in Figure 1.1). It is a common goal in modern computer science to extract relevant information in an automated fashion with the help of algorithms. The criteria that determine what constitutes relevant information can vary from application to application. Only the identification of these criteria and a rigorous mathematical formalization make it possible to evaluate algorithmic solutions to such problems in a quantitative fashion. The value of an algorithm is typically measured by its running time and the quality of its solution.

In this thesis, we deal with spatial problems in various ways. We develop algorithms to solve spatial problems, we analyze the running time and performance of algorithms to better understand their theoretical and practical behavior, and we analyze general properties of geometric structures underlying such spatial problems. We focus on problems where the data is given in the form of polygonal curves in trajectories data or polygonal regions in polygon data. Trajectory data usually consists of the recorded positions of moving objects and the analysis of this type of data is based on the underlying assumption that the inherent order of the measurements carries information. Types of trajectory data include human full-body-motion [85], analysis of hand gestures [110], analysis of the focal point of attention during eye tracking [58, 82] and traffic analysis [119, 122]. Polygonal regions are areas that are enclosed by an ordered sequence of points that are connected with line segments. Prominent examples are building footprints [77, 133], shapes of 2-dimensional objects [72] or zones and areas on a map like wildlife reserves [121], parking lots [116], glaciers [70] or districts [86].

A large field that deals with pattern recognition in spatial data is Clustering. In this field, the task is to partition data into classes according to a similarity measure. Each class is often additionally characterized by a representative, a so-called center. In the case of trajectory data, the sub-area of Clustering called Subtrajectory Clustering is concerned with capturing reoccurring (movement) patterns in long trajectories. One may think,

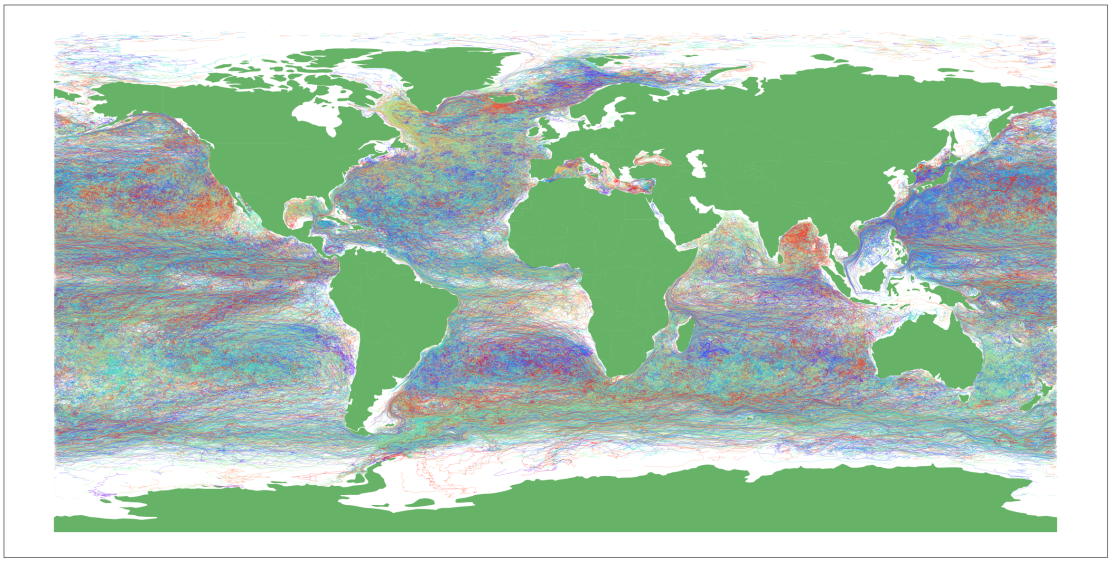

Figure 1.1: Depiction of ocean surface drifters from the NOAA Global Drifter Program [99]. Each drifter has a randomly assigned color. The overwhelming amount of data has not yet been processed to extract relevant information.

for example, about video, GPS or motion-tracking recordings of animals that one wants to classify based on their behavior [38]. Compared to the direct classification of data, subtrajectory clustering poses an additional challenge in finding the start and end points of the behavior patterns. Furthermore, it is usually not clear in advance which and how many different behaviors are to be expected. Since the requirements for good subtrajectory clustering can vary from application to application, various modeling approaches to the problem have been developed over time, see also the survey papers [40, 131, 136]. One of the first modeling questions is how to measure the similarity of two trajectories. For polygonal curves, various similarity measures have been introduced. These include Hausdorff distance, Dynamic time warping (DTW) distance and Fréchet distance, where the Hausdorff distance is also suitable for polygonal regions. We will describe the similarity measures in more detail in Section 2.3 and give a short overview here.

The Hausdorff distance is a classical bottleneck measure for point sets based on the maximum distance of any of the points to the closest point from the other set. The Fréchet distance is also a bottleneck measure, but in contrast to the Hausdorff distance, it takes the natural flow of the polygonal curves into account. It is based on the maximum distance of any two points of the curves under the best continuous monotone mapping between the curves. There is also a discrete variant of the Fréchet distance that only considers the vertices of the curves. For both the discrete Fréchet distance and the Hausdorff distance, there exists a variant that takes the average instead of the maximum distance. These are the DTW distance and the average Hausdorff distance. In this work, we develop algorithms for subtrajectory clustering under the Fréchet distance.

A natural subproblem of subtrajectory clustering is to find the appropriate representative for a given cluster of trajectories. This problem is also known as curve averaging. Depending on the application, curve averaging can be modeled in a wide variety of ways and there are various algorithmic approaches to solving it. One of the most common heuristics to solve it under DTW is probably the DTW Barycenter Averaging (DBA)

algorithm by Petitjean, Ketterlin and Gançarski [107]. We will analyze this algorithm with respect to its running time in more detail in Chapter 6. The algorithm is a prime example of an algorithm that has a much faster running time in practice than the theoretical worst-case analysis would suggest. This makes it suitable for analysis under random perturbation of its input. This type of analysis is called smoothed analysis. It goes back to Spielman and Teng [120] and enables to analyze running times of algorithms under realistic assumptions on the input, in which artificially constructed edge cases are avoided.

To approach the problems differently, we also consider clustering problems in terms of underlying geometric structures whose properties can be utilized. For example, if we take a center and the corresponding cluster, the cluster can be represented as a ball around the center whose radius is the distance of the center to the furthest curve in the cluster. The distance is measured in terms of the selected similarity measure. By using different centers and different radii, we can create balls that represent different clusters. Overall, we get a set of sets (also called range space) in which each set (range) is a ball. Techniques that we use in subtrajectory clustering are based on bounding the complexity of such and similar range spaces. A typical way to measure the complexity is to use the Vapnik-Chervonenkis dimension (VC-dimension) or the shattering dimension. The VC-dimension and shattering dimension are general concepts that are most prominently used to determine bounds on the number of samples needed for classification tasks in statistical learning theory [128] or for the construction of $\varepsilon$-nets [80] and $(\eta, \varepsilon)$-approximations [76] in computational geometry. We give an introduction to the VC-dimension, the shattering dimension and related techniques in Section 2.4. Using these techniques, we derive bounds for the VC-dimension and the shattering dimension of the above-described and similar range spaces in Chapter 3. In the following, we will introduce the topics that appear in this work in more detail and highlight the contributions.

## 1.1 VC-dimension and shattering dimension of elastic distance measures

The VC-dimension is a measure of complexity for range spaces that was introduced by Vapnik and Chervonenkis in their seminal paper [129]. A range space is a combination of a ground set (e.g. $\mathbb{R}^3$) and a set of subsets of this ground set (e.g. all balls or cubes in $\mathbb{R}^3$). These subsets are also called ranges. In the context of clustering, for example, one can imagine that the ranges represent possible clusters. Intuitively, the VC dimension measures how well any kind of classification of the points from the ground set into two classes (e.g. part of the cluster and not part of the cluster) can be represented by ranges of the corresponding range space. The shattering dimension is a related concept that measures the complexity of a range space in a similar way. See Section 2.4.1 for the specific differences. If the VC dimension or shattering dimension of a range space is known, it directly provides sample bounds for various applications that contain the range space as a basis. These applications can be found, for example, in statistical learning theory or computational geometry and range from the clustering applications investigated in this thesis to kernel density estimation [87], neural networks [14, 89], coresets [39, 48, 62, 63], object recognition [96, 97] and more.

The clustering applications that we are interested in have underlying range spaces consisting of balls with respect to a chosen similarity measure. Since the considered range spaces only consist of balls with respect to a similarity measure, they are very fundamental

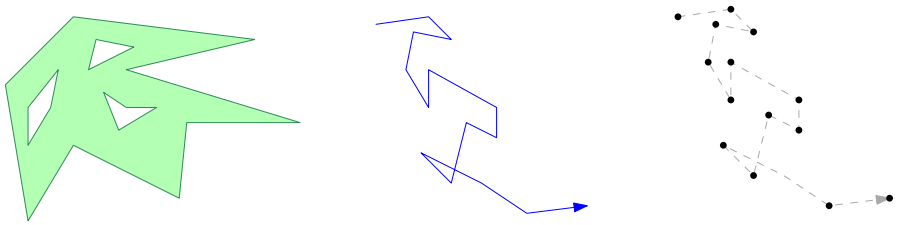

Figure 1.2: From left to right: Examples of a polygonal region with holes, the continuous variant of a polygonal curve and the discrete variant of a polygonal curve.

and their application is not limited to the area of clustering. Other applications include data structures, prediction, density estimation and classification (see also [43] and Chapter 10 of [57]). In our case, the similarity measures are the Hausdorff distance, the Fréchet distance and the dynamic time warping distance. As ground sets of these range spaces, we consider polygonal curves in $\mathbb{R}^d$ and polygonal regions in $\mathbb{R}^2$ that may contain holes. For polygonal curves, a distinction is also made between the discrete and the continuous variant, in which either only the vertices or the entire curve (including the points on the edges between the vertices) are taken into account. See also Figure 1.2 for examples of the elements in the ground sets. In the discrete setting with respect to the Hausdorff distance, polygonal curves are the same as general finite point sets.

**Results** The VC-dimension of balls around polygonal curves with respect to the Hausdorff distance and with respect to the Fréchet distance has previously been studied by Driemel, Nusser, Philips and Psarros [57]. We improve their upper bounds in all considered cases and obtain upper bounds that are asymptotically tight to their lower bounds in each of the considered parameters individually. We further extend our results by applying similar techniques to DTW for polygonal curves and to the Hausdorff distance for polygonal regions that may contain holes. A comparison of our results with the results of [57] is given in Table 1.1. Here, $m$ denotes the number of vertices of an element in the ground set and $k$ denotes the number of vertices of a center. In parallel and independently of us, Cheng and Huang [43] determined the same upper bounds as ours for the Fréchet distance of curves using largely the same techniques. For the shattering dimension, we use the same techniques and get slightly better bounds than the VC dimension bounds in all considered cases. The improvement is a factor of up to $(dk)^{-1}$ in the logarithm. See Table 3.2 in Chapter 3 for the exact bounds.

**Approach** The approaches used in our work and the related works are all based on the following idea. The complexity of a range space is related to how easy it is to decide if a given element of the ground set lies inside a given range. To answer if an element is inside of a ball one has to decide if the element is within a certain distance to the center of the ball. Such distance queries can often be split into simple geometric predicates that are easy to decide individually and together determine the answer to the query. Our approach lies in finding geometric predicates that are so simple that each can be decided by looking at the sign value of a single polynomial. Here, such polynomials may only depend on the radius, the coordinates of the center curve and the coordinates of the query curve. The VC-dimension and shattering dimension of the corresponding range

| | | new | ref. | Driemel et al. [57] |
|---|---|---|---|---|
| finite sets | av. Hausdorff | $O(dk\log(k^m m^k))$ | Thm. 3.2.4 | - |
| | Hausdorff | $O(dk\log(km))$ | Thm. 3.2.1 | $O(dk\log(dkm))$ |
| discrete polygonal curves | Fréchet | $O(dk\log(km))^{(*)}$ | Thm. 3.2.2 | |
| | DTW | $O(dk^2\log(m))$ | Thm. 3.2.3 | - |
| | | $O(dkm\log(k))$ | Thm. 3.2.3 | |
| continuous polygonal curves | Hausdorff | $O(dk\log(km))$ | Thm. 3.4.15 | $O(d^2k^2\log(dkm))$ |
| | Fréchet | $O(dk\log(km))^{(*)}$ | Thm. 3.4.16 | |
| | weak Fréchet | $O(dk\log(km))^{(*)}$ | Thm. 3.4.16 | $O(d^2k\log(dkm))$ |
| polygons $\mathbb{R}^2$ | Hausdorff | $O(k\log(km))$ | Thm. 3.4.15 | - |

Table 1.1: Overview of VC-dimension bounds with references. Results marked with $^{(*)}$ were independently obtained by Cheng and Huang [43].

space can then be bounded based on the number of cells in the arrangement of zero sets of these polynomials. The idea behind the last part of this approach is the following: Let us assume, we fix some elements of the ground set and want to split these elements into two classes in as many different ways as possible by using ranges of our range space. Each ball for which the polynomials have the same sign values splits the points in the same way. So counting the number of cells in the arrangement of zero sets gives a bound on the number of ways in which we can classify the points. The general idea behind this approach goes back to Goldberg and Jerrum [68, 69] and, independently, Ben-David and Lindenbaum [18]. We use, in particular, a theorem by Anthony and Bartlett [14] to bound the VC-dimension based on the number and degree of the polynomials that determine the inclusion in a range (see Theorem 2.4.10). Some of the predicates we use were developed by Driemel, Nusser, Philips and Psarros [57]. In their paper, they also decompose the distance query into predicates, but then restrict the VC dimension based on the number of operations needed to determine a predicate (Theorem 2.4.16), which leads to weaker results than the approach with sign values of polynomials.

**Bibliographical Notes** Chapter 3 is based on the work [27] by Frederik Brüning and Anne Driemel. The paper resulted from a dynamic developing process that both authors equally contributed to. The idea to improve and extend the VC-dimension bounds of [57] by the use of a different technique goes back to the author of the thesis. The detailed analysis was mainly carried out by the author of the thesis under the supervision and consultation of Anne Driemel.

## 1.2 Subtrajectory clustering

Subtrajectory clustering is concerned with capturing reoccurring movement patterns in trajectory data. As already mentioned, there are various ways of modeling the problem under the Fréchet distance. In one of the early works, Buchin, Buchin, Gudmundsson, Löffler and Luo [34] address the problem of finding the single most important cluster

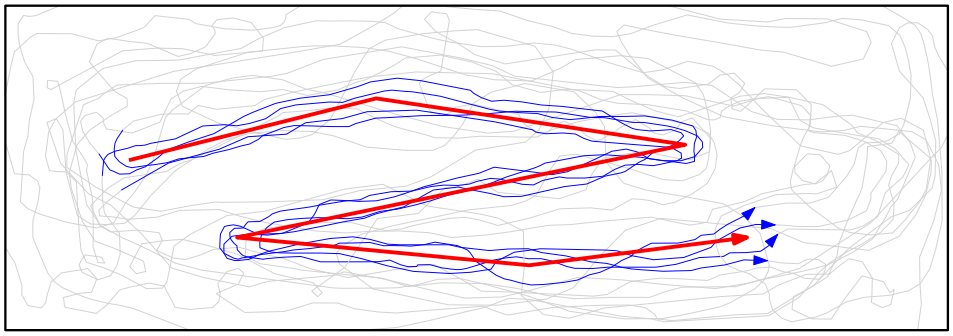

Figure 1.3: Example of a cluster in subtrajectory clustering: The red center curve represents the blue subtrajectories that have small Fréchet distance to the center curve. All other data points (grey) are not part of the cluster since they are not part of any subtrajectory that has small Fréchet distance to the center curve.

with respect to certain qualities. They consider length, number and distance of the subtrajectories inside a cluster and define optimization problems that optimize with regard to each of the parameters while keeping the other ones fixed. They show that the corresponding decision problems are NP-hard and tackle the problems with a sweep line approach that yields $(2-\varepsilon)$-approximation algorithms for the maximum length and the number of subtrajectories. In subsequent work, these ideas were also applied to the problem of reconstructing road maps from GPS data [32, 33]. In a different work, Buchin, Kilgus and Kölzsch [38] study the migration of animals with the idea of capturing the underlying movement patterns in the form of a graph. They build these graphs with the help of a greedy algorithm that iteratively computes the most significant cluster and afterwards removes it from the data. Another modeling approach by Agarwal, Fox, Munagala, Nath, Pan, and Taylor [3] considers subtrajectory clustering under the discrete Fréchet distance with respect to an objective function that is based on the metric facility location problem. Their objective is a weighted sum that takes the number of clusters and the similarity of subtrajectories to their assigned center curves into account and penalizes uncovered parts of the input trajectories. The authors show the NP-hardness of their problem and design approximation algorithms. For input trajectories with a total of $n$ vertices, they obtain a $O(\log^2 n)$-approximation.

In line with the above ideas of formalizing the objective function based on classical clustering problems, we study subtrajectory clustering under an objective function that is inspired by the $k$-center problem [71] and especially the $(k,\ell)$-clustering variant for trajectories, where the complexity of the centers is restricted by a parameter $\ell$ [56, 35, 104, 37]. In particular, we model subtrajectory clustering as a set cover problem where each set consists of all subtrajectories that can be represented by a given center curve. Each set is therefore a potential cluster. See Figure 1.3 for an example of one potential cluster in subtrajectory clustering.

**Problem definition** As input, we receive a long trajectory in the form of a polygonal curve of $n$ points in $\mathbb{R}^d$ where $d$ is assumed to be constant. Intuitively, we want to find the smallest possible set of representatives (center curves) such that the entire curve is represented. This means, that for each point on the curve, there should be a subcurve that contains this point and is similar to one of the selected center curves. Each point is, therefore, part of a behavior (cluster) that can also be represented by a center curve. To

| $\alpha$ | $\beta$ | Running time | Space | Reference |
|---|---|---|---|---|
| $O(\ell \log(k) \log(\ell))$ | 19 | $\widetilde{O}\left(k\ell^3 m^2 + mn\right)$ | $O(n + m\ell)$ | Thm. 4.1.3 |
| $O(\ell \log(k\ell))$ | 12 | $\widetilde{O}(nk^3\ell^3 \log^4(\frac{m}{k\ell}))$ | $\widetilde{O}(nk^2\ell^2 \log^2(\frac{m}{k\ell}))$ | Thm. 5.4.9 |

Table 1.2: Overview of results: For optimal center curves $C \subset (\mathbb{R}^d)^\ell$ of size $k$, covering $P \in (\mathbb{R}^d)^n$ under distance $\Delta$, we design bicriteria-approximation algorithms that compute a set $C' \subset (\mathbb{R}^d)^\ell$ of size $\alpha k$, covering $P$ under distance $\beta\Delta$. Here, we assume that $d$ is constant, $n$ is the complexity of $P$ and $m = \lceil \frac{\lambda}{\Delta} \rceil$ where $\lambda$ is the arclength of $P$.

measure how similar a subcurve is to a center curve, we use the Fréchet distance. An additional input parameter $\Delta$ determines how similar two curves must be to each other to be feasible representations of each other. This parameter $\Delta$ can be seen as a threshold for the radius of any cluster. Furthermore, we use another parameter $\ell$ to restrict how complex a center curve is allowed to be. There are several reasons for this. On the one hand, the aim of clustering is to summarize data in simple patterns to make them easier to interpret. On the other hand, this prevents overfitting of the data. To see this, let us assume that the center curves can be arbitrarily complex. Then the input curve could be chosen as a center to represent itself and would thus be a trivial optimal solution. This would obviously not be in line with our goal of finding simple patterns in the data.

**Results** If we formalize the problem as stated above, then it can be shown that this problem is NP-hard (see [8]). Since we still want to solve the problem in a reasonable time, we develop approximation algorithms (more precisely bicriteria-approximation-algorithms) for this problem. In our case, this means the following: If the optimal solution is a clustering of size $k$ for radius $\Delta$, then our algorithms return a clustering of size $\alpha k$ and radius $\beta\Delta$ for some $\alpha, \beta \in \mathbb{R}_+$. Let $m = \lceil \frac{\lambda}{\Delta} \rceil$ where $\lambda$ is the arclength of the input curve. In Chapter 4, we describe our initial approach that results in approximation factors $\alpha \in O(\ell \log(k) \log(\ell))$ and $\beta = 19$, an expected running time of $\widetilde{O}\left(k\ell^3 m^2 + mn\right)$ and a space requirement of $O(n + m\ell)$ where the $\widetilde{O}(\cdot)$ notation hides polylogarithmic factors in $n$ to simplify the exposition. In Chapter 5, we improve the dependency on the arclength of the curve and slightly improve the approximation factors $\alpha$ and $\beta$. We design an approximation algorithm with approximation factors $\alpha \in O(\ell \log(k\ell))$ and $\beta = 12$, an expected running time of $\widetilde{O}(n(k\ell)^3 \log^4(\frac{m}{k\ell}))$ and a space requirement of $\widetilde{O}(n(k\ell)^2 \log^2(\frac{m}{k\ell}))$. An overview of the results can be found in Table 1.2. Our techniques can also be applied in the case that the input is given by multiple polygonal curves. The only reason why we restrict ourselves to one polygonal curve is to simplify the exposition.

**Approach** The general approach for both algorithms is to replace the set cover instance by an easier set cover instance that still yields an approximate solution. The resulting set cover instance is then solved using the multiplicative weight update method [15, 45, 46]. In particular, we use a variant of this method by Anthony and Bartlett [23] that we adapt to our application. In Section 2.4.2, we describe our adaptation of the method in detail. On a high level, we apply an iterative sampling algorithm that works in the following way. In each iteration, the algorithm first samples a set of center curves as candidates for a solution. If the center curves are a solution and cover the whole input curve $P$ then the algorithm stops and returns this solution. Otherwise, the sample distribution is updated

to increase the probability to sample center curves that cover the currently uncovered parts of the curve $P$. In this step, the sample size might also increase, if there was no solution found in many iterations of the current sample size. The sample size for which the algorithm finds a solution depends on the VC-dimension of the range space underlying the set cover instance and the actual size of an optimal solution. The approaches for constructing an easier set cover instance with corresponding low VC-dimension differ between Chapter 4 and Chapter 5. We give a brief overview of the differences.

In Chapter 4, we first discretize the problem by distributing points evenly along the input curve. Here, the arclength of the part between two consecutive points along the curve depends on $\Delta$. We allow subcurves to start and end only at these so-called breakpoints and additionally restrict the vertices of eligible center curves to the positions of breakpoints. To identify if a point is covered by a center curve, we furthermore simplify the input curve locally and only test the distance to subcurves of the resulting simplification.

Our approach in Chapter 5 also discretizes the problem, but in contrast to Chapter 4, we first simplify the input curve globally and then distribute points evenly along the simplification. We then consider the problem of covering the simplifications with line segments instead of the general problem of covering the input curve with center curves of complexity $\ell$. This enables us to restrict our candidate space for possible center curves to subedges of the simplification that start and end in breakpoints. A careful choice of the global simplification lets us restrict the underlying range space even further. Instead of checking all possible subcurves that contain a point $p$ to determine if $p$ is covered by a center curve, we may restrict the range to only consider short subcurves of constant complexity.

**Bibliographical Notes** Chapter 4 is based on the work [8] by Hugo Akitaya, Frederik Brüning, Erin Chambers and Anne Driemel. The paper resulted from a dynamic developing process that all authors equally contributed to. The author of this thesis contributed by carrying out the technical parts of the majority of the proofs, specifying the auxiliary range space together with an approximate range space oracle and showing upper bounds on the VC-dimension of this range space to adapt the multiplicative weight update method efficiently. In addition to the results in Chapter 4, the work [8] also includes an NP-hardness proof of the underlying set cover problem.

Chapter 5 is based on the work [25] by Frederik Brüning, Jacobus Conradi and Anne Driemel. The general framework of the approach is based on the work [8]. The paper [25] resulted from a dynamic developing process that all authors equally contributed to. The author of this thesis contributed the development of the implicit weight update method and the bounds on the VC-dimension. Furthermore, the author of this thesis contributed the insight that restricting the center candidates to subedges of the simplification enables to only consider subedges of constant length when checking their coverage. The reduction of the general variant of the problem to the variant with line segments is also due to the author of this thesis. In addition to the results in Chapter 5, the work [25] contains another approximation algorithm based on restricting the candidate set even more to a set of $O(n^3)$ extremal subedges. This approach results in a purely combinatorial expected running time of $\widetilde{O}(k^2\ell^2 n + k\ell n^3)$, a space requirement of $\widetilde{O}(k\ell n + n^3)$ and approximation factors $\alpha$ in $O(\ell \log(k\ell))$ and $\beta = 11$. In [25], it was further shown how the developed techniques can be directly applied to the maximum coverage problem.

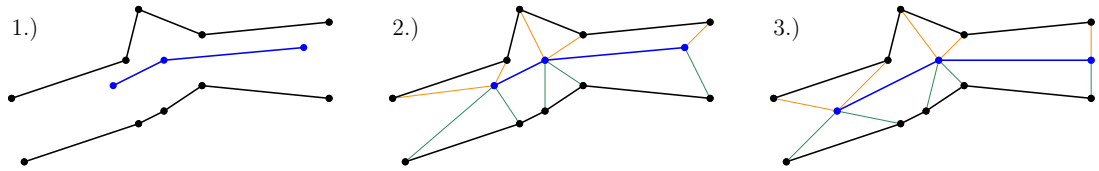


Figure 1.4: Example of the steps of the DBA algorithm. The input curves are depicted in black. The center curve is depicted in blue. 1.) Initialisation with an arbitrary center curve. 2.) Assignment step: Optimal assignments of the input curves to the center curves with respect to DTW are depicted in orange and green. 3.) Update step: The center points are moved to the means of the assigned input points.

## 1.3 Curve averaging

In curve averaging, the aim is to find a good representative (center curve) for a given set (cluster) of curves. Various quality measures (also known as objective functions) have been established in clustering to measure how well a center curve represents a cluster. The value of the objective function is usually referred to as the cost of the cluster. Traditional cost functions consider the distance between the center and the given curves by either taking the maximum distance (1-center), the sum of distances (1-median) or the sum of squared distances (1-means). These problems can be further generalized to clustering problems that consider multiple centers instead of one. There are various ways of combining the costs of the individual centers. The most common way is either to sum up the costs, which leads to $k$-min-sum-radii, $k$-median and $k$-means for $k$ individual centers or to take the maximum of the individual costs, which leads to $k$-center, min-max $k$-median and min-max $k$-means.

We focus on the 1-median problem under DTW. More precisely, we focus on a heuristic for this problem called DTW Barycenter Averaging (DBA), which was developed by Petitjean, Ketterlin and Gançarski [107] in 2011.

### 1.3.1 DTW Barycenter Averaging

The DTW Barycenter Averaging (DBA) algorithm is a curve-averaging algorithm used for the 1-median problem under DTW. Since DTW only considers the vertices of the polygonal curves, we consider point sequences instead of polygonal curves in this context. The objective is to find a center point sequence that minimizes the sum of the DTW distances to a given set of input curves. This Problem is NP-hard [36, 41] and to the best of our knowledge there are no efficient approximation algorithms known. DBA computes a local optimum for this problem. The structure of the DBA algorithm is very similar to the classical $k$-means algorithm by Lloyd [98]. Just as the $k$-means algorithm, it starts with an arbitrary initial solution and then alternates between assignment steps and update steps until convergence. See also Figure 1.4 for an illustration of the steps of the algorithm. In the assignment step, the algorithm computes the optimal assignment of the points in the input sequences to the current center sequence. Here, the optimal assignments are the assignments that realize the DTW distance. In the update step, the algorithm computes the mean of the assigned input points for each center point. The center sequence is then updated by moving the corresponding center points to these means.

The algorithm is very popular in practice and this despite the fact that there is no theoretical guarantee for the quality of its solution. It has for example been applied in the context of optimization of energy systems [124], forecasting of household electricity demand [123], and human activity recognition [117]. The popularity of DBA is probably due, among other things, to its fast running time on instances that occur in practice. We take a closer look at the number of iterations that DBA performs until convergence in Chapter 6. Convergence properties of the algorithm have been studied before [115], but our work [30] seems to be the first to rigorously study the number of iterations. Since the DBA algorithm is closely related to the $k$-means algorithm, we approach the analysis of the number of iterations with techniques that were successful for $k$-means. In the case of $k$-means, it has been shown that the number of iterations can be exponential in the worst-case [17, 75, 84, 130]. Since experiments and applications of $k$-means reported a very fast running time, it was studied in the model of smoothed analysis under random perturbation of the input [16, 101, 120]. Arthur, Manthey and Röglin [16] have shown that the expected number of iterations of $k$-means with respect to the perturbation is only polynomial in the number of points and in $\frac{1}{\sigma}$, where $\sigma$ is the standard deviation of the Gaussian perturbations. Following the same ideas, we analyze the number of iterations of DBA in the worst-case, in experiments and under perturbation of the input.

**Results** We assume that DBA is given $n$ point sequences of $m$ points in $\mathbb{R}^d$ each and an additional parameter $k$ that specifies the length of the center sequence to be computed. An overview of all our theoretical bounds on the number of iterations of DBA is given in Table 1.3. As a lower bound, we show that the number of iterations of DBA can be exponential in $k$ in the worst-case. This is already the case for instances with only two curves. To show this lower bound, we construct an instance of two point sequences with length $m = \Theta(k)$ in $\mathbb{R}^2$ such that DBA needs $2^{\Omega(k)}$ iterations to converge in Section 6.4. In contrast to this high theoretical lower bound, we only observe sublinear growth in our experiments (Section 6.5). This applies to the increase in the number of iterations with respect to the increase in each individual parameter $k, m$ and $n$ while the others are kept fixed. The observations suggest that the worst-case behaviour is likely degenerate and only holds for artificially constructed edge cases.

Concerning upper bounds, we describe two different bounds on the number of iterations in the worst-case in Section 6.2. The first one is an exponential upper bound of $6(4n)^{dk}\binom{m+k-2}{m-1}^{2dk}$ that can be found in Theorem 6.2.4. In addition to that, we have an upper bound that is based on geometric properties of the input data in Theorem 6.2.6. Intuitively speaking, one property bounds the magnitude of each input point by some parameter $B$ from above and another property bounds the similarity of any two center curves that appear at any step of the algorithm by another parameter $\varepsilon$ from below. The number of iterations is then bounded by $\frac{B(m+k)}{\varepsilon}$.

In the case of smoothed analysis, we study the number of iterations of DBA under Gaussian perturbation (with variance $\sigma^2$) of deterministic data. We show in Theorem 6.3.1 that the expected number of iterations is at most $\widetilde{O}\left(n^2 m^{8\frac{n}{d}+6} d^4 k^6 \sigma^{-2}\right)$, where the $\widetilde{O}(\cdot)$ notation hides polylogarithmic factors. Note that this bound is polynomial in $k$ for constant $n$ in contrast to the worst-case lower bound.

**Approach** Our instance for the lower bound in Section 6.4 is an adaptation of an instance of Vattani [130] for the $k$-means algorithm. The points from the instance

| Model | Type | Number of iterations | Reference |
|---|---|---|---|
| Worst-case analysis | Lower bound | $2^{\Omega(k)}$ | Thm. 6.4.3 |
| Worst-case analysis | Upper bound | $6(4n)^{dk}\binom{m+k-2}{m-1}^{2dk}$ | Thm. 6.2.4 |
| Worst-case analysis | Upper bound | $\frac{B(m+k)}{\varepsilon}$ | Thm. 6.2.6 |
| Smoothed analysis | Upper bound | $\widetilde{O}\left(n^2 m^{8\frac{n}{d}+6} d^4 k^6 \sigma^{-2}\right)$ | Thm. 6.3.1 |

Table 1.3: Overview of derived bounds on the number of iterations of the DBA algorithm.

of Vattani are duplicated multiple times and connected to point sequences such that the behaviour of the $k$-means algorithm can be reproduced by the DBA algorithm. A challenge here is to get an assignment that is similar to the $k$-means assignment and still respects the ordering of the DBA algorithm.

For our experiments, we take the data set of time series from the M5 Competition [100]. The data set consists of unit sales of products from Walmart stores in the USA. We create DBA instances either based on natural splits of the data (e.g. product departments) or via sub-sampling. As an initialization method, we derive the first center sequence from a random assignment given by a combination of random walks.

The first upper bound in the worst-case analysis (Theorem 6.2.4) uses similar techniques as our VC-dimension bounds in Chapter 3. We bound the number of iterations based on the cells of the arrangement of zero sets of specific polynomials. The polynomials correspond to differences in the dynamic time warping distance of center and input curve if we calculate this distance based on two fixed assignments instead of the optimal assignments. The second upper bound (Theorem 6.2.6) uses a potential function argument that depends on geometric properties of the input.

The techniques, we use for smoothed analysis include anti-concentration estimates and standard tail bounds for the norm of a random vector. Intuitively, we bound the probability that the geometric properties in Theorem 6.2.6 hold for our perturbed instances.

**Bibliographical Notes** Chapter 6 is based on the work [30] by Frederik Brüning, Anne Driemel, Alperen Ergür and Heiko Röglin. The paper resulted from a dynamic developing process that all authors equally contributed to. The author of this thesis contributed the initial version of the derivation of the upper bound in the case of smoothed analysis. The lower bound and the execution of the experiments were also contributed by the author of this thesis.

# Chapter 2

# Basic notation, concepts and techniques

In this chapter, we introduce basic notation and definitions that are used throughout the thesis and explain basic concepts and techniques underlying our research. The introduction of concepts and techniques is divided into three parts. In the first part, we deal with distance measures including the Hausdorff, dynamic time warping and Fréchet distance measures for polygonal curves and regions. In this context, we introduce Voronoi diagrams and the free space diagram as useful tools for these distance measures. In the second part, we deal with range spaces and explain related concepts including VC-dimenson, shattering dimension, epsilon-nets, hitting sets and set covers. We deal with questions like how to bound the VC-dimension, which sample bounds can be derived for epsilon-nets and how these sample bounds can be used to solve hitting set problems or set cover problems. In the third part, we give an introduction to the subtrajectory clustering problem that we study in Chapter 4 and 5. In order to be able to talk about all of this in a mathematical formal way, we first introduce basic notation for polygonal curves and regions.

## 2.1 General notation

When stating asymptotic bounds, we may use the $\widetilde{O}(\cdot)$ notation hiding polylogarithmic factors to simplify the exposition. For $n \in \mathbb{N}$, we define $[n]$ as the set $\{1, \ldots, n\}$. We call an ordered sequence of points $p_1, \ldots, p_n$ in $\mathbb{R}^d$ a ***point sequence*** of length $n$. For any $n > 1$, a point sequence $p_1, \ldots, p_n \in \mathbb{R}^d$ defines a ***polygonal curve*** $P$ by linearly interpolating consecutive points, that is, for each $i$, we obtain the ***edge*** $e_i : [0,1] \to \mathbb{R}^d; t \mapsto (1-t)p_i + tp_{i+1}$. We may write $e_i = \overline{p_i\, p_{i+1}}$ for edges and $E(P)$ for the set of all edges of $P$. We may think $P$ as a continuous function $P : [0,1] \to \mathbb{R}^d$ by fixing $n$ values $0 = t_1 < \ldots < t_n = 1$, and defining $P(t) = \lambda p_{i+1} + (1-\lambda)p_i$ where $\lambda = \frac{t-t_i}{t_{i+1}-t_i}$ for $t_i \leq t \leq t_{i+1}$. We call the set $(t_1, \ldots, t_n)$ the ***vertex parameters*** of the parametrized curve $P : [0,1] \to \mathbb{R}^d$. For $n = 1$, we may slightly abuse notation to view a point $p_1$ in $\mathbb{R}^d$ as a polygonal curve defined by an edge of length zero with $p_2 = p_1$. We call the number of vertices $n$ the ***complexity*** of the curve. Let $\mathbb{X}_n^d = (\mathbb{R}^d)^n$, and think of the elements of this set as the set of all polygonal curves of $n$ vertices in $\mathbb{R}^d$.

We define the ***concatenation*** of two curves $P, Q : [0,1] \to \mathbb{R}^d$ with $P(1) = Q(0)$ by $P \oplus Q : [0,1] \to \mathbb{R}^d$ with $(P \oplus Q)(t) = P(2t)$ if $t \leq 1/2$, and $(P \oplus Q)(t) = Q(2t-1)$ if

$t \geq 1/2$. Note that the concatenation of two polygonal curves $P$ and $Q$ with vertices $p_1, \ldots, p_n$ and $q_1, \ldots, q_m$ such that $p_n = q_1$ is the polygonal curve defined by the vertices $p_1, \ldots, p_n, q_2, \ldots, q_m$. For any two $a, b \in [0, 1]$ we denote with $P[a, b]$ the ***subcurve*** of $P$ that starts at $P(a)$ and ends at $P(b)$. Note, that $a > b$ is specifically allowed and results in a subcurve in reverse direction. We call the subcurves of edges ***subedges***.

We call $P$ a ***closed curve*** if $p_1 = p_m$ and we call $P$ ***self-intersecting*** if there exist $s \in [0, 1]$, $t \in (0, 1)$ with $s \neq t$ such that $P(s) = P(t)$. In the case that $P$ is a closed curve in $\mathbb{R}^2$ which is not self-intersecting, we call the union of $P$ with its interior a ***simple polygonal region*** $S$ (without holes). We denote with $\partial S$ the boundary of $S$, which is $P$. Given a simple polygonal region $S_0$ and a set of pairwise disjoint simple polygonal regions $S_1, \ldots, S_h$ in the interior of $S_0$, we also consider the set $S = S_0 - \{S_1 \cup \cdots \cup S_h\}$ a polygonal region and we call $S_1, \ldots, S_h$ the ***holes*** of $S$.

## 2.2 Computational Model

Throughout the thesis, we use the following computational model. We describe our algorithms in the real-RAM model of computation, which allows to store real numbers and to perform simple operations in constant time on them. We call the following operations ***simple operations***. The arithmetic operations $+, -, \times, /$. The comparison operations $=, \neq, >, \geq, \leq, <$, for real numbers with output 0 or 1. In addition to the simple operations, we allow the square-root operation.

## 2.3 Distance measures

Over time, various distance measures for polygonal curves and regions have been established [53]. In this section, we will introduce some of the most popular ones in computational geometry and highlight their properties. In particular, we will introduce variations of the Hausdorff distance, the Fréchet distance and the dynamic time warping (DTW) distance. First, we give some background on the distance measures.

The Hausdorff distance is sometimes also referred to as Pompeiu-Hausdorff distance and was introduced by Pompeiu as the set distance [109]. The distance was later studied and popularized by Hausdorff [78, 79]. The Hausdorff distance and its variations have since been used in various applications including shape matching [12, 11], shape morphing [127, 51], image comparison [83] and medical image segmentation [9, 73, 81]. The Fréchet distance was originally defined by Fréchet [65] as a measure of similarity between two parametric curves that takes the flow of the curves into account. Its discrete variant goes back to Eiter and Mannila [59]. The discrete and continuous Fréchet distance has been used for all kinds of trajectory data from movement patterns of migrating animals [38] over tracking of cars [32, 33] and ships [67] to analysis of tropical cyclones [92]. The DTW distance was introduced by Sakoe and Chiba [112] in the 1970s to capture similarities of time series in speech recognition and just like the Fréchet distance it belongs to the family of elastic distance measures. It was later rediscovered by Berndt and Clifford [19] and popularized by Keogh and Ratanamahatana [90]. DTW is often used as a baseline comparison for time series classification and has been used in areas like signature verification [118], touch screen authentication [52], gait analysis [66] and classification of surgical processes [64] to name a few.

All these distance measures can be applied to any metric space, but for our applications, we restrict ourselves to their use in the Euclidean space $\mathbb{R}^d$ for $d \in \mathbb{N}$. For two points

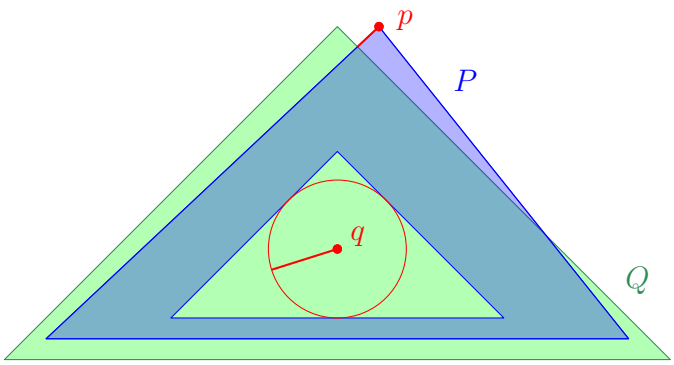


Figure 2.1: Illustration of the Hausdorff distance between two polygonal regions $P$ and $Q$. The directed Hausdorff distances $d_{\overrightarrow{H}}(P,Q)$, $d_{\overrightarrow{H}}(Q,P)$ are realized at $p \in P$ and $q \in Q$, while the Hausdorff distance $d_H(P,Q) = \max\{d_{\overrightarrow{H}}(p,Q), d_{\overrightarrow{H}}(q,P)\}$ is realized at $q$.

$p, q$ in $\mathbb{R}^d$, we denote with $\|p-q\|$ the ***Euclidean distance***, where $\|.\|$ is the standard Euclidean norm. Even though, we only use it for polygonal curves and regions, the Hausdorff distance is a bottleneck distance measure for any pair of sets in $\mathbb{R}^d$. It is a limit for the largest distance that any point of the two sets has to its closest point in the other set. Formally, the ***directed Hausdorff distance*** from $X \subseteq \mathbb{R}^d$ to $Y \subseteq \mathbb{R}^d$ is defined as

$$d_{\overrightarrow{H}}(X,Y) = \sup_{x \in X} \inf_{y \in Y} \|x-y\|$$

and the ***Hausdorff distance*** between $X$ and $Y$ is defined as

$$d_H(X,Y) = \max\{d_{\overrightarrow{H}}(X,Y), d_{\overrightarrow{H}}(Y,X)\}.$$

If a set $X$ consists of a single point $p \in \mathbb{R}^d$, we may write $p$ instead of $\{p\}$ to simplify the notation, e.g. $d_H(p,Y)$ instead of $d_H(\{p\},Y)$. It is also possible to take an average instead of the supremum resulting in the average Hausdorff distance. We introduce it here with respect to the squared Euclidean distance for the discrete case where the sets $X$ and $Y$ have finitely many points. The ***average Hausdorff distance*** between $X$ and $Y$ is defined as

$$d_{aH}(X,Y) = \frac{1}{2}\left(\frac{1}{|X|}\sum_{x \in X}\min_{y \in Y}\|x-y\|^2 + \frac{1}{|Y|}\sum_{y \in Y}\min_{x \in X}\|x-y\|^2\right).$$

Polygonal curves always come with an innate order of their vertices. This order carries information that is not considered by the Hausdorff distance at all. If two curves have the same complexity, one could argue that you can easily compare them, by mapping the points in order one by one to the respective points of the respective other curve and measuring their distance. This might be valid for some applications where it plays a crucial role to compare events that happen at the same time. But there are also many applications where the curves have different lengths, measurements are taken within different time intervals or not even in consistent time intervals or movement patterns can be identified as similar even if they happen at different speeds or with different acceleration and deceleration. For these applications, distance measures have evolved that take the flow of the curves into account and are robust to compression and stretching

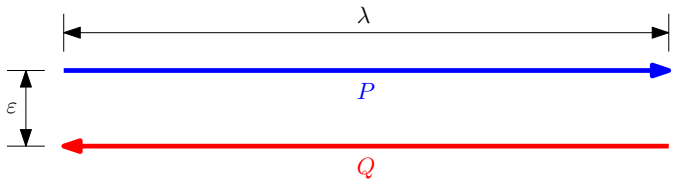


Figure 2.2: Example of two line segments $P, Q$ with Hausdorff distance $d_H(P,Q) = \varepsilon$ and Fréchet distance $d_F(P,Q) = \sqrt{\varepsilon^2 + \lambda^2}$. Note that the distance is the same for the weak and the discrete Fréchet distance. For $\lambda \gg \varepsilon$ we have $d_F(P,Q) \gg d_H(P,Q)$.

of time. They can again be divided into bottleneck distance measures (all variants of the Fréchet distance) and distance measures based on averaging like the dynamic time warping distance. For an easy example where taking the ordering of the vertices into account makes a huge difference, see the comparison of the Hausdorff distance and the Fréchet distance in Figure 2.2. In the discrete case, where we only consider the vertices of the curves, the compression and stretching of time is realized by so-called warping paths. For $m_1, m_2 \in \mathbb{N}$, each sequence $(1,1) = (i_1, j_1), (i_2, j_2), \ldots, (i_M, j_M) = (m_1, m_2)$ such that $i_k - i_{k-1}$ and $j_k - j_{k-1}$ are either 0 or 1 for all $k$ is a ***warping path*** from $(1,1)$ to $(m_1, m_2)$. We denote with $\mathcal{W}_{m_1,m_2}$ the set of all warping paths from $(1,1)$ to $(m_1, m_2)$. For any two polygonal curves $P \in \mathbb{X}^d_{m_1}$ with vertices $p_1, \ldots, p_{m_1}$ and $Q \in \mathbb{X}^d_{m_2}$ with vertices $q_1, \ldots, q_{m_2}$, we also write $\mathcal{W}_{P,Q} = \mathcal{W}_{m_1,m_2}$, and call elements of $\mathcal{W}_{P,Q}$ warping paths between $P$ and $Q$. The ***dynamic time warping distance*** (DTW) between $P$ and $Q$ is defined as

$$d_{DTW}(P,Q) = \min_{w \in \mathcal{W}_{P,Q}} \sum_{(i,j) \in w} \|p_i - q_j\|^2.$$

A warping path that attains the above minimum is also called an ***optimal warping path*** between $P$ and $Q$. We denote with $\mathcal{W}^*_{m_1,m_2} \subset \mathcal{W}_{m_1,m_2}$ the set of warping paths $w$ such that there exist polygonal curves $P \in \mathbb{X}^d_{m_1}$ and $Q \in \mathbb{X}^d_{m_2}$ with this optimum warping path $w$. The ***discrete Fréchet distance*** of two polygonal curves $P \in \mathbb{X}^d_{m_1}$ with vertices $p_1, \ldots, p_{m_1}$ and $Q \in \mathbb{X}^d_{m_2}$ with vertices $q_1, \ldots, q_{m_2}$ is defined as

$$d_{dF}(P,Q) = \min_{w \in \mathcal{W}_{P,Q}} \max_{(i,j) \in w} \|p_i - q_j\|.$$

The dynamic time warping distance and discrete Fréchet distance can also be seen as distance measures of point sequences instead of polygonal curves because they only depend on the vertices of the polygonal curves.

In the continuous case, we define the ***Fréchet distance*** of $P$ and $Q$ as

$$d_F(P,Q) = \inf_{\alpha,\beta:[0,1]\to[0,1]} \sup_{t \in [0,1]} \|P(\alpha(t)) - Q(\beta(t))\|,$$

where $\alpha$ and $\beta$ range over all functions that are non-decreasing, surjective and therefore continuous. We further define their ***weak Fréchet distance*** as

$$d_{wF}(P,Q) = \inf_{\alpha,\beta:[0,1]\to[0,1]} \sup_{t \in [0,1]} \|P(\alpha(t)) - Q(\beta(t))\|,$$

where $\alpha$ and $\beta$ range over all continuous functions with $\alpha(0) = \beta(0) = 0$ and $\alpha(1) = \beta(1) = 1$. For the weak Fréchet distance and the Fréchet distance, we call the pair $(\alpha, \beta)$

a ***traversal***. Every traversal has a distance $\sup_{t\in[0,1]} \|P(\alpha(t)) - Q(\beta(t))\|$ associated to it.

Intuitively you can think of the Fréchet distance as the minimal length needed such that a dog and a dog walker can traverse along the two curves from start to finish. They may vary in speed but never (except for weak Fréchet) walk backward. When deciding which of the introduced distance measures to use, you have to choose between bottleneck distance measures like the Hausdorff distance and all variants of the Fréchet distance or distance measures that are based on averaging like the average Hausdorff distance and the dynamic time warping distance.

Bottleneck distance measures are more sensitive to outliers but usually have a clearer and simpler geometric interpretation. In our cases, they all fulfill the triangle inequality where their counterparts based on averaging do not. The introduced bottleneck distance measures are all, in fact, even pseudo-metrics, which opens up the use of established tools and techniques for pseudo-metrics. Since the distance measures based on averaging consider quantities with respect to all data points instead of just an extreme distance, they offer more statistical insight into the overall behavior of the data.

In the following, we introduce the Voronoi diagram and the free space diagram. Both are concepts that capture structure induced by and underlying the distance measures. While the Voronoi diagram has its applications for all distance measures, the free space diagram is most prominent for its use in algorithms to compute the Fréchet distance of two curves.

### 2.3.1 Voronoi diagram

In clustering and classification, a recurring challenge is to find the best representative for a query data point from a set of possible representatives. The most typical way of quantifying the value of a representative is to measure its distance to the query point. The Voronoi diagram captures the essence of this idea by dividing the space into regions, such that each region corresponds to the points that are closest to a given representative.

While the Voronoi diagram can easily be defined for general spaces and distance measures, we define it here only for the case that the query points are in $\mathbb{R}^2$, the representatives are sets in $\mathbb{R}^2$ and the distance measure is the Hausdorff distance. This is the single case, where Voronoi diagrams are used in this work (see Chapter 3) and we can avoid notational overhead by restricting ourselves to this case. The sketch of the Voronoi diagram of 6 line segments is given in Figure 2.3. Let $X$ be a set of subsets (called sites) of $\mathbb{R}^2$. The ***Voronoi region*** $reg(A)$ of a site $A \in X$ consists of all points $p \in \mathbb{R}^2$ for which $A$ is the closest among all sites in $X$, i.e.

$$reg(A) = \{p \in \mathbb{R}^2 \mid d_{\overrightarrow{H}}(p, A) < d_{\overrightarrow{H}}(p, B) \text{ for all } B \in X \setminus \{A\}\}.$$

The ***Voronoi diagram*** is the complement of the union of all regions $reg(A)$ with $A \in X$, so

$$vd(X) = \mathbb{R}^2 \setminus \cup_{A\in X} reg(A).$$

For any two sites $A, B \in X$, we call the set

$$bisec(A, B) = \{p \in \mathbb{R}^2 \mid d_{\overrightarrow{H}}(p, A) = d_{\overrightarrow{H}}(p, B)\}$$

the ***bisector*** of $A$ and $B$. The ***Voronoi edge*** of $A, B \in X$ is defined as

$$ve(A, B) = vd(X) \cap bisec(A, B)$$

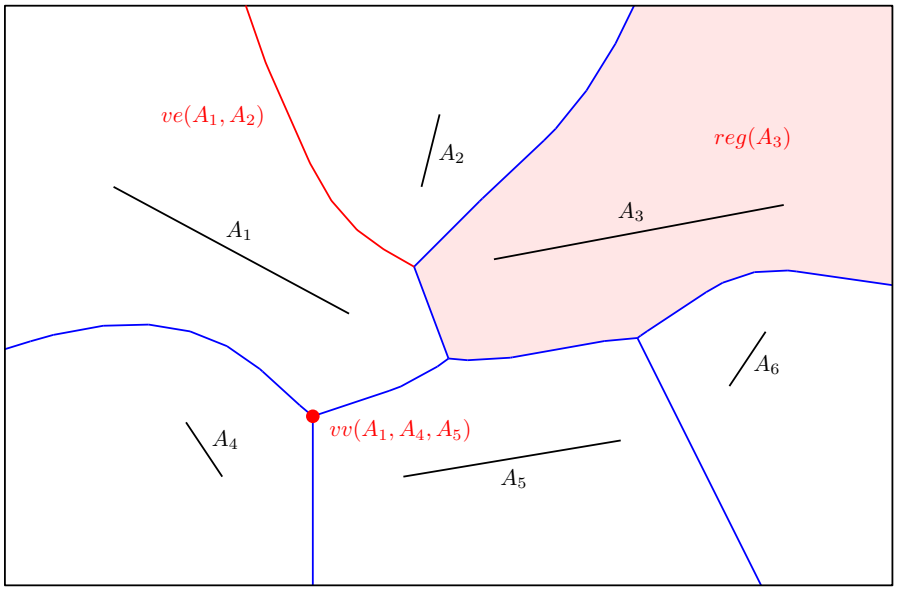


Figure 2.3: Sketch of the Voronoi diagram of 6 line segments including examples for a Voronoi region, a Voronoi edge and a Voronoi vertex.

and the ***Voronoi vertices*** of $A, B, C \in X$ are defined as

$$vv(A, B, C) = vd(X) \cap bisec(A, B) \cap bisec(B, C).$$

### 2.3.2 Free space diagram

The free space diagram which was first introduced by Alt and Godau [13] in an algorithm for computing the Fréchet distance of two polygonal curves. It is a level set of the Euclidean distance between the points of the two curves and enables to answer a Fréchet distance query between these curves.

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

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

*This is the set of points in the parametric space, whose corresponding points on $P$ and $Q$ are at a distance at most $\Delta$. The edges of $P$ and $Q$ segment the free space into cells. We call the intersection of $\mathcal{D}_\Delta(P, Q)$ with the boundary of cells the **free space intervals**.*

See Figure 2.4 for an illustration of the free space diagram. Alt and Godau [13] showed that the $\Delta$-free space inside any cell is convex and has constant complexity. More precisely, it is an ellipse intersected with the cell. Furthermore, the Fréchet distance between two curves is at most $\Delta$ if and only if there exists a continuous path $\pi : [0, 1] \to \mathcal{D}_\Delta(P, Q)$ that starts at $(0, 0)$, ends in $(1, 1)$ and is monotone in both coordinates.

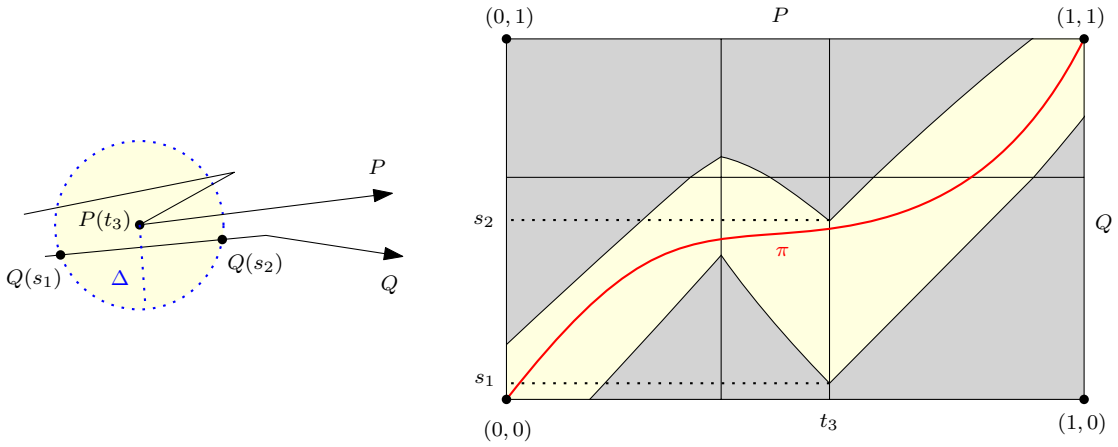


Figure 2.4: Illustration of two curves $P$ and $Q$ and their $\Delta$-free space. The monotone path $\pi$ illustrate that the Fréchet distance between $P$ and $Q$ is at most $\Delta$. The line segment $\overline{(t_3, s_1)(t_3, s_2)}$ is the free space interval corresponding to $P(t_3)$ and the first edge of $Q$.

## 2.4 Range spaces

Range spaces are sometimes also called set systems and are in general terms exactly that: sets that contain sets. Let $X$ be a set. Formally, we call a set $\mathcal{R}$ where any $r \in \mathcal{R}$ is of the form $r \subseteq X$ a ***range space*** with ***ground set*** $X$. The elements of $\mathcal{R}$ are called ***ranges***. We also write $(X, \mathcal{R})$ for a range space $\mathcal{R}$ with ground set $X$. Generally, range spaces can be any kind of sets. In many applications, the ground set is the Euclidean space. The ranges can, for example, be geometric objects like hypercubes, balls or half-spaces. In the case of subtrajectory clustering, we define the ground set to be all points of our trajectory data and the ranges as all points that can be represented by the same center curve. So each range is implicitly represented by a ball with a center and a radius (formal definition in Section 3.1.1). Our goal in subtrajectory clustering is to find the smallest set of centers such that the whole trajectory data (ground set) is represented. This is just one application of a general problem for range spaces called the set cover problem.

Let $\mathcal{R}$ be a range space with ground set $X$. A ***set cover*** of $\mathcal{R}$ is a subset $S \subset \mathcal{R}$ such that the ground set is equal to the union of the sets in $S$. The ***set cover problem*** asks to find a set cover for a given $\mathcal{R}$ using a minimum number of sets. Closely related to the set cover problem is its dual problem, the so-called ***hitting set problem***. A ***hitting set*** of $\mathcal{R}$ is a subset $S \subseteq X$ such that every set of $\mathcal{R}$ contains at least one element of $S$. The hitting set problem is to find a hitting set for a given $\mathcal{R}$ of minimum size. We denote with $\mathcal{R}^*$ the range space ***dual*** to $\mathcal{R}$. The range space $\mathcal{R}^*$ has ground set $\mathcal{R}$ and each set $r_x \in \mathcal{R}^*$ is defined by an element $x \in X$ as $r_x = \{r \in \mathcal{R} | x \in r\}$. The dual range space of $\mathcal{R}^*$ is again $\mathcal{R}$. The hitting set problem for $\mathcal{R}$ is equivalent to the set cover problem for the dual range space $\mathcal{R}^*$. Sometimes, solving the hitting set problem for the dual range space is easier than directly solving the set cover problem or the other way around. We will use this approach for subtrajectory clustering.

A relaxed version of a hitting set is the ***$\varepsilon$-net***. For a given weight function $w$ on the ground set $X$ and a real value $\varepsilon > 0$, we say that a subset $C \subset X$ is an ***$\varepsilon$-net*** if every set of $\mathcal{R}$ of weight at least $\varepsilon \cdot w(X)$ contains at least one element of $C$. For any $A \subseteq X$, we write $w(A)$ short for $\sum_{a \in A} w(a)$. One way to calculate a hitting set is by iteratively

computing $\varepsilon$-nets and updating the weight function until one of the computed $\varepsilon$-nets is a hitting set. We will describe this method in detail in Section 2.4.2. The most popular way to generate $\varepsilon$-nets is via sampling where the sample size depends on the success probability and either the $VC$-dimension or the shattering dimension of the range space. We introduce these concepts in the following section.

### 2.4.1 VC-dimension, shattering and generating epsilon-nets

The VC-dimension (introduced by Vapnik and Chervonenkis [129]) measures the complexity of a range space based on shattering. We say a subset $A \subseteq X$ is ***shattered*** by $\mathcal{R}$ if for any $A' \subseteq A$ there exists an $r \in \mathcal{R}$ such that $A' = r \cap A$. The ***VC-dimension*** of $\mathcal{R}$ (denoted by $VCdim(\mathcal{R})$) is the maximal size of a set $A \subseteq X$ that is shattered by $\mathcal{R}$. A crucial property of the $VC$-dimension is, that it can be used to bound the number of ranges in $\mathcal{R}$ based on the number of elements in the ground set $X$. The growth function of a range space measures this relation and generalizes it to all subsets of $X$. For $m \in \mathbb{N}$, the ***growth function*** $\Pi_{\mathcal{R}}(m)$ is defined as

$$\Pi_{\mathcal{R}}(m) := \max_{A \subseteq X : |A| = m} |\{r \cap A \mid r \in \mathcal{R}\}|.$$

**Lemma 2.4.1** (Sauer's lemma [114, 129])**.** *For any $(X, \mathcal{R})$ of finite VC-dimension $\delta > 1$ and $m > 1$, we have*

$$\Pi_{\mathcal{R}}(m) \leq \sum_{i=1}^{\delta} \binom{m}{i} \leq m^{\delta}$$

A more direct way to measure this relation is the shattering dimension. The ***shattering dimension*** of $\mathcal{R}$ (denoted by $Sdim(\mathcal{R})$) is the smallest $s \in \mathbb{N}$ such that $\Pi_{\mathcal{R}}(m) \leq m^s$ for all $m \geq 2$. The shattering dimension and the VC dimension are closely related. Lemma 2.4.1 already shows that the VC-dimension is always an upper bound for the shattering dimension. In the other direction, they only differ by a logarithmic factor.

**Lemma 2.4.2.** *For any $(X, \mathcal{R})$ of finite VC-dimension $\delta > 1$and shattering dimension $s$, we have*

$$s \leq \delta \leq 2s \log_2(s)$$

*Proof.* By Lemma 2.4.1, we have $s \leq \delta$. The largest set that can be shattered by $\mathcal{R}$ has cardinality $\delta$. Therefore, we have $\Pi_{\mathcal{R}}(\delta) = 2^{\delta}$. By the definition of the shattering dimension, we get

$$\begin{aligned} 2^{\delta} &\leq \delta^s \\ \delta &\leq s \log_2(\delta) \\ \frac{\delta}{\log_2(\delta)} &\leq s \end{aligned}$$

Since $\delta > 1$, we get $s > 1$ and therefore $\delta \leq 2s \log_2(s)$. □

Depending on the range space, it can be easier to bound either the shattering dimension or the VC-dimension. After determining either one of the bounds Lemma 2.4.2 directly yields a bound for the respective other dimension. Calculating the VC-dimension

or even finding good bounds for it, is a challenge in itself. We present in Section 2.4.3 a technique to derive upper bounds for them.

Both dimensions can be used in the context of sampling $\varepsilon$-nets. Perhaps, the most well-known sample bounds for generating $\varepsilon$-nets based on the VC-dimension were derived by David Haussler and Emo Welzl in their seminal work [80]:

**Theorem 2.4.3** ([80]). *For any $(X, \mathcal{R})$ of finite VC-dimension $\delta$, finite $B \subseteq X$ and $0 < \varepsilon, \alpha < 1$, if $N$ is a subset of $B$ obtained by at least*

$$\max\left(\frac{4}{3}\log\left(\frac{2}{\alpha}\right), \frac{8\delta}{\varepsilon}\log\left(\frac{8\delta}{\varepsilon}\right)\right)$$

*random independent draws, then $N$ is an $\varepsilon$-net of $B$ for $\mathcal{R}$ with probability at least $1 - \alpha$.*

The dependency on the VC-dimension in this theorem results from bounding the growth function $\Pi_{\mathcal{R}}(2m)$ by $(2m)^{\delta}$. This directly implies that the shattering dimension can also be used instead of the VC-dimension.

**Corollary 2.4.4** ([80, 103, 126]). *For any $(X, \mathcal{R})$ of finite shattering dimension $s$, finite $B \subseteq X$ and $0 < \varepsilon, \alpha < 1$, if $N$ is a subset of $B$ obtained by at least*

$$\max\left(\frac{4}{3}\log\left(\frac{2}{\alpha}\right), \frac{8s}{\varepsilon}\log\left(\frac{8s}{\varepsilon}\right)\right)$$

*random independent draws, then $N$ is an $\varepsilon$-net of $B$ for $\mathcal{R}$ with probability at least $1 - \alpha$.*

Theorem 2.4.3 has been extended and improved, in particular by Li, Long, Srinivasan [95], Komlós, Pach and Woeginger [91], see also the survey by Mustafa and Varadarajan [103, 126] and the book of Har-Peled [74] for an overview. The following direct improvement for the sample bound is based on a precise analysis of the growth function's dependency on the VC-dimension.

**Lemma 2.4.5** ([74]). *Let $(X, \mathcal{R})$ have finite VC-dimension $\delta > 1$. For $m \geq 2\delta$, we have*

$$\left(\frac{m}{\delta}\right)^{\delta} \leq \Pi_{\mathcal{R}}(m) \leq \sum_{i=1}^{\delta}\binom{m}{i} \leq \left(\frac{em}{\delta}\right)^{\delta}$$

**Theorem 2.4.6** ([74, 91]). *For any $(X, \mathcal{R})$ of finite VC-dimension $\delta$, finite $B \subseteq X$ and $0 < \varepsilon, \alpha < 1$, if $N$ is a subset of $B$ obtained by at least*

$$\max\left(\frac{4}{3}\log\left(\frac{4}{\alpha}\right), \frac{8\delta}{\varepsilon}\log\left(\frac{16}{\varepsilon}\right)\right)$$

*random independent draws, then $N$ is an $\varepsilon$-net of $B$ for $\mathcal{R}$ with probability at least $1 - \alpha$.*

It is also possible to generate $\varepsilon$-nets deterministically at the expense of a running time that is exponential in the VC-dimension.

**Theorem 2.4.7** ([22]). *For any $(X, \mathcal{R})$ of finite VC-dimension $\delta$, finite $B \subseteq X$ and $0 < \varepsilon, \alpha < 1$, an $\varepsilon$-net $N$ of $B$ of size $O\left(\frac{\delta}{\varepsilon}\log\left(\frac{\delta}{\varepsilon}\right)\right)$ can be computed deterministically in $O\left(\delta^{3\delta}\left(\frac{1}{\varepsilon}\log\left(\frac{1}{\varepsilon}\right)\right)^{\delta}|B|\right)$ time.*

### 2.4.2 Multiplicative weight update method for Hitting sets

To compute approximate solutions for the hitting set problem, we use an idea that goes back to Clarkson [45, 46] and was later also applied and extended by Brönniman and Goodrich [23] for range spaces of low VC-dimension. The underlying technique is commonly known as the multiplicative weight update method [15]. It can also be used to compute approximate solutions for the set cover problem by applying it on the dual range space.

In principle, the technique consists of iteratively generating $\varepsilon$-nets for a weighted version of the ground set and updating the weights until the generated $\varepsilon$-net is a hitting set. Generally, all techniques for generating $\varepsilon$-nets, that we discussed in Section 2.4.1 can be used here. For better readability, we limit ourselves to the version that we will apply. This version uses a sample-based approach based on the sample bounds in Theorem 2.4.6. The generated samples are candidates for hitting sets. To evaluate if a sampled candidate is a hitting set, our algorithm needs a verifier as a subroutine.

**Definition 2.4.8** ([23]). *A **verifier** is an algorithm $A$ that, given a subset $H \subseteq X$, either states (correctly) that $H$ is a hitting set, or returns a nonempty set $r$ of $\mathcal{R}$ such that $r \cap H = \emptyset$.*

**Algorithm 1** Multiplicative Weight Update Algorithm

**procedure** MWU-ALGORITHM($X$, $\mathcal{R}$)
  $k \leftarrow 1$
  $\delta \leftarrow$ VC-dim($\mathcal{R}$)
  **repeat**
    $k \leftarrow 2k$ ▷ increase target size for solution
    $C \leftarrow$ $k$-MWU-ALGORITHM$(X, \mathcal{R}, k, \delta)$ ▷ search solution with this size
  **until** $C \neq \emptyset$ ▷ until we find a solution
  **return** $C$

**procedure** $k$-MWU-ALGORITHM$(X, \mathcal{R}, k, \delta)$
  $\varepsilon \leftarrow \frac{1}{2k}$, $k' \leftarrow \lceil 16\delta k \log(16k) \rceil$, $i_{\max} \leftarrow 4k \log_2\left(\frac{|X|}{k}\right)$
  Let $\mathcal{D}_1$ be the uniform distribution over $X$ with weight function $w_1 : X \to \{1\}$
  $i \leftarrow 1$
  **repeat**
    $C \leftarrow$SAMPLE$(k', \mathcal{D}_i)$ ▷ sample candidate $C$ with $k'$ elements from $\mathcal{D}_i$
    $r \leftarrow$ VERIFIER$(C, X, \mathcal{R})$ ▷ Check if $C$ is hitting set
    **if** $r = \emptyset$ **then return** $C$ ▷ if $C$ is hitting set, return it as solution
    **if** $\Pr_{\mathcal{D}_i}[r] \leq \varepsilon$ **then**
      $\mathcal{D}_{i+1} \leftarrow$ WEIGHTUPDATE$(\mathcal{D}_i, r)$ ▷ Double weight of elements in $r$
      $i \leftarrow i + 1$
  **until** $i > i_{\max}$
  **return** $\emptyset$ ▷ no solution found for this target size

**Algorithm** Given a range space $\mathcal{R}$ with ground set $X$, our algorithm proceeds as follows (see also Algorithm 1). Assume, we know there exists a hitting set of size $k$ (see $k$-MWU-ALGORITHM). The algorithm samples a candidate set $C \subseteq X$ for a hitting set

of $\mathcal{R}$. The sample size is chosen based on Theorem 2.4.6 such that $C$ is an $\varepsilon$-net with probability $\frac{1}{2}$ (for $\varepsilon = \frac{1}{2k}$). Then, the algorithm calls the verifier to test if $C$ is a hitting set for $\mathcal{R}$. If this is the case, then the algorithm returns $C$. If it is not, then the verifier returns a witness set $r$ that does not contain any element of $C$. If the weight of $r$ is at most an $\varepsilon$-fraction of the total weight of the ground set, then the algorithm doubles the weight of each element of $r$. It repeats the whole procedure until it finds a hitting set. Since the algorithm does not know the optimal value of $k$, it does a doubling search starting with $k = 2$: To take care of the case that there may not exist a hitting set of size $k$, the algorithm terminates the current round after $4k \log\left(\frac{|X|}{k}\right)$ weight-updates if no hitting set has been found. If this happens, it doubles the value of $k$, resets the weights, and starts over.

In addition to the verifier, the algorithm needs a data structure $\mathcal{D}$ that maintains the probability distributions $\mathcal{D}_i$ and can answer the following three queries: The first query $\textsc{Sample}(k', \mathcal{D}_i)$ asks to sample $k'$ elements from $D_i$, the second query $\textsc{WeightUpdate}(\mathcal{D}_i, r)$ asks to double the weight of each element in $r$ in the weight function of the distribution $\mathcal{D}_i$, the third query asks to evaluate $\Pr_{\mathcal{D}_i}[r] \leq \varepsilon$ for given $r$ and $\varepsilon$. The implementation of this data structure and the data structure for the verifier may vary depending on the structure of the range spaces in different applications. Let $P_{\mathcal{D}}, P_V$ be the preprocessing times and $S_{\mathcal{D}}, S_V$ the space requirements of the two data structures. Let further $T_V$ be the maximal query time for the verifier, let $T_{\mathcal{D}}$ be the maximal query time for all three types of queries of $\mathcal{D}$ and let $U_{\mathcal{D}}$ be the time needed to reset the weight function in $\mathcal{D}$ to create a uniform distribution.

**Theorem 2.4.9.** *For a given finite range space $(X, \mathcal{R})$ with finite VC-dimension $\delta$, assume there exists a hitting set of size $k$. Then, there exists an algorithm that computes a hitting set of size $k' \in O(\delta k \log(k))$ with expected running time in*

$$O\left(k \log\left(\frac{|X|}{k}\right)(T_{\mathcal{D}} + T_V) + \log(k) U_{\mathcal{D}}\right)$$

*after a preprocessing time in $O\Big(P_{\mathcal{D}} + P_V\Big)$ and using space in $O\Big(S_{\mathcal{D}} + S_V\Big)$.*

*Proof.* We use the MWU-Algorithm (Algorithm 1) that does a doubling search on $k$, starting with $k = 2$. In the remaining proof, we analyze the running time of k-MWU-Algorithm for a fixed $k$ in detail. In each iteration of the algorithm, a random sample of size $O(k\delta \log(k))$ is computed. If the sample is a hitting set, the verifier confirms this and the algorithm terminates and returns a solution of given size. In each iteration of the algorithm, the computed random sample is an $\varepsilon$-net with probability greater $\frac{1}{2}$ by Theorem 2.4.6. Therefore, the expected number of iterations until we find an $\varepsilon$-net between any two weight-update steps is at most 2. By construction, $4k \log(\frac{|X|}{k})$ is the maximum number of weight-update steps before the algorithm updates $k$. The analysis in [23] implies that this number of weight-update steps suffices for the algorithm to find a hitting set (assuming there exists a hitting set of size $k$). We include the analysis here and verify that it also holds in our setting.

Let $H$ be a hitting set of $\mathcal{R}$ with $|H| = k$. Let $r$ be the set returned by the verifier in one iteration, where $w(r) \leq \varepsilon \cdot w(X)$. Since $H$ is a hitting set, we have $H \cap r \neq \emptyset$. Let $w$ be our weight function and let $z_h$ be the number of times the weight of $h \in H$ has been doubled after $i$ weight-update steps. Then we have after $i$ weight-update steps that

$$w(H) = \sum_{h \in H} 2^{z_h}, \text{ where } \sum_{h \in H} z_h \geq i.$$

By the convexity of the exponential function, we get $w(H) \geq k2^{\frac{i}{k}}$. Since $\varepsilon = \frac{1}{2k}$, we also have for the ground set $X$ that

$$w(X) \leq |X| \left(1 + \frac{1}{2k}\right)^i \leq |X| e^{\frac{i}{2k}}.$$

Because $H$ is a subset of $X$ and therefore $w(H) \leq w(X)$, we get in total

$$k2^{\frac{i}{k}} \leq |X| e^{\frac{i}{2k}} \leq |X| 2^{\frac{3i}{4k}}.$$

It directly follows that $i \leq 4k \log\left(\frac{|X|}{k}\right)$. Combining this result with the expected number of iterations until we find an $\varepsilon$-net, we conclude that the expected number of iterations before the algorithm terminates is smaller than $8k \log\left(\frac{|X|}{k}\right)$.

In each iteration, the algorithm computes a random sample in $T_{\mathcal{D}}$ time and applies the extended verifier in $T_V$ time. The check if a reweighting needs to be applied together with the reweighting itself can be done in $2T_{\mathcal{D}}$ time. So each iteration of the repeat-loop (line 5-12) has a running time of $O(T_{\mathcal{D}} + T_V)$. In total, we get an expected running time of

$$O\left(k \log\left(\frac{|X|}{k}\right)(T_{\mathcal{D}} + T_V)\right)$$

for all iterations of the repeat loop. Note that this running time is at least linear in $k$, so by a geometric series argument, the doubling search incurs only a constant factor in the total running time. The weights in the data structure $\mathcal{D}$ have to be reset to the uniform distribution every time the value for $k$ gets doubled. This happens at most $\log(k)$ times to get from 2 to $k$. Therefore the total running time for resetting is in $O(\log(k)U_{\mathcal{D}})$. □

### 2.4.3 Zero sets of polynomials: Bounds for VC-dimension and more

In this section, we describe a technique for computing upper bounds on the VC-dimension and shattering dimension of range spaces. The idea behind the technique goes back to Goldberg and Jerrum [68, 69] and, independently, Ben-David and Lindenbaum [18]. We focus on the variant of this technique by Anthony and Bartlett [14].

The fundamental observation on which this technique is based is the following. For complex range spaces with higher VC-dimension and shattering dimension, it tends to be harder to evaluate if an element $x$ is part of a range $r$ than for less complex range spaces. To quantify this relation, the idea is to find a set of boolean functions (predicates) on $x$ and $r$ that uniquely determines if $x$ is part of $r$. The boolean functions should be simple in the sense that they are expressible as sign values of polynomials with bounded degrees. It turns out that in this case one can bound the VC-dimension and shattering dimension based on the number and maximum degree of these polynomials. The bound also depends on the number of real-valued parameters that are needed to describe any range $r$ as input for the polynomials (see Theorem 2.4.10).

Crucial to this technique is counting the number of cells in the arrangement of zero sets of polynomials. The VC-dimension bound is not the only application where we use bounds on the number of these cells. We also use if for deriving running time bounds on the DBA algorithm. This connection between two combinatorial upper bounds that seem totally unrelated at first glance shows how versatile the use of the underlying arrangement is.

Let $\mathcal{R}$ be a range space with ground set $X$, and $F$ be a class of real-valued functions defined on $\mathbb{R}^d \times X$. For $a \in \mathbb{R}$ let $\operatorname{sgn}(a) = 1$ if $a \geq 0$ and $\operatorname{sgn}(a) = 0$ if $a < 0$. We say

that $\mathcal{R}$ is a ***t*-combination** of $\operatorname{sgn}(F)$ if there is a boolean function $g : \{0,1\}^t \to \{0,1\}$ and functions $f_1, \ldots, f_t \in F$ such that for all $r \in \mathcal{R}$ there is a parameter vector $y \in \mathbb{R}^d$ such that

$$r = \{x \in X \mid g(\operatorname{sgn}(f_1(y,x)), \ldots, \operatorname{sgn}(f_t(y,x))) = 1\}.$$

**Theorem 2.4.10** ([14], Theorem 8.3)**.** *Let $F$ be a class of functions mapping from $\mathbb{R}^d \times X$ to $\mathbb{R}$ so that, for all $x \in X$ and $f \in F$ the function $y \to f(y,x)$ is a polynomial on $\mathbb{R}^d$ of degree no more than $l$. Suppose that $\mathcal{R}$ is a $t$-combination of $\operatorname{sgn}(F)$. Then if $m \geq \frac{d}{t}$,*

$$\Pi_{\mathcal{R}}(m) \leq 2\left(\frac{2emlt}{d}\right)^d$$

*and hence $VCdim(\mathcal{R}) \leq 2d\log_2(12lt)$.*

Note that $VCdim(\mathcal{R}) < m$ if $\Pi_{\mathcal{R}}(m) < 2^m$ since in this case no set of size $m$ can be shattered by $\mathcal{R}$. By bounding the growth function, Theorem 2.4.10 also implies the following bound on the shattering dimension.

**Theorem 2.4.11.** *Let $F$ be a class of functions mapping from $\mathbb{R}^d \times X$ to $\mathbb{R}$ so that, for all $x \in X$ and $f \in F$ the function $y \to f(y,x)$ is a polynomial on $\mathbb{R}^d$ of degree no more than $l$. Suppose that $\mathcal{R}$ is a $t$-combination of $\operatorname{sgn}(F)$. Then we have*

$$Sdim(\mathcal{R}) \leq \max\left(d\ln\left(\frac{15lt}{d}\right) + 1, d+2\right)$$

*Proof.* Let $m > 1$. By Theorem 2.4.10 we have $\Pi_{\mathcal{R}}(m) \leq m^{Sdim(\mathcal{R})} \leq m^s$ for all $s$ with

$$2\left(\frac{2emlt}{d}\right)^d \leq m^s.$$

This implies

$$\frac{d\ln\left(\frac{2emlt}{d}\right) + \ln(2)}{\ln(m)} \leq s.$$

If $d \geq 2elt$ then the inequality holds for $s = d+2$ and otherwise for $s = d\ln\left(\frac{15lt}{d}\right) + 1$. □

In the remainder of this section, we give a proof of Theorem 2.4.10. The proof is included here to enable a better understanding of the technique. The proof is based on the following lemma which bounds the growth function via the number of connected components in an arrangement of zero sets.

**Lemma 2.4.12** (Lemma 7.8 [14])**.** *Let $F$ be a class of functions mapping from $\mathbb{R}^d \times X$ to $\mathbb{R}$ that is closed under the addition of constants. Suppose that the functions in $F$ are continuous in their parameters and that $\mathcal{R}$ is a $t$-combination of $\operatorname{sgn}(F)$ for a boolean function $g : \{0,1\}^t \to \{0,1\}$ and functions $f_1, \ldots, f_t \in F$. Then for every $m \in \mathbb{N}$ there exist a subset $\{x_1, \ldots, x_m\} \subset X$ and functions $f'_1, \ldots, f'_t \in F$ such that the number of connected components of the set*

$$\mathbb{R}^d - \bigcup_{i=1}^{t}\bigcup_{j=1}^{m}\{y \in \mathbb{R}^d : f'_i(y, x_j) = 0\}$$

*is at least $\Pi_{\mathcal{R}}(m)$.*

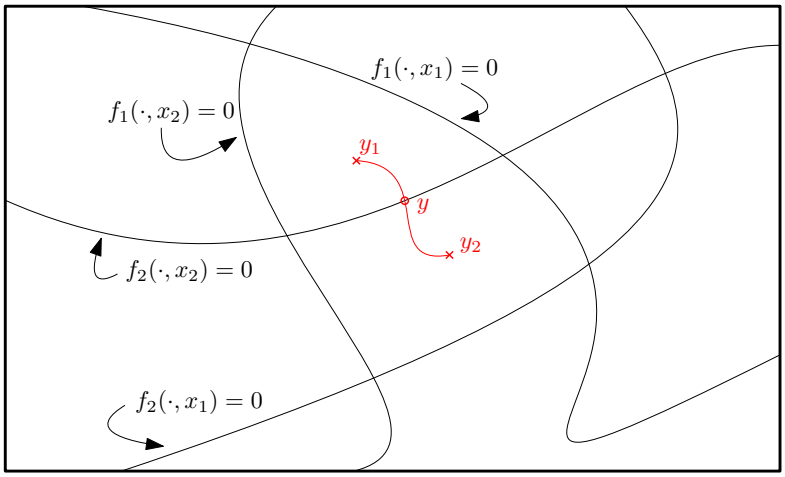


Figure 2.5: Illustration for the proof of Lemma 2.4.12: In this example $y_1$ and $y_2$ differ in $\operatorname{sgn}(f_2(\cdot, x_2))$.

*Proof of Lemma 2.4.12.* The proof is an adaptation of the proof in [14] that uses our notation. Let $A = \{x_1, \ldots, x_m\} \subset X$ be any subset of size $m$ of $X$. Let further $\mathcal{R}_{|A} = \{A \cap r \mid r \in \mathcal{R}\}$ be the restriction of $\mathcal{R}$ to $A$. Observe that $\Pi_{\mathcal{R}}(m)$ is equal to $|\mathcal{R}_{|A}|$ for a set $A$ that maximizes this quantity. Let $A$ be such a set. We denote the arrangement of zero sets of $\mathcal{R}_{|A}$ with $S \coloneqq \mathbb{R}^d - \bigcup_{i=1}^{t} \bigcup_{j=1}^{m} \{y \in \mathbb{R}^d : f_i(y, x_j) = 0\}$. Each range $r_y \in \mathcal{R}_{|A}$ is defined by a parameter $y \in \mathbb{R}^d$ such that

$$r_y = \{x \in A \mid g(\operatorname{sgn}(f_1(y, x)), \ldots, \operatorname{sgn}(f_t(y, x))) = 1\}.$$

The elements of $S$ can be interpreted as these parameters $y$. We want to show that in each connected component of $S$ all parameters define the same range of $\mathcal{R}_{|A}$. Let $y_1, y_2 \in S$ with $r_{y_1} \neq r_{y_2}$. There exist $i$ and $j$ such that $f_i(y_1, x_j)$ and $f_i(y_2, x_j)$ have different signs. So on every continuous path from $y_1$ to $y_2$ there must be a $y$ such that $f_i(y, x_j) = 0$. This follows directly from the continuity of $f_i$. Therefore $y_1$ and $y_2$ have to be in different connected components of $S$ (see Figure 2.5 for an example in the plane). However, in general, it could happen that some ranges of $\mathcal{R}_{|A}$ can only be realized with a parameter $y$ such that $f_i(y, x_j) = 0$ for some $i$ and $j$. In this case, $y \notin S$. To prevent this, we define slightly shifted variations $f_1', \ldots, f_t'$ of the functions $f_1, \ldots, f_t$ such that every $r \in \mathcal{R}_{|A}$ can be realized by some $y \in S'$ where $S' \coloneqq \mathbb{R}^d - \bigcup_{i=1}^{t} \bigcup_{j=1}^{m} \{y \in \mathbb{R}^d : f_i'(y, x_j) = 0\}$. Let $|\mathcal{R}_{|A}| = N$ and $y_1, \ldots, y_N \in \mathbb{R}^d$ such that $\mathcal{R}_{|A} = \{r_{y_1}, \ldots, r_{y_N}\}$. Choose

$$\varepsilon = \frac{1}{2} \min\{|f_i(y_l, x_j)| : f_i(y_l, x_j) < 0, 1 \leq i \leq t, 1 \leq j \leq m, 1 \leq l \leq N\}$$

and set $f_i'(x, y) = f_i(y, x) + \varepsilon$ for all $i$. By construction, the sign values of all functions stay the same and none of them evaluates to zero for $y_1, \ldots, y_N$. Therefore the number of connected components of $S'$ is at least $N$. □

It remains to bound the number of connected components in the arrangement of Lemma 2.4.12 by $2(\frac{2emlt}{d})^d$ for every $t$-combination of $\operatorname{sgn}(F)$ under the specifications on $F$ and $m$ in Theorem 2.4.10. For the number of connected components in the arrangement

of zero sets of polynomials, bounds were obtained by Oleĭnik and Petrovski [108], Milnor [102], Thom [125] and Warren [132], see also the survey paper by Alon [10].

**Theorem 2.4.13** (Warren [132])**.** *Let $f_1, \ldots, f_n$ be real-valued polynomials in $d$ variables, each of degree $l$ or less. Then the number of connected components of the set*

$$\mathbb{R}^d - \bigcup_{i=1}^{n} \{y \in \mathbb{R}^d : f_i(y) = 0\}$$

*is at most $2(2l)^d \sum_{k=0}^{d} 2^k \binom{n}{k}$. In particular for $n \geq d$ it is at most $\left(\frac{4enl}{d}\right)^d$.*

The bound in Theorem 2.4.13 directly yields an upper bound on the growth function of $\left(\frac{4emlt}{d}\right)^d$ by setting $n = mt$. A more careful analysis in [14] improves this to $2\left(\frac{2emlt}{d}\right)^d$ for the specific case in Theorem 2.4.10.

In case of our running time bound for DBA (Theorem 6.2.4), we will use a bound on a slightly different quantity. Instead of the connected components of the arrangement, we are interested in the total number of sign patterns of the polynomials. Let $f_1, \ldots, f_n$ be real-valued polynomials in $d$ variables. The ***sign pattern*** of $f_1, \ldots, f_n$ at a point $x \in \mathbb{R}^d$ is defined as $\operatorname{sgn}(f_1(x)), \ldots, \operatorname{sgn}(f_1(x))$. The bounds of Theorem 2.4.13 can directly be applied to sign patterns as well. This can be proven analogous to the last part of the proof of Lemma 2.4.12.

**Corollary 2.4.14** ([132])**.** *Let $f_1, \ldots, f_n$ be real-valued polynomials in $d$ variables, each of degree $l$ or less. Let $n \geq d$. The total number of sign patterns of $f_1, \ldots, f_n$ is at most $2(2l)^d \sum_{k=0}^{d} 2^k \binom{n}{k}$. In particular for $n \geq d$ it is at most $\left(\frac{4enl}{d}\right)^d$.*

For the case $d \geq n$, the bound $2(2l)^d \sum_{k=0}^{d} 2^k \binom{n}{k}$ in Corollary 2.4.14 is not easily interpretable. We instead use the following bound in the worst-case analysis of the DBA algorithm

**Theorem 2.4.15.** *Let $f_1, \ldots, f_n$ be real-valued polynomials in $d$ variables, each of degree $l \geq 1$ or less. The total number of sign patterns of $f_1, \ldots, f_n$ is at most $6(2ln)^d$.*

*Proof.* By Theorem 2.4.13, we only have to show that $\sum_{k=0}^{d} 2^k \binom{n}{k} \leq 3n^d$. We first analyze the case $n \geq d$: For $d \geq 4$, we have

$$\sum_{k=0}^{d} 2^k \binom{n}{k} \leq \sum_{k=4}^{d} n^k + \sum_{k=0}^{3} \frac{2^k}{k!} n^k \leq 2n^d + (1 + 2n + 2n^2 + \frac{8}{3}n^3) \leq 2n^d + n^4 \leq 3n^d.$$

For $d \in \{1, 2, 3\}$ similar calculations show $\sum_{k=0}^{d} 2^k \binom{n}{k} \leq 3n^d$. It remains to analyze the case $d \geq n$: For $n \geq 4$, we get

$$\sum_{k=0}^{d} 2^k \binom{n}{k} \leq 2^d \sum_{k=0}^{d} \binom{n}{k} \leq 4^d \leq n^d.$$

For $n \in \{1, 2, 3\}$ calculating the exact value for the term $\sum_{k=0}^{d} 2^k \binom{n}{k} = \sum_{k=0}^{n} 2^k \binom{n}{k}$ yields the bound. □

The result of Theorem 2.4.10 also enables to bound the VC-dimension of a range space based on the number of simple operations that are needed to determine if an element of the ground set is included in a range. This was shown by Anthony and Bartlett [14] in the following theorem.

**Theorem 2.4.16** (Theorem 8.4 [14]). *Suppose $h$ is a function from $\mathbb{R}^d \times \mathbb{R}^n$ to $\{0,1\}$. Let $\mathcal{R}$ be the range space with ground set $\mathbb{R}^d$, where a set $r_\alpha \in \mathcal{R}$ for $\alpha \in \mathbb{R}^d$ is defined as*

$$r_\alpha = \{x \in \mathbb{R}^n \mid h(\alpha, x) = 1\}.$$

*Suppose that $h$ can be computed by an algorithm that takes as input the pair $(\alpha, x) \in \mathbb{R}^d \times \mathbb{R}^n$ and returns $h(\alpha, x)$ after no more than $t$ simple operations. Then, the VC-dimension of $\mathcal{R}$ is $\leq 4d(t+2)$.*

*Proof sketch.* The algorithm that computes $h$ can be expressed by a computation tree of depth at most $t$. Each comparison of the algorithm can be expressed by a comparison of polynomials of degree at most $2^t$ and the number of comparisons in an algorithm of depth at most $t$ is bounded by $2^{t-1} - 1$. To be able to compute all types of comparisons $=, \neq, >, \geq, \leq, <$, also negated copies of the polynomial might be needed. This yields a bound of $2^t - 2$ on the number of polynomials. Applying Theorem 2.4.10 directly yields the result. □

Through a careful analysis of the underlying polynomials Theorem 2.4.10 can generally yield better bounds than Theorem 2.4.16. On the other hand, Theorem 2.4.16 has the advantage of easier and more intuitive applicability.

## 2.5 Subtrajectory Clustering: Problem definition

In Chapter 4 and 5, we study subtrajectory clustering under the Fréchet distance in the following way. For a given polygonal curve $P$ with $n$ vertices in fixed dimension, integers $k$, $\ell \geq 1$, and a real value $\Delta > 0$, the goal is to find $k$ center curves of complexity at most $\ell$ such that every point on $P$ is covered by a subtrajectory that has small Fréchet distance to one of the $k$ center curves ($\leq \Delta$).

To formally define this objective, we first introduce the notion of a $\Delta$-coverage. Let $P : [0,1] \to \mathbb{R}^d$ be a polygonal curve[1] of $n$ vertices and let $\ell \in \mathbb{N}$ and $\Delta \in \mathbb{R}$ be fixed parameters. Define the $\Delta$-***coverage*** of a set of center curves $C \subset \mathbb{X}_\ell^d$ as follows:

$$\Psi_\Delta(P, C) = \bigcup_{q \in C} \bigcup_{0 \leq t \leq t' \leq 1} \{s \in [t, t'] \mid d_F(P[t, t'], q) \leq \Delta\}.$$

An illustration of the $\Delta$-coverage is given in Figure 2.6. The $\Delta$-coverage corresponds to the part of the curve $P$ that is covered by the set of all subtrajectories that are within Fréchet distance $\Delta$ to some curve in $C$. If for some $P, C, \Delta$ it holds that $\Psi_\Delta(P, C) = [0,1]$, then we call $C$ a $\Delta$-***covering*** of $P$. The problem we study in this paper is to find a $\Delta$-covering $C \subset \mathbb{X}_\ell^d$ of $P$ of minimum size. We call this problem the $(\Delta, \ell)$-***covering problem*** on $P$. In particular, we study bicriterial approximation algorithms for this problem, which we formalize in the following way.

**Definition 2.5.1** ($(\alpha, \beta)$-approximate solution). *Let $P \in \mathbb{X}_n^d$ be a polygonal curve, $\Delta \in \mathbb{R}_+$ and $\ell \in \mathbb{N}$. A set $C \subseteq \mathbb{X}_\ell^d$ is an $(\alpha, \beta)$-approximate solution to the $(\Delta, \ell)$-covering problem on $P$, if $C$ is an $\alpha\Delta$-covering of $P$ and there exists no $\Delta$-covering $C' \subseteq \mathbb{X}_\ell^d$ of $P$ with $\beta|C'| < |C|$.*

[1] We chose the setting of one input curve to keep the presentation of our algorithmic solutions as simple as possible. All of our algorithms can be easily extended to the setting of multiple input curves.

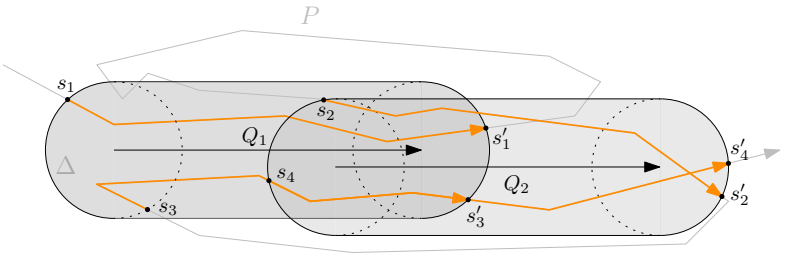


Figure 2.6: Illustration of the Δ-coverage of a set $C = \{Q_1, Q_2\}$ and a curve $P$. Here we have $\Psi_\Delta(P, C) = [s_1, s'_1] \cup [s_2, s'_2] \cup [s_3, s'_4]$, since the subcurves $P[s_1, s'_1]$ and $P[s_3, s'_3]$ have Fréchet distance Δ to $Q_1$, the subcurves $P[s_2, s'_2]$ and $P[s_4, s'_4]$ have Fréchet distance Δ to $Q_2$ and each other subcurve of $P$ that has Fréchet distance at most Δ to $Q_1$ or $Q_2$ is a subcurve of $P[s_i, s'_i]$ for some $1 \leq i \leq 4$.

We define the ***radius*** of the clustering induced by $C$ as the smallest real value Δ such that $\Psi_\Delta(P, C) = [0, 1]$, and we denote the radius with $\psi(P, C)$. Note that our problem definition requires center curves to be of complexity at most $\ell$, which is given as a parameter.[2] Throughout this thesis, we assume that $d$ is a constant independent of $n$ when discussing subtrajectory clustering.

### 2.5.1 Range space formulation

Our approach to the $(\Delta, \ell)$-covering problem works via set covers of suitable range spaces. To this end, we will first define a ***discrete variant*** of the problem, that we use in Chapter 4. In Chapter 5, we discretize the problem slightly differently since we simplify the input curve first.

Assume that the curve $P$ is endowed with a set of $m$ real values $0 = t_1 < t_2 < \cdots < t_m = 1$ which define a set of subcurves of the form $P[t_i, t_j]$. We denote the set of values $t_i$ with $\mathcal{T}$ and we refer to the respective points on the curve $P(t_i)$ for $1 \leq i \leq m$ as ***breakpoints***. Note that the vertices of the curve do not have to be breakpoints and vice versa. For a given curve $P$ with breakpoints we define the Δ-coverage of a set of center curves $C \subset \mathbb{X}_\ell^d$ with respect to these breakpoints as follows

$$\Phi_\Delta(P, C) = \bigcup_{q \in C} \bigcup_{1 \leq i \leq j \leq m} \{s \in [t_i, t_j] \mid d_F(P[t_i, t_j], q) \leq \Delta\}$$

Analogous to the definition in the continuous case, we define the radius of the clustering in the discrete case as the smallest real value Δ such that $\Phi_\Delta(P, C) = [0, 1]$ and we denote this radius with $\phi(P, C)$. Consider the range space $\mathcal{R}$ with ground set $X = \{1, \ldots, m-1\}$ where each set $r_Q \in \mathcal{R}$ is defined by a polygonal curve $Q \in \mathbb{X}_\ell^d$ as follows

$$r_Q = \{z \in X \mid \exists i \leq z < j \text{ with } d_F(Q, P[t_i, t_j]) \leq \Delta\} \tag{2.1}$$

In the discrete case, the problem of finding a minimum-size set of center curves that cover $P$ now reduces to finding a minimum-size set cover for the range space $\mathcal{R}$. Stating the problem in terms of range spaces allows us to draw from a rich background

[2] It is tempting to relax the restriction on the complexity of the center curves in our problem definition. However, without any other regularization of the optimization problem, this would lead to the trivial solution of the curve $P$ being an optimal center curve.

of algorithmic techniques for computing set covers (as the multiplicative weight update method which we introduced in Section 2.4.2).

In the following, we discuss how solutions to the discrete problem help to solve the continuous $(\Delta, \ell)$-covering problem. We can choose breakpoints for the input curve $P$, such that the distance between two consecutive breakpoints is at most $\varepsilon\Delta$ for any fixed $\varepsilon > 0$. This is always possible with $m = \lceil \frac{\lambda}{\varepsilon\Delta} \rceil$ breakpoints, where $\lambda$ is the arclength of $P$. The resulting instance of the discrete problem variant approximates the continuous version of the problem in the following way.

**Lemma 2.5.2.** *Assume there exists a set $C^* \subset \mathbb{X}_\ell^d$ of size $k$, such that $\psi(P, C^*) \leq \Delta$. Then we have $\phi(P, C^*) \leq (1+\varepsilon)\Delta$. Additionally for each $C \subset \mathbb{X}_\ell^d$ with $\phi(P, C) \leq (1+\varepsilon)\Delta$ we have $\psi(P, C^*) \leq (1+\varepsilon)\Delta$.*

*Proof.* We show that for any set of center curves $C \subset \mathbb{X}_\ell^d$ we have $\psi(P, C) \leq \phi(P, C) \leq (1+\varepsilon)\psi(P, C)$. Indeed, if $C$ covers the curve $P$ in the discrete setting, then it also covers the curve $P$ in the continuous setting. Therefore, $\psi(P, C) \leq \phi(P, C)$. For showing the other inequality we observe that the distance between two consecutive breakpoints is at most $\varepsilon\Delta$. Therefore, for any interval $[s, t] \subset [0, 1]$ we can choose breakpoints $t_i \leq s$ and $t_j \geq t$ such that $d_F(P[s, t], P[t_i, t_j]) \leq \varepsilon\Delta$. The claim now follows from the triangle inequality. □

# Chapter 3

# Simplified and Improved Bounds on the VC-Dimension for Elastic Distance Measures

The main content of this chapter previously appeared as the paper *Simplified and Improved Bounds on the VC-Dimension for Elastic Distance Measures* [27] by Frederik Brüning and Anne Driemel which is available on arXiv. An initial version of the work has also been presented at the *40th European Workshop on Computational Geometry (EuroCG 2024)* [28] based on an extended abstract without formal publication. In this chapter, we study the VC-dimension and shattering dimension of range spaces, where the ground set consists of either polygonal curves in $\mathbb{R}^d$ or polygonal regions in the plane that may contain holes and the ranges are balls defined by an elastic distance measure, such as the Hausdorff distance, the Fréchet distance and the dynamic time warping distance. This chapter extends the VC-dimension bounds of [27] with bounds on the shattering dimension and by considering the average Hausdorff distance as an additional distance measure.

## 3.1 Introduction

Previous to our work, Driemel, Nusser, Philips and Psarros [57] derived almost tight bounds on the VC-dimension of balls in the setting of polygonal curves and with respect to the Hausdorff distance and the Fréchet distance. At the heart of their approach lies the definition of a set of boolean functions (predicates) which are based on the inclusion and intersection of simple geometric objects. The predicates depend on the vertices of a center curve and a radius that defines a metric ball as well as the vertices of a query curve. Some of the predicates originate from the work of Afshani and Driemel [2] on range searching. The predicates are chosen such that, based on their truth values, one can determine whether the query curve is contained in the respective ball. Their proof of the VC-dimension bound uses the worst-case number of operations needed to determine the truth values of each predicate. Their approach explicitly uses Theorem 2.4.16 by Anthony and Bartlett [14]. See Section 2.4.3 for more details on the approach.

In this chapter, we extend the known set of predicates to be able to decide the Hausdorff distance between polygonal regions with holes in the plane. We give an improved analysis for the VC-dimension that considers each predicate as a combination of sign

| | | new | ref. | Driemel et al. [57] |
|---|---|---|---|---|
| finite sets | av. Hausdorff | $O(dk\log(k^m m^k))$ | Thm. 3.2.4 | - |
| | Hausdorff | $O(dk\log(km))$ | Thm. 3.2.1 | $O(dk\log(dkm))$ |
| discrete polygonal curves | Fréchet | $O(dk\log(km))^{(*)}$ | Thm. 3.2.2 | |
| | DTW | $O(dk^2\log(m))$ | Thm. 3.2.3 | - |
| | | $O(dkm\log(k))$ | Thm. 3.2.3 | |
| continuous polygonal curves | Hausdorff | $O(dk\log(km))$ | Thm. 3.4.15 | $O(d^2k^2\log(dkm))$ |
| | Fréchet | $O(dk\log(km))^{(*)}$ | Thm. 3.4.16 | |
| | weak Fréchet | $O(dk\log(km))^{(*)}$ | Thm. 3.4.16 | $O(d^2k\log(dkm))$ |
| polygons $\mathbb{R}^2$ | Hausdorff | $O(k\log(km))$ | Thm. 3.4.15 | - |

Table 3.1: Overview of VC-dimension bounds with references. Results marked with $^{(*)}$ were independently obtained by Cheng and Huang [43].

values of polynomials and bound the VC-dimension based on the number of cells in the arrangement of zero sets of these polynomials. This approach does not use the computational complexity of the distance evaluation but instead uses the underlying structure of the range space defined by a system of polynomials directly (using Theorem 2.4.10 instead of Theorem 2.4.16). By the lower bounds in [57], this approach directly leads to tight bounds for $d \geq 4$ for polygonal curves. In addition to the upper bounds on the VC-dimension we give upper bounds on the shattering dimension using the same techniques.

### 3.1.1 Results

To state our results, we first formally introduce the considered range spaces. We define the ball with radius $\Delta$ and center $c$ under the distance measure $d_\rho$ on a set $X$ as $B_{\rho,\Delta}(c) = \{x \in X \mid d_\rho(x, c) \leq \Delta\}$. If $d_\rho$ is the Euclidean distance, we just write $B_\Delta(c)$. We study range spaces with ground set $(\mathbb{R}^d)^m$ of the form

$$\mathcal{R}_{\rho,k} = \{B_{\rho,\Delta}(c) \mid \Delta \in \mathbb{R}_+, \Delta > 0, c \in (\mathbb{R}^d)^k\}.$$

We call $(\mathbb{R}^d)^k$ the ***center set*** of $\mathcal{R}_{\rho,k}$. We derive new bounds on the VC-dimension for range spaces of the form $\mathcal{R}_{\rho,k}$ for some distance measure $d_\rho$ with a ground set $\mathbb{R}^d_m$ consisting of polygonal curves or polygonal regions that may contain holes. To this end, we write each range as a combination of sign values of polynomials with constant degrees and apply Theorem 2.4.10. More precisely, we take predicates that determine if a curve $P \in \mathbb{X}^d_m$ is in a fixed range $r \in \mathcal{R}_{\rho,k}$ and show that each such predicate can be written as a combination of sign values of polynomials with constant degree. See Section 2.4.3 for a proof of Theorem 2.4.10 and a more detailed discussion of the underlying technique.

For the ***Hausdorff distance*** of ***polygonal regions*** (with holes) in the plane, we show that the VC-dimension of $\mathcal{R}_{\rho,k}$ is bounded by $O(k\log(km))$. Interestingly, this bound is independent of the number of holes. To the best of our knowledge, this is the first non-trivial bound for the Hausdorff distance of polygonal regions.

| | | VC | shattering |
|---|---|---|---|
| finite sets | av. Hausdorff | $O(dk\log(k^m m^k))$ | $O(dk\log(k^m m^k/d))$ |
| | Hausdorff | $O(dk\log(km))$ | $O(dk\log(m/d))$ |
| discrete polygonal curves | Fréchet | $O(dk\log(km))$ | $O(dk\log(m/d))$ |
| | DTW | $O(dk^2\log(m))$ | $O(dk\log(m^k/d))$ |
| | | $O(dkm\log(k))$ | $O(dk\log(k^m/d))$ |
| continuous polygonal curves | Hausdorff | $O(dk\log(km))$ | $O(dk\log(km^2/d))$ |
| | Fréchet | $O(dk\log(km))$ | $O(dk\log(km^2/d))$ |
| | weak Fréchet | $O(dk\log(km))$ | $O(dk\log(km^2/d))$ |
| polygons $\mathbb{R}^2$ | Hausdorff | $O(k\log(km))$ | $O(k\log(km))$ |

Table 3.2: Overview of shattering dimension bounds in comparison to the VC-dimension bounds. In case that the term with the logarithm would have a negative value, the shattering dimension is bounded by $O(dk)$.

Note that the construction of the lower bound of $\Omega(\max(k, \log(m)))$ for $d \geq 2$ in [57] for polygonal curves under the Hausdorff distance can easily be generalized to a lower bound for the case of polygonal regions in the plane. To do so, we just have to replace each edge $e$ of a polygonal curve in their construction with a rectangle containing $e$ with suitable small width. This directly implies a bound of $\Omega(\max(k, \log(m)))$. Our upper bounds directly extend to unions of disjoint polygonal regions that may contain holes, where $k$ and $m$ denote the total complexity (number of edges) to describe the set.

For the ***Fréchet distance*** and the ***Hausdorff distance*** of ***polygonal curves***, in the discrete and the continuous case, we show that for the VC-dimension of $\mathcal{R}_{\rho,k}$ our techniques imply the same bound of $O(dk\log(km))$. Parallel and independent of our work, Cheng and Huang [43] obtained the same result for the Fréchet distance of polygonal curves using very similar techniques. An overview of our results with references to theorems and comparison to the known results from [57] and the independent results from [43] is given in Table 3.1.

The results improve upon the upper bounds of [57] in all of the considered cases. By the lower bound $\Omega(\max(dk\log(k), \log(dm)))$ for $d \geq 4$ in [57], the new bounds are tight in each of the parameters $k, m$ and $d$ for each of the considered distance measures on polygonal curves. For the ***Dynamic time warping distance***, we show a new bound of $O(\min(dk^2\log(m), dkm\log(k)))$ and for the ***average Hausdorff distance***, we show a new bound of $O(dkz\log(z))$ where $z = \max(m, k)$. Note that in the discrete setting with respect to the Hausdorff distance and average Hausdorff distance, polygonal curves are equivalent to finite point sets.

For the shattering dimension, we get slightly better bounds than the VC-dimension bounds by applying Theorem 2.4.11 instead of Theorem 2.4.10. The analysis of the bounds is otherwise analogous to the analysis of the VC-dimension bounds. We therefore omit details and only state the resulting bounds in Table 3.2. The improvement is a factor of up to $(dk)^{-1}$ in the logarithm. For the continuous case and the case of polygonal regions, any improvement would be hidden in the constants.

## 3.2 Warm-up: Discrete setting

In the discrete setting, we think of each curve $P \in \mathbb{X}_m^d$ as a sequence of its vertices $(p_1, \ldots, p_m) \in (\mathbb{R}^d)^m$ and not as a continuous function. To emphasize this, we write in this context $P \in (\mathbb{R}^d)^m$ instead of $P \in \mathbb{X}_m^d$.

**Theorem 3.2.1.** *Let $\mathcal{R}_{dH,k}$ be the range space of all balls under the Hausdorff distance centered at point sets in $(\mathbb{R}^d)^k$ with ground set $(\mathbb{R}^d)^m$. Then, we have*

$$VCdim(\mathcal{R}_{dH,k}) \leq 2(dk+1)\log_2(24mk).$$

*Proof.* Let $P \in (\mathbb{R}^d)^m$ with vertices $p_1, \ldots, p_m$ and $Q \in (\mathbb{R}^d)^k$ with vertices $q_1, \ldots, q_k$. The discrete Hausdorff distance between two point sets is uniquely defined by the distances of the points of the two sets. The truth value of $d_H(P,Q) \leq \Delta$ can therefore be determined given the truth values of $\|p-q\|^2 \leq \Delta^2$ for all pairs $(p,q) \in \{p_1, \ldots, p_m\} \times \{q_1, \ldots, q_k\}$. We can write the points $p, q \in \mathbb{R}^d$ as tuples of their coordinates with $p = (p_{(1)}, \ldots, p_{(d)})$ and $q = (q_{(1)}, \ldots, q_{(d)})$. Then we have that $\|p-q\|^2 \leq \Delta^2$ is equivalent to

$$\Delta^2 - \sum_{i=1}^{d}(p_{(i)} - q_{(i)})^2 \geq 0.$$

The term $\Delta^2 - \sum_{i=1}^{d}(p_{(i)} - q_{(i)})^2$ is a polynomial of degree 2 in all its variables. So the truth value of $\|p-q\|^2 \leq \Delta^2$ can be determined by the sign value of one polynomial of degree 2. There are in total $mk$ possible choices for the pair $(p,q)$. Let $y \in \mathbb{R}^{dk+1}$ be the vector consisting of all coordinates of the vertices $q_1, \ldots, q_k$ and of the radius $\Delta$. Then $\mathcal{R}_{dH,k}$ is a $mk$-combination of $\mathrm{sgn}(F)$ where $F$ is a class of functions mapping from $\mathbb{R}^{dk+1} \times (\mathbb{R}_m)^d$ to $\mathbb{R}$ so that, for all $P \in (\mathbb{R}_m)^d$ and $f \in F$ the function $y \to f(y,P)$ is a polynomial on $\mathbb{R}^d$ of degree no more than 2. The VC-dimension bound follows directly by applying Theorem 2.4.10. □

In the discrete case, the VC-dimension for the Fréchet distance can be analyzed in the same way as for the Hausdorff distance.

**Theorem 3.2.2.** *Let $\mathcal{R}_{dF,k}$ be the range space of all balls under the discrete Fréchet distance with ground set $(\mathbb{R}^d)^m$. Then, we have*

$$VCdim(\mathcal{R}_{dF,k}) \leq 2(dk+1)\log_2(24mk).$$

*Proof.* The proof is analogous to the proof of Theorem 3.2.1 given the fact that the discrete Fréchet distance between two polygonal curves is uniquely defined by the distances of the vertices of the two curves. □

**Theorem 3.2.3.** *Let $\mathcal{R}_{DTW,k}$ be the range space of all balls under the dynamic time warping distance with ground set $(\mathbb{R}^d)^m$. Then $VCdim(\mathcal{R}_{DTW,k})$ is in*

$$O(\min(dk^2\log(m), dkm\log(k))).$$

*Proof.* Let $P \in (\mathbb{R}_m)^d$ with vertices $p_1, \ldots, p_m$ and $Q \in (\mathbb{R}_k)^d$ with vertices $q_1, \ldots, q_k$. The truth value of $d_{DTW}(P,Q) \leq \Delta$ can be determined by the truth values of the inequalities $\sum_{(i,j)\in w}\|p_i - q_j\|^2 \leq \Delta$ for all $w \in \mathcal{W}^*_{m,k}$. This inequality is equivalent to

$$\Delta - \sum_{(i,j)\in w}\sum_{t=1}^{d}(p_{i,t} - q_{j,t})^2 \geq 0$$

for which the left side is a polynomial of degree 2 in all its variables. We get $|\mathcal{W}^*_{m,k}| \leq \binom{m+k-2}{m-1} \leq \min\{m^{k-1}, k^{m-1}\}$ by counting all possible optimal warping paths. Let $y \in \mathbb{R}^{dk+1}$ be the vector consisting of all coordinates of the vertices $q_1, \ldots, q_k$ and of the radius $\Delta$. Then $\mathcal{R}_{DTW,k}$ is a $\min\{m^{k-1}, k^{m-1}\}$-combination of $\operatorname{sgn}(F)$ where $F$ is a class of functions mapping from $\mathbb{R}^{dk+1} \times (\mathbb{R}_m)^d$ to $\mathbb{R}$ so that, for all $P \in (\mathbb{R}_m)^d$ and $f \in F$ the function $y \to f(y, P)$ is a polynomial on $\mathbb{R}^d$ of constant degree. The VC-dimension bound follows directly by the application of Theorem 2.4.10. □

The analysis of the VC-dimension for the average Hausdorff distance is similar to the analysis of the VC-dimension for DTW.

**Theorem 3.2.4.** *Let $\mathcal{R}_{daH,k}$ be the range space of all balls under the average Hausdorff distance centered at point sets in $(\mathbb{R}^d)^k$ with ground set $(\mathbb{R}^d)^m$. Then, we have*

$$VCdim(\mathcal{R}_{daH,k}) \leq 2(dk+1)\log_2(24m^k k^m).$$

*Proof.* Let $P \in (\mathbb{R}_m)^d$ with vertices $p_1, \ldots, p_m$ and $Q \in (\mathbb{R}_k)^d$ with vertices $q_1, \ldots, q_k$. Let $F_{m,k}$ be the class of all injective functions mapping from $\{1, \ldots, m\}$ to $\{1, \ldots, k\}$. The truth value of $d_{aH}(P, Q) \leq \Delta$ can be determined by the truth values of

$$\frac{1}{2}\left(\frac{1}{m}\sum_{i=1}^{m}\|p_i - q_{f(i)}\|^2 + \frac{1}{k}\sum_{j=1}^{k}\|q_j - p_{g(j)}\|^2\right) \leq \Delta$$

for all $(f, g) \in (\mathcal{F}_{m,k}, \mathcal{F}_{k,m})$. This inequality is equivalent to the sign value of a polynomial of degree 2 in all its variables. We have $|F_{m,k}| = m^k$. Therefore $\mathcal{R}_{daH,k}$ is a $(m^k k^m)$-combination of $\operatorname{sgn}(F)$ where $F$ is a class of functions mapping from $\mathbb{R}^{dk+1} \times (\mathbb{R}_m)^d$ to $\mathbb{R}$ so that, for all $P \in (\mathbb{R}_m)^d$ and $f \in F$ the function $y \to f(y, P)$ is a polynomial on $\mathbb{R}^d$ of degree no more than 2. The VC-dimension bound follows directly by the application of Theorem 2.4.10. □

## 3.3 Predicates

To bound the VC-dimension of range spaces of the form $\mathcal{R}_{\rho,k}$ in the continuous setting we define geometric predicates for distance queries with $d_\rho$. These predicates can, for example, consist of checking the distances of geometric objects or checking if some geometric intersections exist. They have to be chosen in a way that the query can be decided based on their truth values. We will show that our predicates can be viewed as combinations of simple predicates.

**Definition 3.3.1.** *Let $F$ be a class of functions mapping from $\mathbb{R}^{dm} \times \mathbb{R}^{dk+1}$ to $\mathbb{R}$ so that, for all $f \in F$ the function $(x, y) \to f(x, y)$ is a polynomial of constant degree. Let $\mathcal{P}$ be a function from $\mathbb{R}^{dm} \times \mathbb{R}^{dk+1}$ to $\{0, 1\}$. We say that the predicate $\mathcal{P}$ is **simple** if $\mathcal{P}$ is a $t$-combination of $\operatorname{sgn}(F)$ with $t \in O(1)$. We further say that an inequality is simple if its truth value is equivalent to a simple predicate.*

In our proof of the VC-dimension bounds, we will use the following corollary to Theorem 2.4.10.

**Corollary 3.3.2.** *Suppose that for a given $d_\rho$ there exists a polynomial $p(k, m)$ such that for any $k, m \in \mathbb{N}$ the space $\mathcal{R}_{\rho,k}$ with ground set $\mathbb{R}^{dm}$ is a $p(k, m)$-combination of simple predicates. Then $VCdim(\mathcal{R}_{\rho,k})$ is in $O(dk\log(km))$.*

### 3.3.1 Encoding of the input

To state the predicates, we introduce additional notation. Following [57], we define the following ***basic geometric objects***. Let $s, t \in \mathbb{R}^d$ be two points and $\Delta \in \mathbb{R}_+$ be the radius. We denote with $\ell(\overline{st})$ the line supporting $\overline{st}$. We define the stadium, cylinder and capped cylinder centered at $\overline{st}$ with radius $\Delta$ as $D_\Delta(\overline{st}) = \{x \in \mathbb{R}^d \mid \exists p \in \overline{st}, \|p - x\| \leq \Delta\}$, $C_\Delta(\overline{st}) = \{x \in \mathbb{R}^d \mid \exists p \in \ell(\overline{st}), \|p - x\| \leq \Delta\}$ and $R_\Delta(\overline{st}) = \{p + u \in \mathbb{R}^d \mid p \in \overline{st}$ and $u \in \mathbb{R}^d$ s.t. $\|u\| \leq \Delta$, and $\langle t - s, u\rangle = 0\}$. We define the hyperplane through $s$ with normal vector $\overline{st}$ as $P(\overline{st}) = \{x \in \mathbb{R}^d \mid \langle x - s, s - t\rangle = 0\}$. Let $e_1, e_2 \in \mathbb{X}_2^d$ be two edges. We define the double stadium of the edges $e_1$ and $e_2$ with radius $\Delta$ as $D_{\Delta,2}(e_1, e_2) = D_\Delta(e_1) \cap D_\Delta(e_2)$. Let $p = (p_1, p_2) \in \mathbb{R}^2$. We denote with $hr(p) = \{(x_1, x_2) \in \mathbb{R}^2 \mid x_1 \geq p_1, x_2 = p_2\}$ the horizontal ray starting at $p$.

For two polygonal curves $P \in \mathbb{R}^{dm}$ and $Q \in \mathbb{R}^{dk}$ and a radius $\Delta$, each predicate is a function mapping from $\mathbb{R}^{dm} \times \mathbb{R}^{dk+1}$ to $\{0, 1\}$ that receives the input $(P, (Q, \Delta))$. In the case of polygonal regions that may contain holes, we map from $\mathbb{R}^{3m} \times \mathbb{R}^{3k+1}$ since we add a label that associates each vertex with its boundary component. Other encodings are possible but would only influence the constant in the VC-dimension bound.

### 3.3.2 Polygonal curves

Let $P \in \mathbb{X}_m^d$ with vertices $p_1, \ldots, p_m$ and $Q \in \mathbb{X}_k^d$ with vertices $q_1, \ldots, q_k$ be two polygonal curves. Let further $\Delta \in \mathbb{R}_+$. By [57] the Hausdorff distance query $d_H(P, Q) \leq \Delta$ is uniquely determined by the following predicates:

- $(\mathcal{P}_1)$: Given an edge of $P$, $\overline{p_j p_{j+1}}$, and a vertex $q_i$ of $Q$, this predicate returns true iff there exists a point $p \in \overline{p_j p_{j+1}}$, such that $\|p - q_i\| \leq \Delta$.

- $(\mathcal{P}_2)$: Given an edge of $Q$, $\overline{q_i q_{i+1}}$, and a vertex $p_j$ of $P$, this predicate returns true iff there exists a point $q \in \overline{q_i q_{i+1}}$, such that $\|q - p_j\| \leq \Delta$.

- $(\mathcal{P}_3)$: Given an edge of $Q$, $\overline{q_i q_{i+1}}$, and two edges of $P$, $\{e_1, e_2\} \subset E(P)$, this predicate is equal to $\ell(\overline{q_i q_{i+1}}) \cap D_{\Delta,2}(e_1, e_2) \neq \emptyset$.

- $(\mathcal{P}_4)$: Given an edge of $P$, $\overline{p_j p_{j+1}}$, and two edges of $Q$, $\{e_1, e_2\} \subset E(Q)$, this predicate is equal to $\ell(\overline{p_j p_{j+1}}) \cap D_{\Delta,2}(e_1, e_2) \neq \emptyset$.

**Lemma 3.3.3** (Lemma 7.1, [57])**.** *For any two polygonal curves $P, Q$, given the truth values of all predicates of the type $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \mathcal{P}_4$ one can determine whether $d_H(P, Q) \leq \Delta$.*

By [57] the Fréchet distance query $d_F(P, Q) \leq \Delta$ is uniquely determined by the predicates $(\mathcal{P}_1)$, $(\mathcal{P}_2)$ and the following predicates:

- $(\mathcal{P}_5)$: This predicate returns true if and only if $\|p_1 - q_1\| \leq \Delta$.

- $(\mathcal{P}_6)$: This predicate returns true if and only if $\|p_m - q_k\| \leq \Delta$.

- $(\mathcal{P}_7)$: Given two vertices of $P$, $p_j$ and $p_t$ with $j < t$ and an edge of $Q$, $\overline{q_i q_{i+1}}$, this predicate returns true if there exist two points $a_1$ and $a_2$ on the line supporting the directed edge, such that $a_1$ appears before $a_2$ on this line, and such that $\|a_1 - p_j\| \leq \Delta$ and $\|a_2 - p_t\| \leq \Delta$.

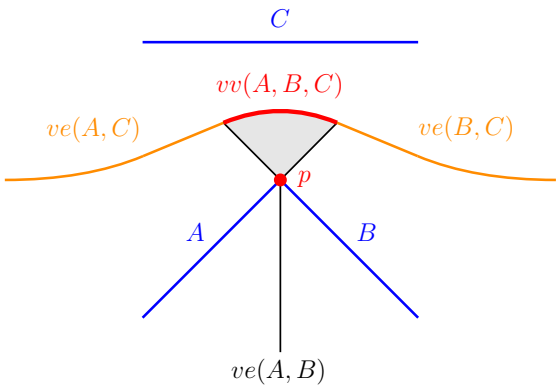


Figure 3.1: Degenerate case: $vv(A, B, C)$ consist of a whole arc and $ve(A, B)$ contains a region.

- ($\mathcal{P}_8$): Given two vertices of $Q$, $q_i$ and $q_t$ with $i < t$ and an edge of $P$, $\overline{p_j p_{j+1}}$, this predicate returns true if there exist two points $a_1$ and $a_2$ on the line supporting the directed edge, such that $a_1$ appears before $a_2$ on this line, and such that $\|a_1 - q_i\| \leq \Delta$ and $\|a_2 - q_t\| \leq \Delta$.

**Lemma 3.3.4** (Lemma 7.1, [2]). *For any two polygonal curves $P, Q$, given the truth values of all predicates of the type $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_5, \mathcal{P}_6, \mathcal{P}_7, \mathcal{P}_8$ one can determine whether $d_F(P, Q) \leq \Delta$.*

**Lemma 3.3.5** (Lemma 8.2, [57]). *For any two polygonal curves $P, Q$, given the truth values of all predicates of the type $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_5, \mathcal{P}_6$ one can determine whether $d_{wF}(P, Q) \leq \Delta$.*

### 3.3.3 Polygonal regions

In the case of polygonal regions that may contain holes, we define some of the predicates based on the Voronoi vertices of the edges of the boundary of the polygonal region. Since degenerate situations can occur if Voronoi sites intersect, we restrict the predicates to the subset of the Voronoi vertices that are relevant to our analysis.

#### Relevant Voronoi vertices

If $A, B$ and $C$ are line segments and $A$ and $B$ intersect in a point $p$, it can happen that there are Voronoi vertices in $vv(A, B, C)$ for which the closest point in $A$ and $B$ is $p$. This may result in degenerate cases where a whole arc of the Voronoi diagram consists of Voronoi vertices and a region is part of a Voronoi edge (see Figure 3.1). For our distance queries, we are only interested in extreme points of the distance to the sites. These are Voronoi vertices that are not degenerate. We define the ***relevant Voronoi vertices*** as the Voronoi vertices for which the distance of the vertices to the sites is realized by at least three distinct points, i.e.

$$rvv(A, B, C) = \left\{ p \in vv(A, B, C) \;\middle|\; \begin{array}{l} \exists a, b, c \in A, B, C \text{ s.t. } a \neq b \neq c \neq a \text{ and} \\ d_{\overrightarrow{H}}(p, a) = d_{\overrightarrow{H}}(p, b) = d_{\overrightarrow{H}}(p, c) = d_{\overrightarrow{H}}(p, A \cup B \cup C) \end{array} \right\}$$

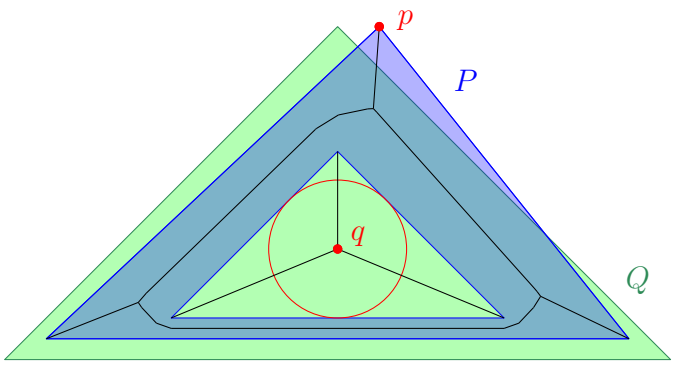


Figure 3.2: Illustration of the two cases: The point $p$ on the boundary of $P$ maximizes $d_{\overrightarrow{H}}(p, Q)$. The point $q$ in the interior of $Q$ that is a Voronoi vertex of the edges of $P$ maximizes $d_{\overrightarrow{H}}(q, P)$.

Additionally, we introduce the notion of Voronoi-vertex-candidates. Let $a = \overline{a_1 a_2}, b = \overline{b_1 b_2}$ and $c = \overline{c_1 c_2}$ be edges of a polygonal region that may contain holes. Consider their vertices and supporting lines $A = \{a_1, a_2, \ell(a)\}$, $B = \{b_1, b_2, \ell(b)\}$ and $C = \{c_1, c_2, \ell(c)\}$. Let $X \in A$, $Y \in B$ and $Z \in C$. If either $X, Y$ or $Z$ is a subset of one of the others, we set $V_0(X, Y, Z) = \emptyset$ otherwise let

$$V_0(X, Y, Z) = \{v \in \mathbb{R}^2 \mid d_{\overrightarrow{H}}(v, X) = d_{\overrightarrow{H}}(v, Y) = d_{\overrightarrow{H}}(v, Z)\}$$

be the set of points with the same distance to all sets $X, Y$ and $Z$. The ***set of Voronoi-vertex-candidates*** $V(a, b, c)$ of the line segments $a, b$ and $c$ is defined as the union over all points that have the same distance to fixed elements of $A$, $B$ and $C$, i.e.

$$V(a, b, c) = \bigcup_{X \in A, Y \in B, Z \in C} V_0(X, Y, Z).$$

Note that the Voronoi-vertex-candidates $V(a, b, c)$ contain all relevant Voronoi vertices $rvv(a, b, c)$ because a relevant Voronoi vertex $v$ has the same distance to all three edges and for each edge, the distance to $v$ is either realized at one of the endpoints of the edge or at the orthogonal projection of $v$ to the supporting line of the edge.

### Additional predicates

Let $P$ and $Q$ be two polygonal regions that may contain holes. Let further $\Delta \in \mathbb{R}_+$. In this section, we give predicates such that the Hausdorff distance query $d_H(P, Q) \leq \Delta$ is determined by them. The query depends on the two queries for the directed Hausdorff distances $d_{\overrightarrow{H}}(P, Q) \leq \Delta$ and $d_{\overrightarrow{H}}(Q, P) \leq \Delta$. We show, how to determine $d_{\overrightarrow{H}}(P, Q) \leq \Delta$, the other direction is analogous. The distance $d_{\overrightarrow{H}}(p, Q)$ for points $p \in P$ can be maximized at points in the interior of $P$ or at points at the boundary of $P$ (see Figure 3.2 for the two cases). Since these cases are different to analyze, we split the query into two general predicates.

- $(\mathcal{B})$ (Boundary): This predicate returns true if and only if $d_{\overrightarrow{H}}(\partial P, Q) \leq \Delta$.
- $(\mathcal{I})$ (Interior): This predicate returns true if $d_{\overrightarrow{H}}(P, Q) \leq \Delta$. This predicate returns false if $d_{\overrightarrow{H}}(P, Q) > d_{\overrightarrow{H}}(\partial P, Q)$ and $d_{\overrightarrow{H}}(P, Q) > \Delta$.

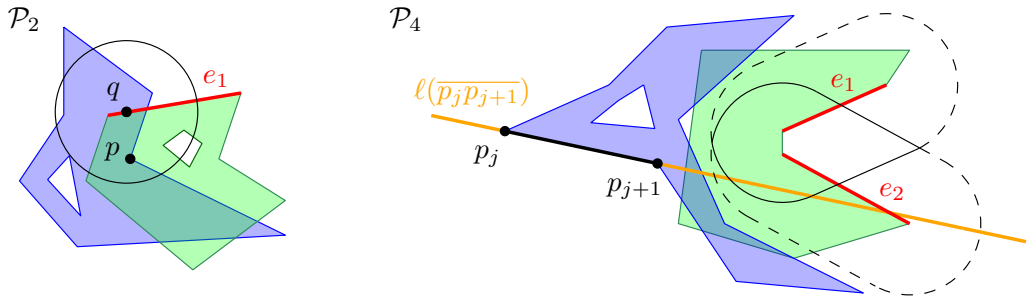


Figure 3.3: Illustration of the predicates $\mathcal{P}_2, \mathcal{P}_4$ : In both depicted cases the corresponding predicates are true.

Note that it is not defined what the predicate $(\mathcal{I})$ returns if $d_{\overrightarrow{H}}(P,Q) = d_{\overrightarrow{H}}(\partial P, Q)$ and $d_{\overrightarrow{H}}(P,Q) > \Delta$. This does not matter, since the correctness of $d_{\overrightarrow{H}}(P,Q) \leq \Delta$ is still equivalent to both $(\mathcal{B})$ and $(\mathcal{I})$ being true.

Since $(\mathcal{B})$ and $(\mathcal{I})$ are very general, we define more detailed predicates that can be used to determine feasible truth values of $(\mathcal{B})$ and $(\mathcal{I})$. To determine $(\mathcal{B})$, we need the following predicates in combination with $\mathcal{P}_2$ and $\mathcal{P}_4$ defined in Section 3.3.2:

- $(\mathcal{P}_9)$: Given a vertex $p$ of $P$, this predicate returns true if and only if $p \in Q$.
- $(\mathcal{P}_{10})$: Given an edge $e_1$ of $P$ and an edge $e_2$ of $Q$, this predicate is equal to $e_1 \cap e_2 \neq \emptyset$.
- $(\mathcal{P}_{11})$: Given a directed edge $e_1$ of $P$ and two edges $e_2$ and $e_3$ of $Q$, this predicate is true if and only if $e_1 \cap e_2 \neq \emptyset$, $e_1 \cap e_3 \neq \emptyset$ and $e_1$ intersects $e_2$ before or at the same point that it intersects $e_3$.
- $(\mathcal{P}_{12})$: Given a directed edge $e_1$ of $P$ and two edges $e_2$ and $e_3$ of $Q$, this predicate is true if and only if $e_1 \cap e_2 \neq \emptyset$ and if there exists a point $b$ on $e_3$ such that $\|a - b\| \leq \Delta$ where $a$ is the first intersection point of $e_1 \cap e_2$.
- $(\mathcal{P}_{13})$: Given a directed edge $e_1$ of $P$ and two edges $e_2$ and $e_3$ of $Q$, this predicate is true if and only if $e_1 \cap e_2 \neq \emptyset$ and if there exists a point $b$ on $e_3$ such that $\|a - b\| \leq \Delta$ where $a$ is the last intersection point of $e_1 \cap e_2$.

Using Voronoi-vertex-candidates, we define the detailed predicates for determining $(\mathcal{I})$:

- $(\mathcal{P}_{14})$: Given 4 edges $e_1, e_2, e_3, e_4$ of $Q$ and a point $v$ from the set of Voronoi-vertex-candidates $V(e_1, e_2, e_3)$, this predicate returns true if and only if there exists a point $p \in e_4$, such that $\|v - p\| \leq \Delta$.
- $(\mathcal{P}_{15})$: Given 3 edges $e_1, e_2, e_3$ of $Q$ and a point $v$ from the set of Voronoi-vertex-candidates $V(e_1, e_2, e_3)$, this predicate returns true if and only if $v \in Q$.
- $(\mathcal{P}_{16})$: Given 3 edges $e_1, e_2, e_3$ of $Q$ and a point $v$ from the set of Voronoi-vertex-candidates $V(e_1, e_2, e_3)$, this predicate returns true if and only if $v \in P$.

Examples for the predicates $\mathcal{P}_2$ and $\mathcal{P}_4$ for polygonal regions are depicted in Figure 3.3. Examples for the predicates $\mathcal{P}_9, \ldots \mathcal{P}_{16}$ are depicted in Figure 3.4.

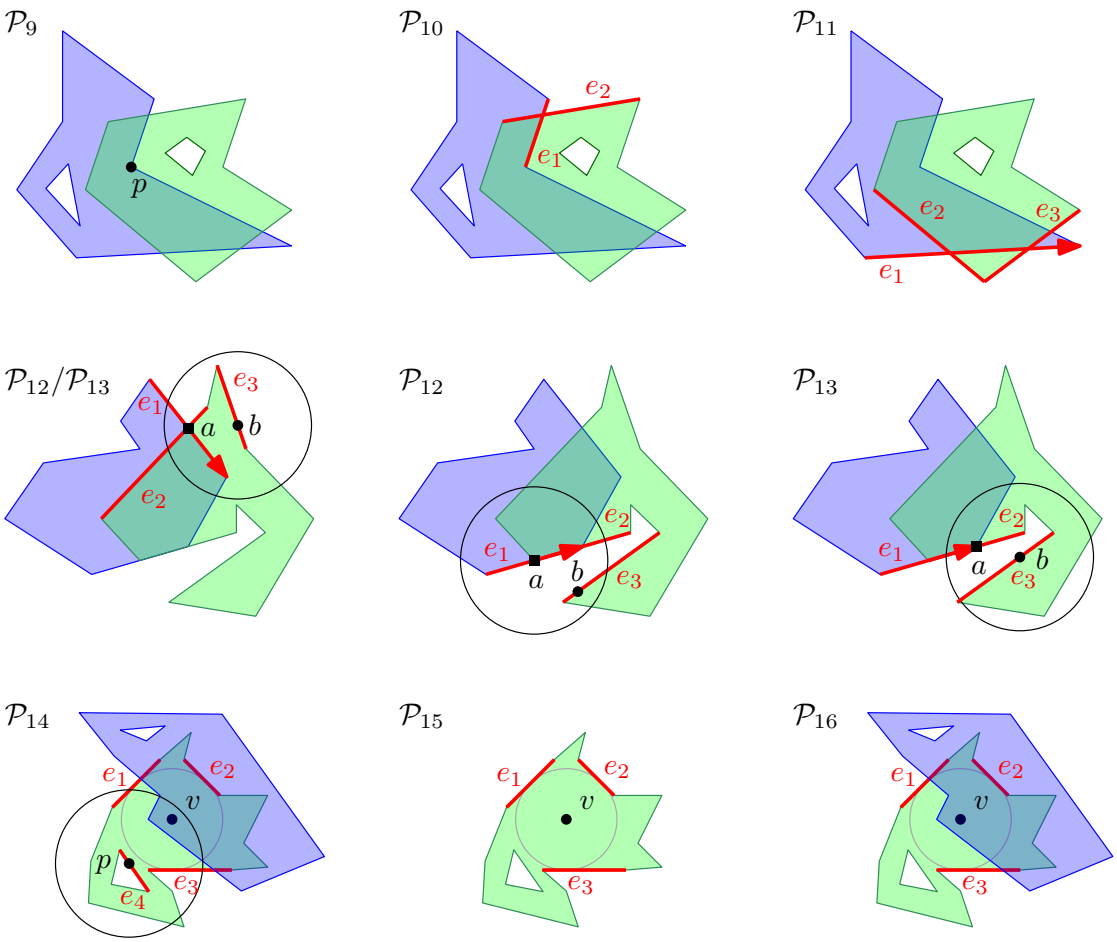


Figure 3.4: Illustration of the predicates $\mathcal{P}_9, \ldots, \mathcal{P}_{16}$ : In all depicted cases the corresponding predicates are true.

**Lemma 3.3.6.** *For any two polygonal regions $P, Q$ that may contain holes, given the truth values of all predicates of the type $\mathcal{P}_2, \mathcal{P}_4, \mathcal{P}_9, \mathcal{P}_{10}, \mathcal{P}_{11}, \mathcal{P}_{12}$ and $\mathcal{P}_{13}$ one can determine a feasible truth value for a predicate of type $(\mathcal{B})$.*

*Proof.* To determine $(\mathcal{B})$ it suffices to check for each edge $e$ of $P$ if $d_{\overrightarrow{H}}(e, Q) \leq \Delta$. If this is true for all edges, we return true, otherwise false. Let $e = \overline{uv}$ be an edge of $P$. We first determine which points of $e$ lie outside of $Q$. The $\mathcal{P}_9$ for the point $u$, tells us if $u$ lies in $Q$. Checking $\mathcal{P}_{10}$ and $\mathcal{P}_{11}$ for $e$ and all edges (respectively pairs of edges) of $Q$ then determines which edges of $Q$ get intersected and in which order they get intersected. Each intersection changes the state of the edge $e$ between lying inside and outside of $Q$. So in total, we get a sequence of intersections of edges with $Q$ where we know for each part between two consecutive intersections, if this part is inside or outside of $Q$.

Let one subset $s = \overline{s_1 s_2}$ of $e$ be given that is defined by two edges $e_1$ and $e_2$ of $Q$ that intersect $e$ consecutively. If $s$ lies inside of $Q$ then we have $d_{\overrightarrow{H}}(s, Q) = 0$. If $s$ lies outside of $Q$ then we claim that $d_{\overrightarrow{H}}(s, Q) \leq \Delta$ if and only if there exists a sequence of edges $\overline{q_{j_1} q_{j_1+1}}, \overline{q_{j_2} q_{j_2+1}}, \ldots, \overline{q_{j_t} q_{j_t+1}}$ for some integer value $t$, such that $\mathcal{P}_{13}(e, e_1, \overline{q_{j_1} q_{j_1+1}})$ and

$\mathcal{P}_{12}(e, e_2, \overline{q_{j_t} q_{j_t+1}})$ evaluate to true and the conjugate

$$\bigwedge_{i=1}^{t-1} \mathcal{P}_4(e, \overline{q_{j_i} q_{j_i+1}}, \overline{q_{j_{i+1}} q_{j_{i+1}+1}})$$

evaluates to true. The proof of this claim is analogous to the key part of the proof of Lemma 7.1 in [57]. We include a full proof of the claim here for the sake of completeness.

Assume such a sequence $\overline{q_{j_1} q_{j_1+1}}, \ldots, \overline{q_{j_t} q_{j_t+1}}$ exists. In this case, there exists a sequence of points $a_1, \ldots, a_t$ on the line supporting $s$, with $a_1 = s_1$, $a_t = s_2$, and such that for $1 \leq i < t$, $a_i, a_{i+1} \in D_\Delta(\overline{q_{j_i} q_{j_i+1}})$. That is, two consecutive points of the sequence are contained in the same stadium. Indeed, for $i = 1$, we have that there exists points $a_1, a_2$ with the needed properties since the predicates $\mathcal{P}_{13}(e, e_1, \overline{q_{j_1} q_{j_1+1}})$ and $\mathcal{P}_4(e, \overline{q_{j_1} q_{j_1+1}}, \overline{q_{j_2} q_{j_2+1}})$ evaluate to true. Likewise, for $i = t - 1$, it is implied by the corresponding $\mathcal{P}_{12}$ and $\mathcal{P}_4$ predicates and for the remaining $1 < i < t - 1$ it follows from the corresponding $\mathcal{P}_4$ predicates. Now, since each stadium is a convex set, it follows that each line segment connecting two consecutive points of this sequence $a_i, a_{i+1}$ is contained in one of the stadiums. Note that the set of line segments obtained this way forms a connected polygonal curve which fully covers the line segment $s$. It follows that

$$s \subseteq \bigcup_{0 \leq i < t} \overline{a_i a_{i+1}} \subseteq \bigcup_{0 \leq i < t} D_\Delta(\overline{q_{j_i} q_{j_i+1}})$$

Therefore, any point on $s$ is within distance $\Delta$ of some point on $Q$ and thus $d_{\overrightarrow{H}}(s, Q) \leq \Delta$

For the other direction of the proof, assume that $d_{\overrightarrow{H}}(s, Q) \leq \Delta$. Let $E(Q)$ be the set of all edges of $Q$ The definition of the directed Hausdorff distance implies that

$$s \subseteq \bigcup_{e \in E(Q)} D_\Delta(e)$$

since any point on the line segment $s$ must be within distance $\Delta$ of some point on the curve $Q$. Consider the intersections of the line segment $s$ with the boundaries of stadiums

$$s \cap \bigcup_{e \in E(Q)} \partial D_\Delta(e).$$

Let $w$ be the number of intersection points and let $l = w + 2$. We claim that this implies that there exists a sequence of edges $\overline{q_{j_1} q_{j_1+1}}, \overline{q_{j_2} q_{j_2+1}}, \ldots, \overline{q_{j_t} q_{j_t+1}}$ with the properties stated above. let $a_1 = s_2$, $a_t = s_2$ and let $a_i$ for $1 < i < t$ be the intersection points ordered in the direction of the line segment $s$. By construction, it must be that each $a_i$ for $1 < i < t$ is contained in the intersection of two stadiums, since it is the intersection with the boundary of a stadium and the entire edge is covered by the union of stadiums. Moreover, two consecutive points $a_i, a_{i+1}$ are contained in exactly the same subset of stadiums - otherwise there would be another intersection point with boundary of a stadium in between $a_i$ and $a_{i+1}$. This implies a set of true predicates of type $\mathcal{P}_4$ with the properties defined above. The predicates of type $\mathcal{P}_{12}$ and $\mathcal{P}_{13}$ follow trivially from the definitions of $s_1$, $s_2$ and the directed Hausdorff distance. This concludes the proof of the other direction.

It remains to consider the first and last parts of $e$. Let $s$ be a subset of $e$ defined by its boundaries $s_1, s_2$ where one of the boundaries is one of the vertices $u$ and $v$ of $e$ and the other boundary is the closest intersection of $e$ with an edge of $Q$ or (if this does not exist) the other vertex of $e$. The proof for this case is analogous to the previous case. It

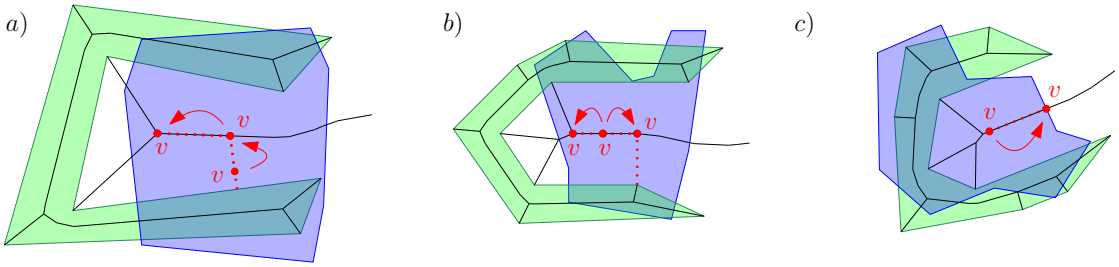


Figure 3.5: Illustration of the different cases in the proof of Lemma 3.3.7. It is demonstrated how $v$ can be moved to either increase the distance to $Q$ (a) or to stay in the same distance to $Q$ (b,c).

only needs predicates of type $\mathcal{P}_2$ for $u$ and $v$ instead of the respective predicates of type $\mathcal{P}_{13}$ and $\mathcal{P}_{12}$.

□

**Lemma 3.3.7.** *For any two polygonal regions $P, Q$, given the truth values of all predicates of the type $\mathcal{P}_{14}, \mathcal{P}_{15}$ and $\mathcal{P}_{16}$ one can determine a feasible truth value for a predicate of type $(\mathcal{I})$.*

*Proof.* We claim that, if $d_{\overrightarrow{H}}(P,Q) > d_{\overrightarrow{H}}(\partial P, Q)$ and $d_{\overrightarrow{H}}(P,Q) > \Delta$, then there has to be a point $v$ in the interior of $P$ that maximizes the Hausdorff distance to $Q$ (i.e. $d_{\overrightarrow{H}}(v,Q) = \max_{p \in P}(d_{\overrightarrow{H}}(p,Q))$) and that is a relevant Voronoi vertex of the edges of $Q$. Before we prove the claim, we show that it implies the statement of the lemma. It follows from the claim that in case of $d_{\overrightarrow{H}}(P,Q) > d_{\overrightarrow{H}}(\partial P, Q)$ and $d_{\overrightarrow{H}}(P,Q) > \Delta$ there is a relevant Voronoi vertex $v$ that lies in $P$ and outside of $Q$ with $d_{\overrightarrow{H}}(v,e) > \Delta$ for all edges $e$ of $Q$. The predicates $\mathcal{P}_{14}, \mathcal{P}_{15}$ and $\mathcal{P}_{16}$ check exactly these properties for a superset of the relevant Voronoi vertices of the edges of $Q$. So, we set $(\mathcal{I})$ to false, if and only if there is a vertex in this superset that fulfills $\mathcal{P}_{16}$, does not fulfill $\mathcal{P}_{15}$ and does not fulfill $\mathcal{P}_{14}$ for any edge of $Q$. Since all relevant Voronoi vertices are checked, $(\mathcal{I})$ will be set to false in all relevant cases. On the other hand, if we have $d_{\overrightarrow{H}}(P,Q) \leq \Delta$, then there will be no point that is in $P$, outside of $Q$ and has distance greater $\Delta$ to all edges of $Q$ and $(\mathcal{I})$ is set to true. It remains to show the claim.

We prove the claim by contradiction. We assume that $d_{\overrightarrow{H}}(P,Q) > \Delta$ and $d_{\overrightarrow{H}}(P,Q) > d_{\overrightarrow{H}}(\partial P, Q)$ and that none of the points in the interior of $P$ that maximize the Hausdorff distance to $Q$ is a relevant Voronoi vertex of the edges of $Q$. Let $v \in P$ be a point maximizing $d_{\overrightarrow{H}}(v,Q)$. Assume that $v$ lies in the Voronoi region of an edge $e$ of $Q$. Then $d_{\overrightarrow{H}}(v,Q)$ can be increased by moving $v$ in perpendicular direction away from $e$ (see Fig. 3.5a)). This would contradict that $v$ maximizies $d_{\overrightarrow{H}}(v,Q)$. So instead, assume that $v$ lies on the Voronoi edge defined by the Voronoi regions of two edges $e_1$ and $e_2$ of $Q$ and that $v$ is not a relevant Voronoi vertex. If $e_1$ and $e_2$ are not parallel, then it can be shown with a straightforward case analysis that there is a direction in which $v$ can be moved along the Voronoi diagram to increase $d_{\overrightarrow{H}}(v,Q)$ (see Fig. 3.5a)). The direction depends on the the closest points $v_1, v_2$ to $v$ on $e_1, e_2$. If $v_1$ and $v_2$ are perpendicular projections of $v$ to $e_1$ and $e_2$, then $v$ can be moved along the angle bisector of $e_1$ and $e_2$ away from the intersection of $\ell(e_1)$ and $\ell(e_2)$. If only one of $v_1$ and $v_2$ is not a perpendicular projection,

then $v$ can be moved along the parabola defined by $v_1$ and $e_2$ (or $v_2$ and $e_1$) in both directions. If both $v_1$ and $v_2$ are not a perpendicular projection, then $v$ can be moved in any direction $d$ with $\langle v - v_2, d\rangle \geq 0$ and $\langle v - v_1, d\rangle \geq 0$. If $e_1$ and $e_2$ are parallel, it can happen that locally $d_{\overrightarrow{H}}(p, Q)$ stays constant for moving $v$ along the Voronoi edge in both directions until either a relevant Voronoi vertex is reached (see Fig. 3.5b)), the boundary of $P$ is reached (see Fig. 3.5c)) or the orthogonal projection of $v$ to either one of the edges $e_1$ and $e_2$ reaches an endpoint of the respective edge (see Fig. 3.5b)). In all three cases, we get a contradiction. The first case contradicts the assumption that there is no relevant Voronoi vertex $v$ that maximizes $d_{\overrightarrow{H}}(v, Q)$, the second case contradicts the assumption that $d_{\overrightarrow{H}}(P, Q) > d_{\overrightarrow{H}}(\partial P, Q)$ and in the third case $d_{\overrightarrow{H}}(p, Q)$ would start increasing and contradict that $v$ maximizes $d_{\overrightarrow{H}}(v, Q)$. □

## 3.4 The predicates are simple

It remains to show that the predicates defined in Section 3.3.2 and 3.3.3 can be determined by a polynomial number of simple predicates. Corollary 3.3.2 then implies that all considered range spaces have $VCdim(\mathcal{R}_{\rho,k})$ in $O(dk\log(km))$.

### 3.4.1 Technical lemmas

In this section, we establish some general lemmas that will help us to show that predicates can be determined by a polynomial number of simple predicates. We first introduce some new notation.

**Definition 3.4.1.** *Let $d \in \mathbb{N}$. We call a function $f : \mathbb{R}^d \to \mathbb{R}$ **well behaved** if $f$ is a linear combination of constantly many rational functions of constant degree. Let $x_1 \in \mathbb{R}^{d_1}, \dots x_i \in \mathbb{R}^{d_i}$ with $\sum_{j=1}^{i} d_j = d$. Let $X = \{x_1, \dots x_i\}$ and $x$ be the concatenation of $x_1, \dots, x_i$. We denote $f(x)$ also with $f(x_1, \dots, x_i)$ or $f(X)$.*

For many of our predicates, we have to determine the order of two points on a line. For example, when we check if the intersections of a line with other geometric objects overlap. The following lemma shows that determining the order is simple.

**Lemma 3.4.2.** *Let $d \in \mathbb{N}$. Let $P \subset \mathbb{R}^d$ be a finite set of points and $p, q \in P$. Consider two points $v$ and $w$ on the line $\ell(\overline{pq})$ given by*

$$v = p + t_1(P)(q - p)$$
$$w = p + t_2(P)(q - p)$$

*with $t_i(P) = a_i(P) + b_i(P)\sqrt{c_i(P)}$ where $a_i, b_i$ and $c_i$ are well behaved functions for $i \in \{1, 2\}$. It is a simple predicate to determine the order of $v$ and $w$ in direction $(q - p)$.*

Note that the order in Lemma 3.4.2 can be decided by solving $t_1(P) \geq t_2(P)$. So Lemma 3.4.2 directly follows from the following more general lemma.

**Lemma 3.4.3.** *Consider the 3 inequalities*

$$a(x) \geq 0 \tag{3.1}$$
$$b\left(x, \sqrt{c(x)}, \sqrt{d(x)}\right) \geq 0 \tag{3.2}$$
$$e\left(x, \sqrt{f(x)}, \sqrt{g\left(x, \sqrt{f(x)}\right)}\right) \geq 0 \tag{3.3}$$

*for well behaved functions $a, b, c, d, e, f, g$. The following statements hold:*

1. *Inequality (3.1) is simple.*
2. *Inequality (3.2) is simple if $c(x) \geq 0$ and $d(x) \geq 0$.*
3. *Inequality (3.3) is simple if $f(x) \geq 0$ and $g(x, \sqrt{f(x)}) \geq 0$.*

Observe that the inequalities $c(x) \geq 0$, $d(x) \geq 0$ and $f(x) \geq 0$ are simple by the first statement and $g(x, \sqrt{f(x)}) \geq 0$ is simple by the second statement.

*Proof of Lemma 3.4.3.* 1. If we multiply both sides of the inequality $a(x) \geq 0$ by the square of the product of all denominators of the rational functions in $a$, then we get an equivalent inequality that only consists of a polynomial of constant degree on the left side and 0 on the right side. This inequality is by definition simple.

2. If we also here multiply both sides of the inequality $b\left(x, \sqrt{c(x)}, \sqrt{d(x)}\right) \geq 0$ by the square of the product of all denominators of the rational functions in $b$, then we get an equivalent inequality

$$b_0\left(x, \sqrt{c(x)}, \sqrt{d(x)}\right) \geq 0$$

where $b_0$ is a polynomial of constant degree. If we rearrange the terms in $b_0$ we get an equivalent inequality

$$b_1(x) + b_2(x)\sqrt{c(x)}\sqrt{d(x)} \leq b_3(x)\sqrt{c(x)} + b_4(x)\sqrt{d(x)} \tag{3.4}$$

where $b_1, b_2, b_3$ and $b_4$ are polynomials of constant degree. To show that (3.4) is simple, we first show that the sign values of both sides of (4) are determined by simple inequalities. To check the sign of the left side

$$b_1(x) + b_2(x)\sqrt{c(x)}\sqrt{d(x)} \geq 0 \tag{3.5}$$

we have to check the signs of $b_1(x)$ and $b_2(x)$. Since $b_1$ and $b_2$ are polynomials of constant degree, their signs are determined by a simple inequality. If $\operatorname{sgn}(b_1(x)) = \operatorname{sgn}(b_2(x))$ then the truth value of (3.5) is directly implied. Otherwise, we can square both sides of

$$b_2(x)\sqrt{c(x)}\sqrt{d(x)} \geq -b_1(x)$$

and (3.5) is equivalent to

$$b_2(x)^2 c(x) d(x) \geq b_1(x)^2.$$

After rearranging, this is a simple inequality because it has the same form as (3.1). The check for the sign value of the right side of (3.4) is analogous. If the sign values of the two sides differ, we get an immediate solution for the truth value of (3.4). Otherwise we square both sides and (3.4) is equivalent to

$$\begin{aligned} &b_1(x)^2 + b_2(x)^2 c(x) d(x) + 2 b_1(x) b_2(x)\sqrt{c(x)}\sqrt{d(x)} \leq \\ &b_3(x)^2 c(x) + b_4(x)^2 d(x) + 2 b_3(x) b_4(x)\sqrt{c(x)}\sqrt{d(x)} \end{aligned} \tag{3.6}$$

Multiplying both sides of (3.6) by the square of the product of all denominators of the rational functions in $c$ and $d$ and then rearranging the terms gives an equivalent inequality

$$b_5(x) + b_6(x)\sqrt{c(x)}\sqrt{d(x)} \geq 0.$$

where $b_5$ and $b_6$ are polynomials of constant degree. This inequality is simple as it has the same form as (3.5). In total, inequality (3.2) is simple as a constant combination of simple predicates.

3. The structure of the proof of the third statement is very similar to the structure of the proof of the second statement. We still include the details here for completeness.

If we multiply both sides of inequality $e\left(x, \sqrt{f(x)}, \sqrt{g\left(x, \sqrt{f(x)}\right)}\right) \geq 0$ by the square of the product of all denominators of the rational functions in $e$ and rearrange some terms, then we get an equivalent inequality

$$e_1(x) + e_2(x)\sqrt{f(x)} \leq e_3(x)\sqrt{g\left(x, \sqrt{f(x)}\right)} + e_4(x)\sqrt{f(x)}\sqrt{g\left(x, \sqrt{f(x)}\right)} \tag{3.7}$$

where $e_1, e_2, e_3$ and $e_4$ are polynomials of constant degree. We again show that the sign values of both sides of (3.7) are determined by a simple inequality. The check of the left side is analogous to checking inequality (3.5). To check the right side

$$e_3(x)\sqrt{g\left(x, \sqrt{f(x)}\right)} + e_4(x)\sqrt{f(x)}\sqrt{g\left(x, \sqrt{f(x)}\right)} \geq 0 \tag{3.8}$$

we have to check the signs of $e_3(x)$ and $e_4(x)$. Since $e_3$ and $e_4$ are polynomials of constant degree, their signs are determined by a simple inequality. If $\operatorname{sgn}(e_3(x)) = \operatorname{sgn}(e_4(x))$ then the truth value of (3.5) is directly implied. Otherwise, we can square both sides of

$$-e_3(x)\sqrt{g\left(x, \sqrt{f(x)}\right)} \leq e_4(x)\sqrt{f(x)}\sqrt{g\left(x, \sqrt{f(x)}\right)}$$

to get that (3.8) is equivalent to

$$e_3(x)^2 g\left(x, \sqrt{f(x)}\right) \leq e_4(x)^2 f(x) g\left(x, \sqrt{f(x)}\right). \tag{3.9}$$

After rearranging, this is a simple inequality because it has the same form as (3.2). If the sign values of the two sides of inequality (3.3) differ, we get an immediate solution for it. Otherwise, we get the following equivalent inequality by squaring both sides:

$$\begin{aligned} e_1(x)^2 + e_2(x)^2 f(x) + 2e_1(x)e_2(x)\sqrt{f(x)} \leq \\ e_3(x)^2 g\left(x, \sqrt{f(x)}\right) + e_4(x)^2 g\left(x, \sqrt{f(x)}\right) + e_3(x)e_4(x)\sqrt{f(x)} g\left(x, \sqrt{f(x)}\right). \end{aligned} \tag{3.10}$$

Rearranging the terms in (3.10) gives an inequality

$$e_0(x, \sqrt{f(x)}) \geq 0$$

where $e_0$ is a well behaved function. Since this inequality is a special case of inequality (3.2) it is simple. □

In general, we have to deal with different types of points as part of the predicates. We classify them in the following way.

**Definition 3.4.4.** *Let $d \in \mathbb{N}$ and $\Delta \in \mathbb{R}_+$. Let $P \subset \mathbb{R}^d$ be a finite set of points and $p, q \in P$. Let further $v \in \mathbb{R}^d$. We say that $v$ is a point of* ***root-type 1****,* ***2*** *or* ***3*** *with respect to $P$ if and only if the coordinates of $v = (v_1, \ldots, v_d)$ can be written as*

1. $v_i = a_i(P)$
2. $v_i = b_i\left(P, \sqrt{c(P)}, \sqrt{d(P)}\right)$
3. $v_i = e_i\left(P, \sqrt{f(P)}, \sqrt{g\left(P, \sqrt{f(P)}\right)}\right)$

*where $a_i, b_i, c, d, e_i, f, g$ are well behaved functions for all $i$ with $c(P) \geq 0$, $d(P) \geq 0$, $f(P) \geq 0$ and $g\left(P, \sqrt{f(P)}\right) \geq 0$.*

**Lemma 3.4.5.** *Let $a = \overline{a_1 a_2}, b = \overline{b_1 b_2}$ and $c = \overline{c_1 c_2}$ be edges of a polygonal region that may contain holes. Let $P = \{a_1, a_2, b_1, b_2, c_1, c_2\}$. All points in the set of Voronoi-vertex-candidates $V(a, b, c)$ are of root-type 1, 2 or 3 with respect to $P$. There is only a constant number of combinations of well behaved functions that can define a point in $V(a, b, c)$ (by using these functions as the well behaved functions in Definition 3.4.4). These combinations are uniquely determined by $a, b$ and $c$. For each combination, it is a simple predicate to check if it defines a point in $V(a, b, c)$.*

*Proof.* Consider the sets $A = \{a_1, a_2, \ell(a)\}$, $B = \{b_1, b_2, \ell(b)\}$ and $C = \{c_1, c_2, \ell(c)\}$. Let $X \in A$, $Y \in B$ and $Z \in C$. This combination of $X, Y$ and $Z$ only contributes points to $V(a, b, c)$ if neither of $X, Y$ and $Z$ is a subset of one of the others. This can be checked with a simple predicate by Lemma 3.4.3: To check if two points $(u_1, u_2)$ and $(v_1, v_2)$ are the same, we need to check if $u_1 = v_1$ and $u_2 = v_2$. Checking an equation $(=)$ can be done by checking both inequalities $(\leq)$ and $(\geq)$. Checking if a point $(u_1, u_2)$ lies on a line $\ell(\overline{vw})$ with $v = (v_1, v_2)$ and $w = (w_1, w_2)$ can be done by checking if there exists a $t$ such that

$$\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} + t \begin{pmatrix} w_1 - v_1 \\ w_2 - v_2 \end{pmatrix} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}.$$

This is equivalent to the following equation being true.

$$v_2 + \frac{u_1 - v_1}{w_1 - v_1}(w_2 - v_2) = u_2$$

To check if two lines $\ell(\overline{pq})$ and $\ell(\overline{vw})$ coincide, we have to check if $p$ is on the line $\ell(\overline{vw})$ as before and additionally if the lines have the same slope. This can be done by checking the truth value of $\frac{p_2 - q_2}{p_1 - q_1} = \frac{v_2 - w_2}{v_1 - w_1}$.

If the checks determine that one of $X, Y$ and $Z$ is the subset of one of the others, then all combinations of well behaved functions based on the combination $X, Y, Z$ can be ignored. Otherwise, we have 4 different cases for the types of objects $X, Y, Z$. It can be that there are 3 points, 2 points and 1 line, 1 point and 2 lines or 3 lines. Consider the biscetors of the pairs $(X, Y)$, $(X, Z)$ and $(Y, Z)$. The Voronoi-vertex-candidates are the intersections of these bisectors. It suffices to find the intersections of two bisectors, since the third one intersects the other two in the same points (by definition).

**Case 1 (3 points):** The bisector of the 3 points $u = (u_1, u_2)$, $v = (v_1, v_2)$ and $w = (w_1, w_2)$ intersect if the points are not colinear. This is just a simple predicate to check, as we have seen before by checking if $u$ lies on the line $\ell(\overline{vw})$. Assume $u$, $v$ and $w$ are not colinear. Then the bisector of $u$ and $v$ parameterized in $s$ is given by

$$\ell_1(s) = \frac{1}{2}\begin{pmatrix} u_1 + v_1 \\ u_2 + v_2 \end{pmatrix} + s \begin{pmatrix} v_2 - u_2 \\ u_1 - v_1 \end{pmatrix}$$

and the bisector of $w$ and $v$ parameterized in $t$ is given by

$$\ell_2(s) = \frac{1}{2}\begin{pmatrix} w_1 + v_1 \\ w_2 + v_2 \end{pmatrix} + t\begin{pmatrix} v_2 - w_2 \\ w_1 - v_1 \end{pmatrix}.$$

So if we set $\ell_1(s) = \ell_2(t)$, we get two linear equations with the two variable $s$ and $t$ of the form

$$f(P) + g(P)s + h(P)t = 0$$

where $f, g, h$ are well behaved functions. Since the points were not colinear, the solution for $t$ is a uniquely determined well behaved function and therefore the intersection point of the bisectors is a point of root-type 1.

**Case 2 (2 points, 1 line):** Note that a line between two points in $P$ can be written as

$$f_1(P)x + f_2(P)y + f_3(P) = 0$$

where $f_1, f_2, f_3$ are well behaved functions. The bisector between such a line and a point $(u_1, u_2)$ is given by the parabola

$$\frac{f_1(P)x + f_2(P)y + f_3(P)}{f_1(P)^2 + f_2(P)^2} = (x - u_1)^2 + (y - u_2)^2. \tag{3.11}$$

For the bisector of two points $u = (u_1, u_2)$ and $(v_1, v_2)$ parameterized in $t$, we have as before

$$x = \frac{1}{2}(u_1 + v_1) + t(v_2 - u_2) \tag{3.12}$$

$$y = \frac{1}{2}(u_2 + v_2) + t(u_1 - v_1) \tag{3.13}$$

Inserting (3.12) and (3.13) into (3.11) gives a quadratic equation in $t$ with solutions of the form

$$t_{1/2} = g_1(P) \pm \sqrt{g_2(P)}$$

where $g_1, g_2$ are well behaved functions. If $g_2(P) < 0$ then there is no intersection. This can be checked with a simple predicate by Lemma 3.4.3. Otherwise, the (up to two) intersections of the two bisectors are points of root-type 2.

**Case 3 (1 point, 2 lines):** In this case it can happen that the 2 lines are parallel. We have already shown that it can be checked by a simple predicate if the two lines have the same slope. Let the two lines be $\ell(\overline{pq})$ and $\ell(\overline{vw})$ with $p = (p_1, p_2)$, $q = (q_1, q_2)$, $v = (v_1, v_2)$ and $w = (w_1, w_2)$.

**Case 3.1 (2 lines are parallel):** The bisector of $\ell(\overline{pq})$ and $\ell(\overline{vw})$ parameterized in $t$ is given by

$$x = \frac{1}{2}(p_1 + v_1) + t(p_1 - q_1)$$
$$y = \frac{1}{2}(p_2 + v_2) + t(p_2 - q_2)$$

So analogous to Case 2, the existence of an intersection of such a bisector with a parabola of the form (3.11) is a simple predicate and if an intersection exists, the (up tp two) intersections are points of root-type 2.

**Case 3.2 (2 lines are not parallel):** The bisector of $\ell(\overline{pq})$ and $\ell(\overline{vw})$ is the union of their two angle bisectors. The angle bisectors are uniquely determined by the intersection

point of $\ell(\overline{pq})$ and $\ell(\overline{vw})$ and their slopes. Analogous to Case 1, it can be seen that the intersection of $\ell(\overline{pq})$ and $\ell(\overline{vw})$ is a point $(g_1(P), g_2(P))$ where $g_1$ and $g_2$ are well behaved functions. Let $m'$ and $m''$ be the slopes of $\ell(\overline{pq})$ and $\ell(\overline{vw})$. The angle of two lines with slope $m'$ and $m''$ is given by $\tan^{-1}(\frac{m'-m''}{1+m'm''})$. Since the angle bisectors have the same angle to both of the lines just with different sign, we get for the slope $m$ of an angle bisector that

$$\frac{m-m'}{1+mm'} = -\frac{m-m''}{1+mm''}$$

Solving this equation for $m$ gives two solutions of the form

$$m_{1/2} = g_3(P) \pm \sqrt{g_4(P)}$$

where $g_3$ and $g_4$ are well behaved functions with $g_4(P) \geq 0$. So in total the angle bisectors are given by

$$x = g_1(P) + t \tag{3.14}$$

$$y = g_2(P) + t(g_3(P)) \pm \sqrt{g_4(P)} \tag{3.15}$$

For each of the angle bisectors, inserting (3.14) and (3.15) in (3.11) gives a quadratic equation in $t$ of the form

$$t^2 h_1(P, \sqrt{g_4(P)}) + t h_2(P, \sqrt{g_4(P)}) + h_3(P, \sqrt{g_4(P)})$$

where $h_1, h_2, h_3$ are well behaved functions. The solutions for $t$ therefore have the form

$$t_{1/2} = h_4(P, \sqrt{g_4(P)}) \pm \sqrt{h_5(P, \sqrt{g_4(P)})}$$

where $h_4, h_5$ are well behaved functions. If $h_5(P, \sqrt{g_4(P)}) < 0$, then there is no intersection. This is simple by Lemma 3.4.3. Otherwise, the (up to two) intersections are points of root-type 3. In total, there can be up to four intersections because there are two angle bisectors.

**Case 4 (3 lines):** As we have seen before, all occurring bisectors are unions of (up to two) lines of the form given in (3.14) and (3.15). Note that the bisector of two parallel lines can also be realized in that way by setting $g_4(P) = 0$. Consider the intersection of two of these bisectors $\ell_1(s)$ and $\ell_2(t)$ where

$$\ell_1(s) = \begin{pmatrix} f_1(P) \\ f_2(P) \end{pmatrix} + s \begin{pmatrix} 1 \\ f_3(P) + \sqrt{f_4(P)} \end{pmatrix}$$

and

$$\ell_2(t) = \begin{pmatrix} g_1(P) \\ g_2(P) \end{pmatrix} + t \begin{pmatrix} 1 \\ g_3(P) + \sqrt{g_4(P)} \end{pmatrix}$$

with $f_{1-4}, g_{1-4}$ being well behaved functions, $f_4(P) \geq 0$ and $g_4(P) \geq 0$. If we set $\ell_1(s) = \ell_2(t)$, we get a system of two linear equations in $s$ and $t$. The system has a unique solution if $f_3(P) + \sqrt{f_4(P)} \neq g_3(P) + \sqrt{g_4(P)}$. Otherwise, there is no intersection (since the lines $X, Y, Z$ may not have the same slope). The Inequality is simple by Lemma 3.4.3. If there is a solution for $t$ it has the form

$$t = h(P, \sqrt{f_4(P)}, \sqrt{g_4(P)})$$

where $h$ is a well behaved function. So the intersection is a point of root-type 2. The proof is analogous if one or two of the considered bisectors have a minus in (3.15).

□

A reoccurring predicate is the decision if a given point is within a given distance to a given edge. We show in the following lemma that such predicates are simple.

**Lemma 3.4.6.** *Let $d \in \mathbb{N}$ and $\Delta \in \mathbb{R}_+$. Let $P \subset \mathbb{R}^d$ be a finite set of points and $p, q \in P$. Let further $v \in \mathbb{R}^d$. Let $\mathcal{P}$ be the predicate to decide if there exists a point $u \in \overline{pq}$ such that $\|u - v\| \leq \Delta$. $\mathcal{P}$ is simple if $v$ is a point of root-type 1, 2 or 3 w.r.t. $P$.*

*Proof.* The truth value of the predicate $\mathcal{P}$ can be determined by checking if $v$ is in the stadium $D_\Delta(\overline{pq})$. For this check, it suffices to check if $v$ is in at least one of $B_\Delta(p)$, $B_\Delta(q)$ and $R_\Delta(\overline{pq})$. For $B_\Delta(p)$ and $B_\Delta(q)$ ,we have to check the inequalities

$$\Delta^2 - \|v - p\|^2 \geq 0 \tag{3.16}$$

$$\Delta^2 - \|v - q\|^2 \geq 0. \tag{3.17}$$

To check if $v \in R_\Delta(\overline{pq})$ consider the closest point $s$ to $v$ on the line $\ell(\overline{pq})$. The truth value of

$$\Delta^2 - \|s - v\|^2 \geq 0 \tag{3.18}$$

uniquely determines if $v$ is in the cylinder $C_\Delta(\overline{pq})$. The truth values of

$$\|p - q\|^2 - \|p - s\|^2 \geq 0, \tag{3.19}$$

$$\|p - q\|^2 - \|q - s\|^2 \geq 0 \tag{3.20}$$

further determine if $s$ is on the edge $\overline{pq}$. So the truth values of the inequalities (3.18), (3.19) and (3.20) determine the truth value of $v \in R_\Delta(\overline{pq})$. The closest point to $v$ on the line $\ell(\overline{pq})$ is

$$s = p + \frac{(p - q)\langle (p - q), v\rangle}{\|p - q\|^2}.$$

For each coordinate of $s$, we have

$$s_j = p_j + (p_j - q_j)\frac{\sum_{i=1}^{d}(p_i - q_i)v_i}{\sum_{i=1}^{d}(p_i - q_i)^2}.$$

Note that for any two points $x, y \in \mathbb{R}^d$, we have that

$$\|x - y\|^2 = \sum_{i=1}^{d}(x_i - y_i)^2$$

is a polynomial of constant degree. So for any of the inequalities (3.16), (3.17), (3.18), (3.19) and (3.20) the following is true. If we insert all coordinates of $v$ into the inequality and rearrange the terms, we get (depending on the root-type of $v$) an equivalent inequality of one of the following types

$$h_1(P) \geq 0$$

$$h_2\left(P, \sqrt{c(P)}, \sqrt{d(P)}\right) \geq 0$$

$$h_3\left(P, \sqrt{f(P)}, \sqrt{g\left(P, \sqrt{f(P)}\right)}\right) \geq 0$$

where $c, d, f, g, h_1, h_2$ and $h_3$ are well behaved functions. By Lemma 3.4.3 all three types of inequalities are simple. So, in all three cases of different coordinates of $v$ only a constant number of simple inequalities have to be checked to determine $\mathcal{P}$. Therefore $\mathcal{P}$ is simple. □

Many of our predicates depend on the intersections of geometric objects. We address in the next lemmas that these intersections have nice properties and that the existence of these intersections can be determined by a simple predicate.

**Lemma 3.4.7.** *Let $P \subset \mathbb{R}^2$ be a finite set of points and $p = (p_1, p_2), q = (q_1, q_2) \in P$. Consider the intersection of the horizontal ray $hr(v)$ starting at $v \in \mathbb{R}^2$ and the edge $\overline{pq}$. Let $\mathcal{P}$ be the predicate to decide if $hr(v) \cap \overline{pq} \neq \emptyset$. $\mathcal{P}$ is simple if $v$ is a point of root-type 1, 2 or 3 w.r.t. $P$.*

*Proof.* To check if $hr(v)$ intersects the line $\ell(\overline{pq})$, one can first check if $p_2 - q_2 = 0$ by checking the simple inequalities $p_2 - q_2 \geq 0$ and $p_2 - q_2 \leq 0$. If this is the case, then an intersection is still possible if $v_2 - p_2 = 0$. The inequalities $v_2 - p_2 \geq 0$ and $v_2 - p_2 \leq 0$ are also simple by Lemma 3.4.3. The root-type of $v$ determines which case of the lemma to use. If $v_2 - p_2 \leq 0$ is also true, then it can be determined if $hr(v) \cap \overline{pq} \neq \emptyset$ by checking $v_1 - p_1 \geq 0$ and $v_1 - q_1 \geq 0$. These inequalities determine the relative positions of $v$ to $p$a and $q$ on the horizontal line. They are again simple by Lemma 3.4.3.

If $p_2 \neq q_2$, then the intersection of the horizontal line through $v$ and the line $\ell(\overline{pq})$ is a uniquely defined point $s = p + t(p - q)$ with $t = \frac{(v_2 - p_2)}{(p_2 - q_2)}$. In this case, it remains to check if $1 \geq t$ and $t \geq 0$ to see if $s$ lies on the edge $\overline{pq}$ and to check if $s_1 \geq v_1$ to see if $s$ lies on the right side of $v$ and is on the ray $hr(v)$. The inequalities $1 \geq t$, $t \geq 0$ and $s_1 \geq v_1$ are simple by Lemma 3.4.3 (Rearrange and choose the case of the lemma based on the root-type of $v$). □

**Lemma 3.4.8.** *Let $P \subset \mathbb{R}^2$ be a finite set of points and $p, q, u, v \in P$. Consider the intersection of the edge $\overline{pq}$ and the edge $\overline{uv}$. If the intersection exists, it is either a uniquely defined point $s$ given by*

$$s = p + t(P)(q - p)$$

*where $t$ is a well behaved function or the intersection is an edge $\overline{xy}$ with endpoints $x, y \in \{p, q, u, v\}$. Let $\mathcal{P}$ be the predicate to decide if $\overline{pq} \cap \overline{uv} \neq \emptyset$. $\mathcal{P}$ is simple. In case that the intersection is an edge, it is also a simple predicate to decide if a given pair of points $x, y \in \{p, q, u, v\}$ defines the intersection.*

*Proof.* We can write the line $\ell(\overline{pq})$ as $p + t(p - q)$ parameterized in $t$ an the line $\ell(\overline{uv})$ as $u + t'(v - u)$ parameterized in $t'$. The intersection of the lines is therefore defined by the solutions of the system of linear equations

$$p + t(p - q) = u + t'(v - u)$$

which is equivalent to

$$t(p - q) + t'(u - v) + (p - u) = 0.$$

The above is a system of two linear equations with two variables $t, t'$ of the form

$$a_i t + b_i t' + c_i = 0$$

where $a_i = (q_i - p_i)$, $b_i = (u_i - v_i)$ and $c_i = (p_i - u_i)$ for $i \in \{1, 2\}$. This system has a unique solution if $\frac{a_1}{a_2} \neq \frac{b_1}{b_2}$, no solution $\frac{a_1}{a_2} = \frac{b_1}{b_2} \neq \frac{c_1}{c_2}$ and an infinite number of solutions if $\frac{a_1}{a_2} = \frac{b_1}{b_2} = \frac{c_1}{c_2}$. Each of these equations can be checked by replacing $=$ (or $\neq$) with $\leq$ and $\geq$ and checking both inequalities. So the existence of an intersection can be checked by checking a constant number of simple inequalities.

Note that the coefficients of the linear equations are linear combinations of coordinates of points in $P$. So, if the system has a unique solution, the solution for $t$ can be written as a well behaved function with input $P$. In this case, it still remains to check $t \geq 0$ and $t \leq 1$ to see if the intersection is on the edge $\overline{pq}$. By Lemma 3.4.3, these are simple inequalities.

If the system does have an infinite number of solutions, the lines $\ell(\overline{pq})$ and $\ell(\overline{uv})$ must coincide. In this case, the solutions $t_u$ and $t_v$ of the equations $p_1 + t_u(q_1 - p_1) = u_1$ and $p_1 + t_v(q_1 - p_1) = v_1$ are uniquely determined values that can be written as well behaved functions with input $P$. Comparing $t_1, t_2, 0$ and 1 decides if the edges $\overline{pq}$ and $\overline{uv}$ intersect and which points $x, y \in \{p, q, u, v\} \subseteq P$ determine the intersection $\overline{xy}$ (if existent). Since $t_1$ and $t_2$ are well behaved functions with input $P$, each comparison is a simple predicate. □

**Lemma 3.4.9.** *Let $d \in \mathbb{N}$ and $\Delta \in \mathbb{R}_+$. Let $P \subset \mathbb{R}^d$ be a finite set of points and $p, q, v \in P$. Consider the intersection of the line $\ell(\overline{pq})$ and the ball $B_\Delta(v)$. If the intersection exists, the first and the last point of the intersection in direction $(q - p)$ are uniquely defined by*

$$s_{1,2} = p + t_{1,2}(P)(q - p)$$

*with $t_{1,2}(P) = f(P) \pm \sqrt{g(P)}$ where $f$ and $g$ are well behaved functions. Let $\mathcal{P}$ be the predicate to decide if $\ell(\overline{pq}) \cap B_\Delta(v) \neq \emptyset$. $\mathcal{P}$ is simple.*

*Proof.* We can write the line $\ell(\overline{pq})$ as $p + t(p - q)$ parameterized in $t$. The intersection of the lines is therefore defined by the solutions of

$$\begin{aligned} \|p + t(q - p) - v\|^2 \leq \Delta^2 \qquad & \iff \\ \sum_{i=1}^{d} (t(q_i - p_i) + (p_i - v_i))^2 \leq \Delta^2 & \end{aligned}$$

The inequality is equivalent to a quadratic equation of the form $t^2 + at + b \leq 0$, where

$$a = \frac{2\sum_{i=1}^{d}(p_i - v_i)(q_i - p_i)}{\sum_{i=1}^{d}(q_i - p_i)^2} \quad \text{and} \quad b = \frac{\sum_{i=1}^{d}(p_i - v_i)^2 - \Delta^2}{\sum_{i=1}^{d}(q_i - p_i)^2}.$$

We therefore have $t_{1,2} = -\frac{a}{2} \pm \sqrt{\frac{a^2}{4} - b}$ as long as $\frac{a^2}{4} - b \geq 0$. If we have $\frac{a^2}{4} - b < 0$ then the intersection is empty. By Lemma 3.4.3.1 this inequality is simple. □

**Lemma 3.4.10.** *Let $d \in \mathbb{N}$ and $\Delta \in \mathbb{R}_+$. Let $P \subset \mathbb{R}^d$ be a finite set of points and $p, q, u, v \in P$. Consider the intersection of the line $\ell(\overline{pq})$ and the capped cylinder $R_\Delta(\overline{uv})$. If the intersection exists, the first and the last point of the intersection in direction $(q - p)$ are given by*

$$s_{1,2} = p + t_{1,2}(P)(q - p)$$

*with* $t_i(P) = f_i(P) + h_i(P)\sqrt{g_i(P)}$ *where* $f_i, g_i$ *and* $h_i$ *are well behaved functions for* $i \in \{1, 2\}$*. Let* $\mathcal{P}$ *be the predicate to decide if* $\ell(\overline{pq}) \cap R_\Delta(\overline{uv}) \neq \emptyset$*.* $\mathcal{P}$ *is simple. There exists a constant number of candidates for the first and the last point that are uniquely defined by* $p, q, u, v$ *and* $\Delta$*. It is a simple predicate to decide for two of these candidates if they define the intersection.*

*Proof.* This proof of Lemma 3.4.10 is based on the proof of Lemma 7.2 in [57] that uses similar arguments. We can write the line $\ell(\overline{pq})$ as $p + t(p - q)$ parameterized in $t$ an the line $\ell(\overline{uv})$ as $u + t'(v - u)$ parameterized in $t'$. To determine the intersection of $\ell(\overline{pq})$ and $R_\Delta(\overline{uv})$: The intersection with the boundary of the infinite cylinder $C_\Delta(\overline{uv})$ and the intersections with the two limiting hyperplanes $P(\overline{uv})$ and $P(\overline{vu})$.

The intersection of $\ell(\overline{pq})$ with the boundary of $C_\Delta(\overline{uv})$ is defined by all pairs $(t, t')$ that fulfill the equality

$$\begin{aligned} \|(p + t(q - p)) - (u + t'(v - u))\| = \Delta^2 \qquad &\iff \\ \sum_{i=1}^{d}((p_i - u_i) + t(q_i - p_i) + t'(v - u))^2 - \Delta^2 = 0 \qquad &(3.21) \end{aligned}$$

For any fixed $t$ the above equation is a quadratic equation in $t'$ where the discriminant is a quadratic equation in $t$ of the form

$$a(P)t^2 + b(P)t + c(P)$$

where $a$, $b$ and $c$ are well behaved functions. If the discriminant is equal to 0, then equation (3.21) has exactly one solution. This is only the case for points on the boundary of $C_\Delta(\overline{uv})$ since the ball around such points intersects $\ell(\overline{uv})$ exactly once. Note that in case $a(P) = b(P) = c(P) = 0$ all points of $\ell(\overline{pq})$ are on the boundary of $C_\Delta(\overline{uv})$ and the intersection of the boundary of $C_\Delta(\overline{uv})$ and $\ell(\overline{pq})$ therefore consists of the whole line $\ell(\overline{pq})$. The truth value of $a(P) = b(P) = c(P) = 0$ can be checked by checking $a(P) \geq 0$, $a(P) \leq 0$, $b(P) \geq 0$, $b(P) \geq 0$, $c(P) \leq 0$ and $c(P) \leq 0$ which are simple by Lemma 3.4.3.1.

If the intersection is finite, the solutions $t = s_{1,2}$ for $a(P)t^2 + b(P)t + c(P) = 0$ define the intersection points of the boundary of $C_\Delta(\overline{uv})$ and $\ell(\overline{pq})$. We have

$$s_{1,2} = -\frac{b(P)}{2a(P)} \pm \sqrt{\frac{b(P)^2}{4a(P)^2} - \frac{c(P)}{a(P)}}$$

as long as $\frac{b(P)^2}{4a(P)^2} - \frac{c(P)}{a(P)} \geq 0$. If we have $\frac{b(P)^2}{4a(P)^2} - \frac{c(P)}{a(P)} < 0$ then the intersection is empty. By Lemma 3.4.3.1 this inequality is simple.

The intersection of $\ell(\overline{pq})$ with $P(\overline{uv})$ is given by all parameters $z \in \mathbb{R}$ such that

$$\begin{aligned} \langle p + z(q - p) - u, v - u\rangle = 0 \qquad &\iff \\ \langle p - u, v - u\rangle + z\langle q - p, v - u\rangle = 0 \qquad & \end{aligned}$$

It is possible that either the whole line intersects the plane, there is no intersection or the intersection is only one point. The truth value of $\langle p - u, v - u\rangle = 0$ tells us, if the line $\ell(\overline{pq})$ is parallel to the plane $P(\overline{uv})$ and if that is the case, the truth value of $\langle p - u, v - u\rangle = 0$ tells us if it lies on the plane. By replacing $=$ with $\leq$ and $\geq$ we can

get a constant number of simple inequalities that are equivalent to these checks (simple by Lemma 3.4.3. If the intersection is unique, it is given by the parameter

$$z_u = -\frac{\langle p-u, v-u\rangle}{\langle q-p, v-u\rangle}$$

The intersection with $P(\overline{ba})$ is analogous and we get in the case of a unique point the parameter

$$z_v = -\frac{\langle p-v, v-u\rangle}{\langle q-p, v-u\rangle}.$$

To check if the parameters $z_u$ and $z_v$ define points on $R_\Delta(\overline{uv})$, we can check

$$\|z_u - u\|^2 \leq \Delta^2 \quad \text{and} \quad \|z_v - v\|^2 \leq \Delta^2$$

which are simple by Lemma 3.4.6 where we choose $\overline{uu}$ (respectively $\overline{vv}$) as the degenerate edge that just consists of one point. Comparing $s_1, s_2, z_u$ and $z_v$ decides which points determine the intersection of $\ell(\overline{pq})$ and $C_\Delta(\overline{uv})$ (if existent). Each comparison is a simple predicate by Lemma 3.4.3.1. □

### 3.4.2 Predicates for polygonal curves

In this section we show that the predicates $\mathcal{P}_1, \ldots, \mathcal{P}_8$ are simple.

**Lemma 3.4.11.** *For any two polygonal curves $P \in \mathbb{X}_m^d, Q \in \mathbb{X}_k^d$ and a radius $\Delta \in \mathbb{R}_+$, each of the predicates of type $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \mathcal{P}_4$ is simple (as a function mapping from $\mathbb{R}^{dm} \times \mathbb{R}^{dk+1}$ to $\{0,1\}$ that gets the input $(P,(Q,\Delta))$).*

*Proof.* For $\mathcal{P}_1, \mathcal{P}_2$ this statement directly follows from Lemma 3.4.6. Let $\mathcal{P}$ be a predicate of type $\mathcal{P}_3$ or $\mathcal{P}_4$ with input $((P,Q),\Delta)$. $\mathcal{P}$ can be determined by checking if a line $\ell(\overline{pq})$ intersects a double stadium $D_{\Delta,2}(\overline{uv}, \overline{xy})$ for some points $p, q, u, v, x, y \in P \cup Q$. For $\mathcal{P} = \mathcal{P}_3$, we have $\overline{pq} = \overline{q_i, q_{i+1}}$ and for $\mathcal{P} = \mathcal{P}_4$, we have $\overline{pq} = \overline{p_j, p_{j+1}}$. In both cases, we have $\overline{uv} = e_1$ and $\overline{xy} = e_2$. The truth value of $\ell(\overline{pq}) \cap D_{\Delta,2}(\overline{uv}, \overline{xy}) \neq \emptyset$ can be determined with the help of the intersection of $\ell(\overline{pq})$ with $B_\Delta(u), B_\Delta(v), B_\Delta(x), B_\Delta(y), R_\Delta(\overline{uv})$ and $R_\Delta(\overline{xy})$. If and only if there is an overlap of the intersection of $\ell(\overline{pq})$ with any of these geometric objects belonging to the first stadium and the intersection of $\ell(\overline{pq})$ with any of these geometric objects belonging to the second stadium, then the predicate is true. By Lemma 3.4.9 and Lemma 3.4.10, it is a simple predicate to check which of these intersections exists and it can be decided with the help of a constant number of simple predicates which candidates define each of the intersections. All candidates for intersection points have the form

$$v = p + t(P \cup Q)(q-p)$$

with $t(P \cup Q) = f(P \cup Q) + h(P \cup Q)\sqrt{g(P \cup Q)}$ where $f, g$ and $h$ are well behaved functions. So by Lemma 3.4.2, the order of two candidates along $\ell(\overline{pq})$ is decided by a simple predicate. Comparing the order of all pairs of candidates determines the order of all candidates along the line. Together with the information on which intersections exist and which candidates determine the intersections, one can decide if $\ell(\overline{pq}) \cap D_{\Delta,2}(\overline{uv}, \overline{xy}) \neq \emptyset$. Since this information is given by a constant number of simple predicates, the whole predicate $\mathcal{P}$ is simple. □

**Lemma 3.4.12.** *For any two polygonal curves $P \in \mathbb{X}_m^d, Q \in \mathbb{X}_k^d$ and a radius $\Delta \in \mathbb{R}_+$, each of the predicates of type $\mathcal{P}_5, \mathcal{P}_6, \mathcal{P}_7, \mathcal{P}_8$ is simple (as a function mapping from $\mathbb{R}^{dm} \times \mathbb{R}^{dk+1}$ to $\{0,1\}$ that gets the input $(P,(Q,\Delta))$).*

*Proof.* For $\mathcal{P}_5, \mathcal{P}_6$ this directly follows from Lemma 3.4.6 if we interpret points $q_1$ and $q_k$ in $\mathcal{P}_5$ and $\mathcal{P}_6$ as degenerate edges $\overline{q_1 q_1}$ and $\overline{q_k q_k}$. Let $\mathcal{P}$ be a predicate of type $\mathcal{P}_7$ or $\mathcal{P}_8$ with input $((P,Q),\Delta)$. The truth value of $\mathcal{P}$ can be determined by checking if there is an intersection of a line segment $\overline{pq}$ with the intersection of two balls $B_\Delta(u)$ and $B_\Delta(v)$. For $\mathcal{P} = \mathcal{P}_7$, we have $\overline{pq} = \overline{q_i, q_{i+1}}$, $u = p_j$ and $v = p_t$. For $\mathcal{P} = \mathcal{P}_8$, we have $\overline{pq} = \overline{p_j, p_{j+1}}$, $u = q_i$ and $v = q_t$. To answer the predicate, one can compute the intersections of the line $\ell(\overline{pq})$ with each of the balls $B_\Delta(u)$ and $B_\Delta(v)$ and then check if they overlap. The remainder of the proof is analogous to the proof of Lemma 3.4.11 since it just has to be checked if two intersections overlap. □

### 3.4.3 Predicates for polygonal regions that may contain holes

In the following, we show that each of the predicates $\mathcal{P}_9, \ldots, \mathcal{P}_{16}$ is either simple or a combination of a polynomial number of simple predicates.

**Lemma 3.4.13.** *For any two polygonal regions $P \in (\mathbb{R}^{2+1})^m$ and $Q \in (\mathbb{R}^{2+1})^k$ that may contain holes and a radius $\Delta \in \mathbb{R}_+$, each of the predicates of type $\mathcal{P}_{10}, \mathcal{P}_{11}, \mathcal{P}_{12}, \mathcal{P}_{13}$ and $\mathcal{P}_{14}$ is simple (as a function mapping from $\mathbb{R}^{3m} \times \mathbb{R}^{3k+1}$ to $\{0,1\}$ that gets the input $(P,(Q,\Delta))$).*

*Proof.* Let $\mathcal{P}$ be a predicate with input $((P,Q),\Delta)$. If $\mathcal{P}$ is of type $\mathcal{P}_{10}$, then it directly follows by Lemma 3.4.8 that $\mathcal{P}$ is simple. If $\mathcal{P}$ is of type $\mathcal{P}_{11}$ then it is a simple predicate to check (Lemma 3.4.8) if the two intersections exist and as described in Lemma 3.4.2, it needs only a constant number of simple predicates to determine the order of the intersections (if existent). If $\mathcal{P}$ is of type $\mathcal{P}_{12}$ or $\mathcal{P}_{13}$, then it is a simple predicate (Lemma 3.4.8) to check if the two intersections exist and which points are the first and the last points of the intersection (if existent). Since all candidates for the first and last point are of root-type 1, the distance of each of the candidates to the edge $e_3$ can be checked with a simple predicate by Lemma 3.4.6. If $\mathcal{P}$ is of type $\mathcal{P}_{14}$ then it directly follows by Lemma 3.4.6 that $\mathcal{P}$ is simple because all Voronoi-vertex-candidates are vertices of root-type 1, 2 or 3 by Lemma 3.4.5. □

**Lemma 3.4.14.** *For any two polygonal regions $P \in (\mathbb{R}^{2+1})^m$ and $Q \in (\mathbb{R}^{2+1})^k$ that may contain holes and a radius $\Delta \in \mathbb{R}_+$, each of the predicates of type $\mathcal{P}_9, \mathcal{P}_{15}, \mathcal{P}_{16}$ can be determined by a polynomial number (with respect to $k$ and $m$) of simple predicates (which are functions mapping from $\mathbb{R}^{3m} \times \mathbb{R}^{3k+1}$ to $\{0,1\}$ that get the input $(P,(Q,\Delta))$).*

*Proof.* Let $\mathcal{P}$ be a predicate of type $\mathcal{P}_9, \mathcal{P}_{15}$ or $\mathcal{P}_{16}$ with input $((P,Q),\Delta)$. The truth value of $\mathcal{P}$ can be determined by checking if a vertex $v$ is contained in a polygonal region $A \in \{P,Q\}$. In all cases, $v$ is a point of root-type 1, 2 or 3 (see Lemma 3.4.5). Consider the following two types of predicates.

- $(\mathcal{P}')$ : Given an edge $e$ of $A$, this predicate returns true if and only if $hr(v) \cap e \neq \emptyset$.
- $(\mathcal{P}'')$ : Given a vertex $a$ of $A$, this predicate returns true if and only if $hr(v) \cap a \neq \emptyset$.

Knowing all of these predicates can determine how many times the horizontal ray $hr(v)$ crosses the boundary of $A$. If $hr(v)$ crosses the boundary an even amount of times, then $v \notin A$ and for an odd amount of times, we have $v \in A$. The vertices have to be considered in $\mathcal{P}''$ to not count any intersection twice. Each predicate of the form $\mathcal{P}'$ or $\mathcal{P}''$ is simple by Lemma 3.4.7 (interpret a vertex $a$ as a degenerate edge $\overline{aa}$). Since there are only a polynomial number of predicates of the form $\mathcal{P}'$ and $\mathcal{P}''$ we have that $\mathcal{P}$ can be determined by a polynomial number of simple predicates. □

### 3.4.4 Putting everything together

In the previous sections, it was shown that all predicates for all analyzed range spaces of the form $\mathcal{R}_{\rho,k}$ can be determined by a polynomial number of simple predicates. Together with Corollary 3.3.2, this implies our following main results.

**Theorem 3.4.15.** *Let $\mathcal{R}_{d_H,k}$ be one of the following range spaces under the Hausdorff distance: Either the range space of balls centered at polygonal curves in $\mathbb{X}_k^d$ with ground set $\mathbb{X}_m^d$ or the range space of balls centered at polygonal regions that may contain holes in $(\mathbb{R}^{2+1})^k$ with ground set $(\mathbb{R}^{2+1})^m$. In the case of polygonal curves $VCdim(\mathcal{R}_{d_H,k})$ is in $O(dk\log(km))$ and in the case of polygonal regions $VCdim(\mathcal{R}_{d_H,k})$ is in $O(k\log(km))$.*

**Theorem 3.4.16.** *Let $\mathcal{R}_{\rho,k}$ be the range space of balls under distance measure $\rho$ centered at polygonal curves in $\mathbb{X}_k^d$ with ground set $\mathbb{X}_m^d$. Let $\rho$ be either the Fréchet distance ($\rho = d_F$) or the weak Fréchet distance ($\rho = d_{wF}$). In both cases $VCdim(\mathcal{R}_{\rho,k})$ is in $O(dk\log(km))$.*

*Proof of Theorems 3.4.15, 3.4.16.* The number of predicates of each type $\mathcal{P}_1, \dots \mathcal{P}_{13}$ is polynomial in $k$ and $m$. By Lemma 3.3.3, 3.3.4, 3.3.5, 3.3.6 and 3.3.7 the relevant distance queries are determined by the truth values of these predicates. Furthermore Lemma 3.4.11, 3.4.12, 3.4.13 and 3.4.14 imply that all these predicates are determined by a polynomial number (with respect to $m$ and $k$) of simple predicates. Therefore, applying Corollary 3.3.2 directly results in the claimed bounds on the VC-dimension. □

# Chapter 4

# Subtrajectory Clustering

The main content of this chapter previously appeared as the paper *Subtrajectory Clustering: Finding Set Covers for Set Systems of Subcurves* [8] by Hugo Akitaya, Frederik Brüning, Erin Chambers and Anne Driemel which was published in *Computing in Geometry and Topology: Volume 2(1), 2023* and is also available on arXiv [6]. An initial version of the work has been presented at the *37th European Workshop on Computational Geometry (EuroCG 2021)* [7] based on an extended abstract without formal publication. This chapter extends the algorithm from [8] by a slight improvement of the VC-dimension analysis in Section 4.5.3 that reduces the bound on the VC-dimension from $O(d^2\ell^2 \log(d\ell))$ to $O(d\ell \log(\ell))$ based on the techniques from Chapter 3. The improved bound together with the use of Theorem 2.4.6 instead of Theorem 2.4.3 for generating $\varepsilon$-nets results in an improved solution size of $O(kd\ell \log(k) \log(\ell))$ instead of the original $O(k\delta \log(k\delta))$ with $\delta = O(d^2\ell^2 \log(d\ell))$.

## 4.1 Introduction

In this and the following chapter, we study subtrajectory clustering under the Fréchet distance. In particular, we design bicriterial approximation algorithms for the $(\Delta, \ell)$-covering problem that was introduced in Section 2.5. While this chapter contains our initial approach, the next one improves on the results. Both are based on the multiplicative weight update method described in Section 2.4.2.

### 4.1.1 Organization

We give an overview of our main results in Section 4.1.2. We then discuss our main techniques in Section 4.2. Using these techniques, we develop solutions to the discrete and the continuous variants of the $(\Delta, \ell)$-covering problem. Sections 4.3 and 4.4 contain our solutions to the discrete variant of the problem and Section 4.5 contains our solution to the continuous variant of the problem. In Sections 4.6 we discuss additional results on the VC-dimension of the underlying range space.

### 4.1.2 Main results

We study the problem of subtrajectory clustering in the concrete form of the $(\Delta, \ell)$-covering problem as defined in Section 2.5. We think that this problem formulation provides a natural formalization of the problem as it is studied in many applications (see

also the discussion in Section 7.2). We develop bicriterial approximation algorithms for this problem, where the approximation is with respect to the following two criteria

(i) the number of clusters $k$, and

(ii) the radius of the clustering $\Delta$.

In Sections 4.3 and 4.4 we describe our approach for the discrete variant of the subtrajectory clustering problem defined in Section 2.5.1, before we turn to the main problem in Section 4.5. We first discuss the special case where cluster centers are restricted to be directed line segments (the case $\ell = 2$). In this case, we get the following result.

**Theorem 4.1.1.** *Let $P : [0,1] \to \mathbb{R}^d$ be a polygonal curve of complexity $n$ with breakpoints $0 \leq t_1, \ldots, t_m \leq 1$ and let $\Delta > 0$ be a parameter. Assume there exists a set $C^* \subset \mathbb{X}_2^d$ of size $k \leq m$, such that $\phi(P, C^*) \leq \Delta$. There exists an algorithm that computes a set $C \subset \mathbb{X}_2^d$ of size $O(k \log^2(m))$ such that $\phi(P, C) \leq 6\Delta$. The algorithm has expected running time in $\widetilde{O}\left(km^2 + mn\right)$ and uses space in $O(n + m^2)$.*

The main idea is to define a suitable range space that preserves optimal solutions up to approximation and at the same time allows for efficient range space oracles. A range space oracle is a data structure that answers queries with a set $r$ and an element of the ground set $x$ and returns whether $x \in r$. We solve this by defining a linear number of "proxy" curves which are simplifications of subcurves that are locally maximal. The proxy curves allow to solve a range query by computing a partial Fréchet distance with some additional conditions. In the more general case, where cluster centers can be curves of complexity $\ell > 2$, we use the bi-criterial simplification algorithm of Agarwal, Har-Peled, Mustafa and Wang [4] to define suitable proxy curves. This is described in Section 4.4 and leads to the following result.

**Theorem 4.1.2.** *Let $P : [0,1] \to \mathbb{R}^d$ be a polygonal curve of complexity $n$ with breakpoints $0 \leq t_1, \ldots, t_m \leq 1$. Assume there exists a set $C^* \subset \mathbb{X}_\ell^d$ of size $k \leq m$, such that $\phi(P, C^*) \leq \Delta$. Then there exists an algorithm that computes a set $C \subset \mathbb{X}_\ell^d$ of size $O(k \log^2(m))$ such that $\phi(P, C) \leq 50\Delta$. The algorithm has expected running time in $\tilde{O}\left(k\ell^2 m^2 + mn\right)$ and uses space in $O(n + m\ell + m^2)$.*

Finally, in Section 4.5, we present our solution to the main problem of subtrajectory clustering, where subtrajectories can start and end at any two points along the curve (see Section 2.5). We use the techniques developed in Section 4.4, but we obtain better approximation factors and running times, compared to a naive application of Lemma 2.5.2. The improved running time results from the fact that we do not need to keep track of breakpoints explicitly in the range space oracle. Crucial to obtaining better approximation factors is the analysis of the VC-dimension of the dual range space. We obtain the following theorem.

**Theorem 4.1.3** (Main Theorem)**.** *Let $P : [0,1] \to \mathbb{R}^d$ be a polygonal curve of complexity $n$, let $\ell \in \mathbb{N}$ and $\Delta > 0$ be parameters. Let $k$ be the minimum size of a solution to the $(\Delta, \ell)$-covering problem on $P$. Let further $\lambda(P)$ be the arc length of the curve $P$. There exists an algorithm that outputs a $(19, O(\ell \log(k) \log(\ell)))$-approximate solution. Let $m = \left\lceil \frac{\lambda(P)}{\Delta} \right\rceil$. The algorithm has expected running time in $\widetilde{O}\left(k\ell^3 m^2 + mn\right)$ and uses space in $O(n + m\ell)$.*

In particular, in the above theorem, when the complexity of center curves $\ell$ is constant, the VC-dimension $\delta$ is constant, and the approximation factor for the size of the set cover is $O(\log k)$. Note that we assume that $d$ is constant as stated in the problem definition in Section 2.5. For a comparison, using Theorem 4.1.2 and Lemma 2.5.2 directly would result in an approximation factor of $O(\log^2(\frac{\lambda}{\Delta}))$ which could be large even if $\ell$ and $d$ are small. We summarize our results in Table 4.1.

| Size $k'$ | $\Delta'$ | Running time | Space | Setting | Reference |
|---|---|---|---|---|---|
| $O(k\log^2(m))$ | $6\Delta$ | $\widetilde{O}\left(km^2+mn\right)$ | $O(n+m^2)$ | $\ell=2$, discr. | Thm. 4.1.1 |
| $O(k\log^2(m))$ | $50\Delta$ | $\tilde{O}\left(km^2+mn\right)$ | $O(n+m^2)$ | $\ell\geq 1$, discr. | Thm. 4.1.2 |
| $O(k\log(k))$ | $19\Delta$ | $\tilde{O}(k\lceil\frac{\lambda}{\Delta}\rceil^2+\lceil\frac{\lambda}{\Delta}\rceil n)$ | $O(n+\lceil\frac{\lambda}{\Delta}\rceil)$ | $\ell\geq 1$, contin. | Thm. 4.1.3 |

Table 4.1: For optimal $C\subset(\mathbb{R}^d)^\ell$ of size $k$, covering $P\in(\mathbb{R}^d)^n$ under distance $\Delta$, we design bicriteria-approximation algorithms that compute $C'\subset\mathbb{X}_{\ell'}^d$ of size $k'$, covering $P$ under distance $\Delta'$. Here, we assume that $\ell$ and $d$ are constant, $n$ is the complexity of $P$ and $\lambda$ is the arclength of $P$.

The improved approximation factors that are obtained in the continuous case in Theorem 4.1.3 raise the question if the approximation factor could be improved in the discrete case. Unfortunately, this does not seem to be the case. In Section 4.6 we study lower bounds to the VC-dimension for two natural problem variants. We study the dual range space (i) in the discrete case and (ii) the range space directly corresponding to our main clustering problem. For (i) we show a lower bound of $\Omega(\log m)$ directly corresponding to the upper bound, see Theorem 4.6.3. For (ii) we show that—surprisingly—it inherently depends on the number of vertices of the input curve $n$, even when cluster centers are restricted to be line segments, see Theorem 4.6.2 for the exact statement. Thus, ultimately, our modified range space with proxy curves not only makes the algorithm faster, but also has the benefit of a significantly lower VC-dimension, compared to the exact range space inherent to the problem.

In addition to the results in this thesis, we also investigate the question of hardness for the discrete problem defined, see the paper [8] for details. If the complexity of center curves $\ell$ can be large, then NP-hardness follows from the hardness of the shortest common superstring problem, see also the result by Buchin, Driemel, Gudmundsson, Horton, Kostitsyna, Löffler and Struijs [35] on $(k,\ell)$-center clustering under the Fréchet distance. In particular, in this case, the problem is also hard to approximate. We show that even if we require cluster centers to be points by setting $\ell=1$, the problem remains NP-hard, via a reduction from Planar-Monotone-3SAT.

## 4.2 Setup of techniques

In this section we introduce the main ideas and concepts that we use in our algorithms.

We start in Section 4.2.1 with a simple algorithm that illustrates our general approach in a nutshell: we derive an auxiliary range space that has a simpler structure and smaller size compared to the range space of Section 2.5.1, while preserving optimal solutions up to approximation. A preliminary result that follows by applying the greedy set cover algorithm is stated in Theorem 4.2.2. Then, in Section 4.2.2 we adapt the multiplicative

weight update method for hitting sets described in Section 2.4.2 to get an efficient algorithm. The approximation quality of the resulting algorithm strongly depends on the VC-dimension of the dual range space. Therefore, we aim for auxiliary range spaces with constant VC-dimension. This is not always possible when breakpoints are given with the input. We discuss this in Section 4.2.3.

### 4.2.1 A range space for approximation

In this section, we discuss a simple algorithmic solution to the discrete variant of the problem we study. We emphasize that the approach works for any choice of breakpoints and is thus interesting in its own right. The algorithm yields a bicriteria approximation in the radius $\Delta$ and the number of clusters $k$. Although this algorithm is suboptimal, we include it here as an illustration of our general approach to the subtrajectory clustering problem: modify the range space in a way that preserves the initial structure up to approximation but allows for more efficient algorithms for the clustering problem.

To modify the range space, we use simplifications. A curve $Q \in \mathbb{X}_\ell^d$ is called an ***$\ell$-simplification*** of a curve $P$ if its Fréchet distance to $P$ is minimum among all curves in $\mathbb{X}_\ell^d$. We denote with $T_S(n,\ell)$ the time needed to compute such an $\ell$-simplification for a polygonal curve of $n$ vertices. Let $\mathcal{S} = \{(i,j) \in \mathbb{N}^2 \mid 1 \le i < j \le m\}$. For any $(i,j) \in \mathcal{S}$ let $\mu_\ell(P[t_i,t_j])$ denote the $\ell$-simplification of the corresponding subcurve of $P$. Consider a range space $\widetilde{\mathcal{R}}_0$ defined on the ground set $X = \{1,\dots,m-1\}$, where each set $r_{i,j} \in \widetilde{\mathcal{R}}_0$ is defined by a tuple $(i,j) \in \mathcal{S}$ and is of the form

$$r_{i,j} = \{z \in X \mid \exists i' \le z < j' \text{ with } d_F(P[t_{i'},t_{j'}], \mu_\ell(P[t_i,t_j])) \le 3\Delta\}$$

We will see (Lemma 4.2.1, below), that $\widetilde{\mathcal{R}}_0$ approximates the structure of $\mathcal{R}$ as defined in (2.1) to the extent that a set cover for $\widetilde{\mathcal{R}}_0$ corresponds to an approximate solution for our clustering problem. The well-known greedy set cover algorithm, which incrementally builds a set cover by taking the set with the largest number of still uncovered elements in each step, yields an $O(\log m)$ approximation for a ground set of size $m$ [44]. Applying this algorithm to the range space $\widetilde{\mathcal{R}}_0$, we obtain a set $C$ consisting of $\ell$-simplifications $\mu_\ell(P[t_i,t_j]))$ for each $r_{i,j}$, such that $\phi(P,C) \le 3\Delta$.

**Building the incidence matrix** To this end, we compute the binary incidence matrix $M$ of the range space $\widetilde{\mathcal{R}}_0$ explicitly in $O(m^3(n\ell+m) + m^2 T_S(n,\ell))$ time, as follows. Initially, we set all entries of the matrix to 0. In the first step, we compute the $O(m^2)$ simplifications $\mu_\ell(P[t_i,t_j])$ of all subcurves between two breakpoints. For each simplification $\mu$, we compute the $\Delta$-free space with the curve $P$, which was defined in Section 2.3.2 as the level set

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

Computing the associated diagram can be done in $O(n\ell)$ time and space [13]. Note that the simplification $\mu$ corresponds to the vertical axis of the $\Delta$-free space diagram and $P$ corresponds to the horizontal axis. Now, for each breakpoint $t_{i'}$ we compute the maximal breakpoint $t_{j'}$ that is reachable by a monotone path from the bottom of the diagram at $(t_{i'},0)$ to the top of the diagram at $(t_{j'},1)$. This can be done in $O(n\ell)$ time using standard techniques [13]. For all $i' \le q < j'$, we set the entry corresponding to $q$ and $\mu$ to 1. This takes $O(m)$ time. We do this for all simplifications. After that, each entry of $M$ is 1 if the corresponding element is contained in the corresponding set and 0 otherwise.

**Applying greedy set cover** We initially scan the incidence matrix to compute the number of uncovered elements $n_{i,j}$ for every range $r_{i,j} \in \widetilde{\mathcal{R}}_0$. After this, we can compute the set with the highest number of uncovered elements in $O(m^2)$ time. Then, we can update all $n_{i,j}$ on the fly every time we select a new set for the set cover. To do so, we scan for each newly covered element all the $m^2$ entries of the incidence matrix corresponding to this element and reduce $n_{i,j}$ by 1 if the entry corresponding to $r_{i,j}$ is equal to 1. Since each of the $m$ elements gets covered for the first time only once, this can be done in a total time of $O(m^3)$.

**Lemma 4.2.1.** *For any $r_Q \in \mathcal{R}$, there is a $r_{i,j} \in \widetilde{\mathcal{R}}_0$ such that $r_Q \subseteq r_{i,j}$.*

*Proof.* Let $Y$ be the set of tuples $(i,j) \in \mathbb{N}^2$ with $1 \leq i < j \leq m$ and $d_F(Q, P[t_i, t_j]) \leq \Delta$. We have that $r_Q = \bigcup_{(i,j)\in Y} [i,j) \cap \mathbb{N}$. Let $(i,j), (i',j') \in Y$. Using the triangle inequality, we can upper bound $d_F(P[t_{i'}, t_{j'}], \mu_\ell(P[t_i, t_j]))$ by

$$d_F(P[t_{i'}, t_{j'}], Q) + d_F(Q, P[t_i, t_j]) + d_F(P[t_i, t_j], \mu_\ell(P[t_i, t_j])) \leq 3\Delta.$$

By the definition of $r_{i,j}$, we have $[i',j') \cap \mathbb{N} \subseteq r_{i,j}$ and therefore $r_Q \subseteq r_{i,j}$. In other words, we can choose any maximal set of covered intervals within $r_Q$ and use the simplification of the corresponding subcurve of $P$ to cover all parts of $P$ that are covered by $Q$. □

**Theorem 4.2.2.** *Given a polygonal curve $P : [0,1] \to \mathbb{R}^d$ with breakpoints $0 \leq t_1, \ldots, t_m \leq 1$. Assume there exists a set of curves $C^* \subset \mathbb{X}_\ell^d$ of size $k$, such that $\phi(P, C^*) \leq \Delta$. There exists an algorithm that computes a set $C \subset \mathbb{X}_\ell^d$ of size $O(k \log m)$ and has running time in $O(m^3 n\ell + m^4 + m^2 T_S(n, \ell)))$ such that $\phi(P, C) \leq 3\Delta$, where $T_S(n, \ell)$ denotes the the running time for computing an $\ell$-simplification of a polygonal curve of $n$ vertices.*

*Proof.* The algorithm builds the incidence matrix of the range space and applies greedy set cover, as described above. The bound of the running time is immediate. It remains to argue correctness. The existence of a set of curves $C^*$ of size $k$ with $\phi(P, C^*) \leq \Delta$ implies that there exists a set cover of $\mathcal{R}$ of size $k$. Lemma 4.2.1 implies that for any set cover of $\mathcal{R}$, there exists a set cover of $\widetilde{\mathcal{R}}_0$ of the same size. Thus, the $O(\log m)$-approximate set cover $S$ computed by the algorithm for $\widetilde{\mathcal{R}}_0$ has size at most $O(k \log m)$. Let

$$C = \{\mu_\ell(P[t_i, t_j]) \mid r_{i,j} \in S\}.$$

Since $S$ is a set cover for $\widetilde{\mathcal{R}}_0$, and by the definition of $r_{i,j}$, we have $\phi(P, C) \leq 3\Delta$. □

### 4.2.2 Adaptation of the multiplicative weight update method

For obtaining our main result, we use the multiplicative weight update method for hitting sets described in Section 2.4.2. To solve the set cover problem, we compute an approximately minimal hitting set of the dual range space. Let $\mathcal{R}$ be a range space with ground set $X$. Think of $\mathcal{R}$ as the dual range space for which we want to compute a hitting set. To apply Theorem 2.4.9 directly, we need to specify how the data structures for the verifier and for sampling are implemented. In the following, we assume that we have a range space oracle $\mathcal{O}$ for $\mathcal{R}$.

**Definition 4.2.3** (Range space oracle)**.** *For a given range space $\mathcal{R}$ with ground set $X$ a **range space oracle** is a data structure $\mathcal{O}$ that can be queried with any $r \in \mathcal{R}$ and $z \in X$ and answers whether $z \in r$. We denote with $P_{\mathcal{O}}$ the preprocessing time to build the data structure $\mathcal{O}$ for the oracle and with $T_{\mathcal{O}}$ the time needed to answer the query. We denote with $S_{\mathcal{O}}$ the space required by the data structure.*

The verifier needs to decide for a query set $H \subseteq X$ if $H$ is a hitting set and return a nonempty set $r$ of $\mathcal{R}$ such that $r \cap H = \emptyset$ if it is not. After preprocessing the oracle, the verifier can be implemented to run in $O(|H| \cdot |\mathcal{R}| \cdot T_{\mathcal{O}})$ time by using $|\mathcal{R}|$ linear scans over $H$, one for each set in $\mathcal{R}$. We determine for every set $r \in \mathcal{R}$ whether it is hit by an element of $H$, by calling the range space oracle on $r$ and the corresponding set and elements in $H$.

The data structure $\mathcal{D}$ that maintains the probability distributions $\mathcal{D}_i$ needs to be able to answer the following three queries: The first query $\textsc{Sample}(k', \mathcal{D}_i)$ asks to sample $k'$ elements from $D_i$, the second query $\textsc{WeightUpdate}(\mathcal{D}_i, r)$ asks to double the weight of each element in $r$ in the weight function of the distribution $\mathcal{D}_i$, the third query asks to evaluate $\Pr_{\mathcal{D}_i}[r] \leq \varepsilon$ for given $r$ and $\varepsilon$.

We let $\mathcal{D}$ store the cumulative probability distribution explicitly in an array. For the initial distribution $\mathcal{D}_1$, all weights are set to 1. Therefore the initiation and reset of $\mathcal{D}$ takes $O(1)$ time and $O(|X|)$ space. A sample of $k'$ elements can be drawn from $\mathcal{D}$ in a total time of $O(k' \log(|X|))$ by a binary search on the array for a given random number. Furthermore, we can compute the weight of a set $r$ explicitly scanning over the whole array and calling the oracle $\mathcal{O}$ for each element in $X$. This takes a total time of $O(|X|T_{\mathcal{O}})$. The same scanning method can be used for doubling the weight of each element in $r$. In the first case, you keep track of the weights in the second one, you update the weight distribution. Using the above implementations of the verifier and the data structure $\mathcal{D}$, we get the following result directly from Theorem 2.4.9.

**Theorem 4.2.4.** *For a given finite range space $(X, \mathcal{R})$ with finite VC-dimension $\delta$, assume there exists a hitting set of size $k$. Then, there exists an algorithm that computes a hitting set of size $k' \in O(\delta k \log(k))$ with expected running time in*

$$O\left(k \log\left(\frac{|X|}{k}\right)(k' \log(|X|) + T_{\mathcal{O}}(k'|\mathcal{R}| + |X|))\right)$$

*after a preprocessing time in $O\Big(P_{\mathcal{O}}\Big)$ and using space in $O\Big(S_{\mathcal{O}} + |X|\Big)$.*

### 4.2.3 Bounding the VC-dimension

In order to use Theorem 4.2.4 of Section 4.2.2, we need to bound the VC-dimension of the dual range space. In our case, this will be a range space that has a similar structure as the range space of balls under the Fréchet distance studied in Chapter 3. In Theorem 3.4.16, we showed a bound of $O(dn \log(n))$ for polygonal curves in $\mathbb{R}^d$ of complexity at most $n$. Using this result directly would not gain us any useful bounds, as the subcurves $P[t_i, t_j]$ in the definition of the range space may have linear complexity in $n$—even for the simpler variant of Section 4.2.1. In fact, it turns out that the VC-dimension of the dual range space for the main problem defined in Section 2.5 does indeed inherently depend on $n$, as we show in Theorem 4.6.2 in Section 4.6.1.

In Section 4.5 we instead define an auxiliary range space that preserves solutions up to approximation and—more importantly—which has low VC-dimension in the dual. We

show this by using the techniques of Chapter 3. Our analysis of the VC-dimension is given in Section 4.5.3. Crucial to our approach is that we can answer a range query for the auxiliary range space based on the geometric predicates of Driemel, Nusser, Phillips and Psarros [57]. This is possible since the range query is a distance evaluation of a certain type of partial Fréchet distance with specific conditions that occur in our range space with proxy curves. The result is stated in Theorem 4.5.9 and implies that the VC-dimension is constant, if the complexity of the center curves $\ell$ and the ambient dimension $d$ is constant.

One may ask if a similar bound can be proven in the case where breakpoints are given with the input. Trivially, the size of the range space already gives a bound of $O(\log m)$, however this depends on the number of breakpoints $m$ and can be large even if $\ell$ is small. We study this problem in Section 4.6.2. For the range space defined in Section 2.5.1 we show a lower bound of $\Omega(\log m)$ even in the case that $d = 1$ and $\ell = 2$ (see Theorem 4.6.3). Technically, this does not rule out the existence of an auxiliary range space with low VC-dimension in the dual. However, it is not clear what such a range space would look like as Theorem 4.6.3 makes only few assumptions on the range space. Thus, perhaps surprisingly, the discretization with breakpoints which was supposed to simplify the problem, actually makes it more difficult. Therefore, our approximation guarantee in the continuous case is better than what we can currently achieve in the discrete case when breakpoints are given with the input.

## 4.3 Warm-up — Clustering with line segments

In this section, we show how to apply Theorem 4.2.4 to the discrete problem where we are given a curve $P$ with breakpoints. We assume in this section that cluster centers are restricted to be line segments (the case $\ell = 2$). The general case ($\ell \geq 2$) is discussed in Section 4.4. In contrast to the solution described in Section 4.2.1, our algorithm finds an approximate set cover without computing the range space explicitly leading to better running times.

### 4.3.1 The range space

We start by defining the range space $\widetilde{\mathcal{R}}_2$ with ground set $Z = \{1, \dots, m-1\}$. For a subsequence $i_1, \dots, i_{m'}$ of $1, \dots, m$, denote

$$\pi(i_1, \dots, i_{m'}) = \overline{P(t_{i_1})P(t_{i_2})} \oplus \overline{P(t_{i_2})P(t_{i_3})} \oplus \cdots \oplus \overline{P(t_{i_{m'-1}})P(t_{i_{m'}})}.$$

A tuple $(i, j)$ with $1 \leq i \leq j \leq m$ defines a set $r_{i,j} \in \widetilde{\mathcal{R}}_2$ as follows

$$r_{i,j} = \{z \in Z \mid \exists x \in [x_z, z], y \in [z+1, y_z] \text{ with } d_F(\pi(x, z, z+1, y), \overline{P(t_i)P(t_j)}) \leq 2\Delta\},$$

where $x_z \leq z < y_z$ are indices which we obtain as follows. We scan breakpoints starting from $z$ in the backwards order along the curve and to test for each breakpoint $x$, whether

$$d_F(\overline{P(t_x)P(t_z)}, P[t_x, t_z]) \leq 4\Delta. \tag{4.1}$$

If $x$ satisfies (4.1), then we decrement $x$ and continue the scan. If $x = 0$ or if $x$ does not satisfy (4.1), then we set $x_z = x + 1$ and stop the scan. To set $y_z$ we use a similar approach: we scan forwards from $z + 1$ along the curve and test for each breakpoint

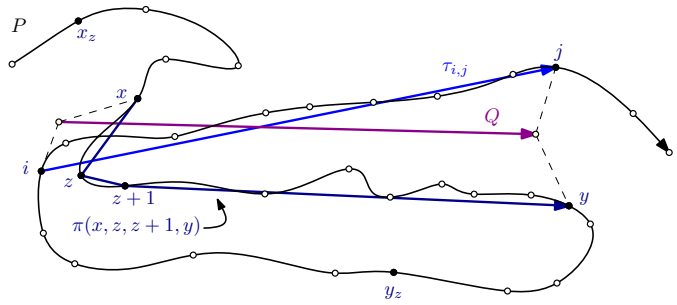


Figure 4.1: Example of a curve $P$ and index $z$, such that $z \in r_{i,j}$ for some $r_{i,j} \in \widetilde{\mathcal{R}}_2$. Also shown is a line segment $Q$, such that $z \in r_Q$ of the initial range space $\mathcal{R}$. After preprocessing, we can test $z \in r_{i,j}$ in constant time.

$y$ the same property with $\overline{P(t_{z+1})P(t_y)}$ and $P[t_{z+1}, t_y]$. If $y$ satisfies the property, we increment $y$ and continue the scan. If $y = m + 1$ or if $y$ does not satisfy the property we set $y_z = y - 1$ and stop the scan. Figure 4.1 shows an example of $z, x_z$ and $y_z$.

### 4.3.2 Analysis of the approximation error

In this section, we show how we use a set cover of the range space $\widetilde{\mathcal{R}}_2$ to construct an approximate solution for our clustering problem and analyze the resulting approximation error. In particular, we prove Lemma 4.3.2 and Lemma 4.3.3. To prove them, we first prove the following simple lemma.

**Lemma 4.3.1.** *Let $1 \leq i \leq j \leq m$ and let $I = i_1, \ldots, i_{m'}$ be a subsequence of $i, \ldots, j$. If there exists a line segment $Q \in \mathbb{X}_2^d$ such that it holds $d_F(Q, P[t_i, t_j]) \leq \alpha$, then we have $d_F(\pi(I), P[t_{i_1}, t_{i_{m'}}]) \leq 2\alpha$.*

*Proof.* For any pair $(i', j')$ of indices in $I$, there exists a line segment $Q[a, b]$ with $[a_{i'}, b_{j'}] \subseteq [0, 1]$ such that $d_F(Q[a_{i'}, b_{j'}], P[t_{i'}, t_{j'}]) \leq \alpha$. Since shortcutting cannot increase the Fréchet distance to a line segment, we also have $d_F(Q[a, b], \overline{P(t_{i'})P(t_{j'})}) \leq \alpha$. By triangle inequality, it now follows that

$$d_F(\overline{P(t_{i'})P(t_{j'})}, P[t_{i'}, t_{j'}]) \leq d_F(\overline{P(t_{i'})P(t_{j'})}, Q[a_{i'}, b_{j'}]) + d_F(Q[a_{i'}, b_{j'}], P[t_{i'}, t_{j'}]).$$

Since the inequality holds for all $(i', j') \in I$, we have $d_F(\pi(I), P[t_{i_1}, t_{i_{m'}}]) \leq 2\alpha$. □

**Lemma 4.3.2.** *Assume there exists a set cover for $\mathcal{R}$ with parameter $\Delta$. Let $S$ be a set cover of size $k$ for $\widetilde{\mathcal{R}}_2$. We can derive from $S$ a set of $k$ cluster centers $C \subseteq \mathbb{X}_2^d$ and such that $\phi(P, C) \leq 6\Delta$.*

*Proof.* We set $C = \{\overline{P(t_i)P(t_j)} \mid r_{i,j} \in S\}$. Let $r_{i,j} \in S$ and let $z \in r_{i,j}$. By the definition of $r_{i,j}$ there are $x \in [x_z, z]$ and $y \in [z + 1, y_z]$ such that

$$d_F(\pi(x, z, z + 1, y), \overline{P(t_i)P(t_j)})) \leq 2\Delta.$$

In the following we show that $d_F(\overline{P(t_i)P(t_j)}, P[t_x, t_y]) \leq 6\Delta$. With the triangle inequality we get that $d_F(\overline{P(t_i)P(t_j)}, P[t_x, t_y])$ is at most the sum of

$$d_F(\overline{P(t_i)P(t_j)}, \pi(x, z, z + 1, y))$$

and the maximum of

$$d_F(\overline{P(t_x)P(t_z)}, P[t_x, t_z]), d_F(\overline{P(t_z)P(t_{z+1})}, P[t_z, t_{z+1}]), d_F(\overline{P(t_{z+1})P(t_y)}, P[t_{z+1}, t_y])).$$

By the choice of $x$ and $y$ we have that

$$\max(d_F(\overline{P(t_x)P(t_z)}, P[t_x, t_z]), d_F(\overline{P(t_{z+1})P(t_y)}, P[t_{z+1}, t_y])) \leq 4\Delta.$$

It remains to show that $d_F(\overline{P(t_z)P(t_{z+1})}, P[t_z, t_{z+1}]) \leq 4\Delta$. Since there exists a set cover of $\mathcal{R}$ with parameter $\Delta$, there exists a line segment $Q \in \mathbb{X}_2^d$ and $1 \leq i' \leq z \leq z+1 \leq j' \leq m$ such that $d_F(Q, P[t_{i'}, t_{j'}]) \leq \Delta$. Therefore we get with Lemma 4.3.1, that

$$d_F(\overline{P(t_z)P(t_{z+1})}, P[t_z, t_{z+1}]) \leq 2\Delta.$$

Since $S$ is a set cover, it holds for the ground set $Z$, that $Z = \bigcup_{(i,j)\in S} r_{i,j}$. Therefore, if we choose $C = \{\overline{P(t_i)P(t_j)} \mid r_{i,j} \in S\}$, then $\phi(P, C) \leq 6\Delta$. □

**Lemma 4.3.3.** *If there exists a set cover $S$ of $\mathcal{R}$, then there exists a set cover of the same size for $\widetilde{\mathcal{R}}_2$.*

*Proof.* We claim that for any set $r_Q \in \mathcal{R}$ there exists a set $r \in \widetilde{\mathcal{R}}_2$, such that $r_Q \subseteq r$. This claim implies the lemma statement. It remains to prove the claim.

Let $Y$ be the set of tuples $(i, j) \in \mathbb{N}^2$ with $1 \leq i < j \leq m$ and $d_F(Q, P[t_i, t_j]) \leq \Delta$. We have that $r_Q = \bigcup_{(i,j)\in Y} [i, j) \cap \mathbb{N}$.

Let $(i, j) \in Y$. We show that $r_Q \subseteq r_{i,j} \in \widetilde{\mathcal{R}}_2$. Let $z \in r_Q$. By the definition of $r_Q$ we have

$$\exists\ x \leq z < y \text{ s.t. } d_F(Q, P[t_x, t_y]) \leq \Delta.$$

To show that $z \in r_{i,j}$, we prove that the following two conditions hold:

(i) $x \in [x_z, z]$ and $y \in [z+1, y_z]$,

(ii) $d_F(\pi(x, z, z+1, y), \overline{P(t_i)P(t_j)}) \leq 2\Delta$.

Since $d_F(Q, P[t_x, t_y]) \leq \Delta$ and shortcutting cannot increase the Fréchet-distance to a line segment, we also have

$$d_F(Q, \pi(x, z, z+1, y)) \leq \Delta.$$

Similarly, we can conclude $d_F(Q, \overline{P(t_i)P(t_j)}) \leq \Delta$. It now follows from the triangle inequality, that

$$d_F(\pi(x, z, z+1, y), \overline{P(t_i)P(t_j)}) \leq d_F(\pi(x, z, z+1, y), Q) + d_F(Q, \overline{P(t_i)P(t_j)}) \leq 2\Delta.$$

This implies condition (ii).

The first condition (i) follows in a similar way. Since $r_Q \in S$, there exists $[a, b] \subseteq [0, 1]$, such that $d_F(Q[a, b], P[t_x, t_z]) \leq \Delta$. Therefore, by Lemma 4.3.1, for all $x' \in [x, z]$ $d_F(\overline{P(t_{x'})P(t_z)}, P[t_{x'}, t_z]) \leq 2\Delta$. As such, $x$ is encountered in the scan and ends up being contained in the interval $[x_z, z]$.

We can make a symmetric argument to show that $d_F(\overline{P(t_{z+1})P(t_y)}, P[t_{z+1}, t_y]) \leq 2\Delta$ and conclude using Lemma 4.3.1 that $y \in [z+1, y_z]$. This proves condition (i).

Together, the above implies that $z \in r_{i,j}$ for $r_{i,j} \in \widetilde{\mathcal{R}}_2$. Therefore $r_Q \subseteq r_{i,j}$ for some $r_{i,j} \in \widetilde{\mathcal{R}}_2$. □

### 4.3.3 The algorithm

We intend to use the algorithm of Theorem 4.2.4 to find a set cover of the range space $\widetilde{\mathcal{R}}_2$, since such a set cover gives a 6-approximation for our clustering problem. The algorithm requires a range space oracle for $\widetilde{\mathcal{R}}_2$. In this section, we describe such a range space oracle. In particular, we show how to build a data structure that answers a query, given indices $i, j$ and $z$, for the predicate $z \in r_{i,j}$ in $O(1)$ time.

**The data structure.** To build the data structure for the oracle, we first compute the indices $x_z$ and $y_z$ for each $1 \leq z \leq m-1$, as specified in the definition of the range space in Section 4.3.1. Next, we construct a data structure that can answer for a pair of breakpoints $i$ and $z$ if there is a breakpoint $x$ with $x_z \leq x \leq z$ such that $\|P(t_i) - P(t_x)\| \leq 2\Delta$ in $O(1)$ time. For this, we build an $m \times m$ matrix $M$ in the following way. For each breakpoint $i$ we go through the sorted list of breakpoints and check if $\|P(t_i) - P(t_j)\| \leq 2\Delta$ for each $1 \leq j \leq m$. While doing that, we determine for each $j$ which is the first breakpoint $z_{i,j} \geq j$ with $\|P(t_i) - P(t_{z_{i,j}})\| \leq 2\Delta$. The entries $z_{i,j}$ are then stored in the matrix $M$ at position $M(i,j)$. Given the Matrix $M$ the oracle can answer if there is a breakpoint $x$ with $x_z \leq x \leq z$ such that $\|P(t_i) - P(t_x)\| \leq 2\Delta$ by checking if $M(i, x_z) \leq z$. The data structure can also answer if there is a breakpoint $y$ with $z+1 \leq y \leq y_z$ such that $\|P(t_j) - P(t_y)\| \leq 2\Delta$ by checking if $M(j, z+1) \leq y_z$. The final data structure stores the matrix $M$ only.

**The query.** We answer queries as follows. Given $z, i$ and $j$, we want to determine if $z \in r_{i,j}$. We return "yes", if the following three conditions are satisfied:

(i) $M(i, x_z) \leq z$

(ii) $M(j, z+1) \leq y_z$

(iii) At least one of the following holds:

  (1) $\|s - P(t_z)\| \leq 2\Delta$, where $s$ is the intersection of the bisector between the points $P(t_z)$ and $P(t_{z+1})$ and the line segment $\overline{P(t_i)P(t_j)}$

  (2) $\|o_1 - P(t_z)\| \leq 2\Delta$ and $\|o_2 - P(t_{z+1})\| \leq 2\Delta$, where $o_1$ and $o_2$ are the orthogonal projections of the points $P(t_z)$ and $P(t_{z+1})$ on the line segment $\overline{P(t_i)P(t_j)}$

Otherwise, the algorithm returns "no".

**Correctness.** The above-described range space oracle returns the correct answer. Correctness is implied by the following observation, which follows from the analysis of Alt and Godau [13]. See also Figure 4.2.

**Observation 4.3.4.** *$d_F(\pi(x, z, z+1, y), \overline{P(t_i)P(t_j)}) \leq 2\Delta$ if and only if the following three conditions are satisfied:*

*(i)* $\|P(t_x) - u\| \leq 2\Delta$

*(ii)* $\|P(t_y) - v\| \leq 2\Delta$

*(iii)* $\min_{\substack{\lambda, \lambda' \in [0,1] \\ \lambda \leq \lambda'}} (\|a - (\lambda v + (1-\lambda)u)\|, \|b - (\lambda' v + (1-\lambda')u)\|) \leq 2\Delta$

*where $a = P(t_z)$, $b = P(t_{z+1})$, $u = P(t_i)$, and $v = P(t_j)$.*

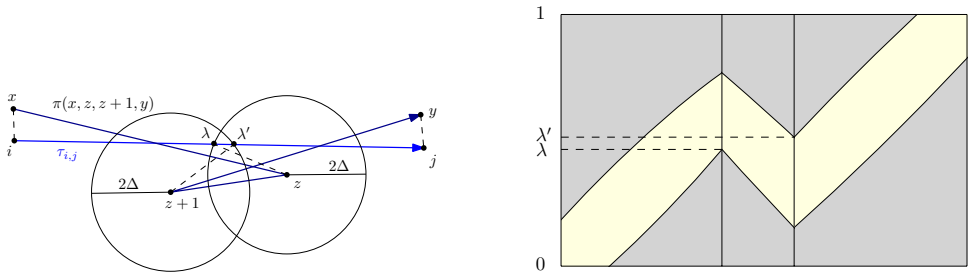


Figure 4.2: Illustration of Observation 4.3.4. The figure on the right shows the 2Δ-free space diagram of the two curves on the left. A monotone path from the bottom left to the upper right corner of the diagram is feasible iff the three conditions stated in the observation are satisfied. We slightly abuse notation by referring to the vertex $P(t_z)$ with $z$ in all figures, when context is clear.

**Running time.** Next, we analyze the running time of constructing an oracle for the case $\ell = 2$ and query time $O(1)$. In particular, we analyze the running time of the scan for the indices $x_z$ (or $y_z$) with $1 \leq z < m$ and the running time for building the matrix $M$.

As described above the index-scan for $x_z$, given $z$, can be done by checking for breakpoints $x \in \{z-1, \ldots, 1\}$ in backwards order from $z$ if $d_F(\overline{P(t_x)P(t_z)}, P[t_x, t_z]) \leq 4\Delta$. Since $\overline{P(t_x)P(t_z)}$ has complexity 2 and $P[t_x, t_z]$ has complexity at most $n$, the check $d_F(\overline{P(t_x)P(t_z)}, P[t_x, t_z]) \leq 4\Delta$ can be done in $O(n)$ time and $O(n)$ space for any $x, z \in \{1, \ldots, m\}$ using standard methods [13]. The scan for $y_z$ is analogous, so we need a total time of $O(mn)$ to scan for all indices.

For building the matrix $M$, the algorithm computes the Euclidean distances of all $\binom{m}{2}$ pairs of breakpoints and while doing that records for each breakpoint $t_j$ the smallest index of a breakpoint after $t_j$ that lies within distance $2\Delta$ to this breakpoint. In total, this it takes $O(m^2)$ time. Together with the scan for the indices, we get the following runtime for building the oracle.

**Theorem 4.3.5.** *One can build a data structure of size $O(m^2)$ in time $O(m(m+n))$ and space $O(n+m^2)$ that answers for an element of the ground set $Z$ and a set of $\widetilde{\mathcal{R}}_2$, whether this element is contained in the set in $O(1)$ time.*

### 4.3.4 The result

For the range space $(Z, \widetilde{\mathcal{R}}_2)$, we have $|Z| = m$ and $|\widetilde{\mathcal{R}}_2| = O(m^2)$. Thus, the VC-dimension $\delta$ of the dual range space is trivially bounded by $O(\log m)$. We combine this with the result for constructing the oracle in Theorem 4.3.5 and apply Theorem 4.2.4 to get the following lemma on computing set covers of $\widetilde{\mathcal{R}}_2$. Note that we must have $k < m$, since there are only $m-1$ elements in the ground set.

**Lemma 4.3.6.** *Let $k$ be the minimum size of a set cover for $\widetilde{\mathcal{R}}_2$. There exists an algorithm that computes a set cover for $\widetilde{\mathcal{R}}_2$ of size $O(k \log^2(m))$ with an expected running time in $\widetilde{O}\left(km^2 + mn\right)$ and using space in $O(n+m^2)$.*

As a direct consequence, we get the following result for our clustering problem in the case $\ell = 2$ with the help of Lemma 4.3.2 and Lemma 4.3.3.

**Theorem 4.1.1.** *Let $P : [0,1] \to \mathbb{R}^d$ be a polygonal curve of complexity $n$ with breakpoints $0 \leq t_1, \ldots, t_m \leq 1$ and let $\Delta > 0$ be a parameter. Assume there exists a set $C^* \subset \mathbb{X}_2^d$ of size $k \leq m$, such that $\phi(P, C^*) \leq \Delta$. There exists an algorithm that computes a set $C \subset \mathbb{X}_2^d$ of size $O(k \log^2(m))$ such that $\phi(P, C) \leq 6\Delta$. The algorithm has expected running time in $\widetilde{O}\left(km^2 + mn\right)$ and uses space in $O(n + m^2)$.*

## 4.4 The main algorithm

In this section, we extend the scheme described in Section 4.3 to the case $\ell > 2$. As in the previous section, we only consider the discrete problem, where the input is a polygonal curve with breakpoints. Again, the crucial step is a careful definition of a range space for approximation which allows for an efficient implementation of a range space oracle. The main idea is to replace the edges of the proxy curve $\pi$ from Section 4.3 by simplifications of the corresponding subcurves. We show that we can do this in a way that ensures that these simplifications are nested in a certain way. This, in turn, will allow us to build efficient oracle data structures for this range space. We will later show how to use the main elements of this algorithm for the continuous case in Section 4.5.

### 4.4.1 Simplifications

We begin by introducing the following slightly different notion of simplification. A curve $Q \in \mathbb{X}_\ell^d$ is an $(\varepsilon, \ell)$-***simplification*** of a curve $P$ if $Q$ has at most $\ell$ vertices and its Fréchet distance to $P$ is at most $\varepsilon$. We call the simplification ***vertex-restricted*** if $V(Q) \subseteq V(P)$ and the vertices of $Q$ have the same order as in $P$. In this context, we say that a point $p$ of $P$ ***corresponds*** to an edge $e$ of a vertex-restricted simplification of $P$ if it lies in between the two endpoints of $e$ in $P$. The main purpose of this section is to define simplifications $\sigma^+(i,j)$, $\sigma^-(i,j)$ and $\sigma^\circ(i,i+1)$ for $i,j \in \{1,\ldots,m\}$ that we will use in the definition of the range space in the next section. Concretely, the simplifications will be defined as the output of the algorithm from [4]. In a nutshell, their algorithm works the following way: Let $P$ be a curve with vertices $P(s_1), \ldots, P(s_n)$. Let $f(\frac{\varepsilon}{2})$ denote the minimum number of vertices in a vertex-restricted $(\frac{\varepsilon}{2}, n)$-simplification of $P$. To compute a vertex-restricted $(\varepsilon, f(\frac{\varepsilon}{2}))$-simplification $P'$ of the curve $P$, the algorithm iteratively adds new vertices to the simplification starting with the first vertex $P(s_1)$ of the curve. In each step it takes the last vertex $P(s_i)$ of the simplification and determines with an exponential search the last integer $j \geq 0$ such that $d_F(\overline{P(s_i)P(s_{i+2^j})}, P[s_i, s_{i+2^j}]) \leq \varepsilon$. After determining $j$ it finds with a binary search the last integer $r \in [2^j, 2^{j+1}]$ such that $d_F(\overline{P(s_i)P(s_{i+r})}, P[s_i, s_{i+r}]) \leq \varepsilon$ and adds $P(s_{i+r})$ to the simplification. The algorithm terminates when it reaches $P(s_n)$.

**Generating simplifications.** We now describe how to generate a set of simplifications that will be used in the definition of our range space in Section 4.4.2. We apply the above described algorithm on subcurves of $P$ in the following way: Consider the parameterization $P\colon [0,1] \to \mathbb{R}^d$ of $P$ where $P(t_i)$ gives the $i$-th breakpoint of $P$ for $1 \leq i \leq m$ and $P(s_j)$ gives the $j$-th vertex of $P$ for $1 \leq j \leq n$. For each $z \in \{1,\ldots,m\}$ we apply the algorithm with $\varepsilon = 4\Delta$ on $P[t_z, 1]$ to get a simplification $P_z^+$. We stop the algorithm early if the complexity of the simplification reaches $2\ell$. If $|P_z^+| = 2\ell$ let $P(s_{z_{2\ell}})$ be the $2\ell$-th vertex of $P_z^+$. Otherwise set $P(s_{z_{2\ell}}) = P(s_n)$. Let $y_z$ be the last breakpoint of $P$ before

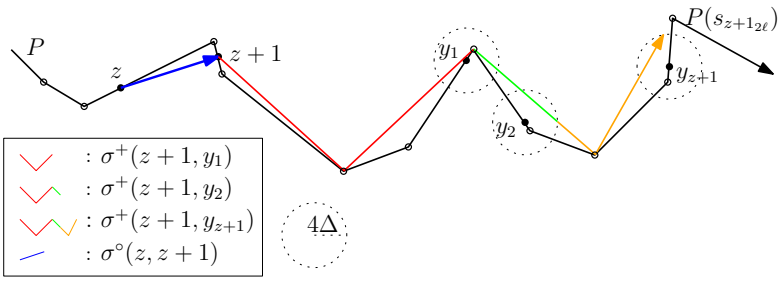


Figure 4.3: Example of the generated $(4\Delta, 2\ell)$-simplifications for a curve $P$ with breakpoints $z, z+1, y_1, y_2$ and $y_{z+1}$ in the case $\ell = 2$.

$P(s_{z_{2\ell}})$. Let $z \leq y \leq y_z$. Since $P_z^+$ is a $(4\Delta, 2\ell)$-simplification of $P[t_z, 1]$, there exists a subcurve $\sigma^+(z, y)$ of $P_z^+$ such that $d_F(\sigma^+(z, y), P[t_z, t_y]) \leq 4\Delta$. From each possible subcurve with the above property let $\sigma^+(z, y)$ be the longest subcurve that does not contain any vertex $P(s_i)$ with $s_i \geq t_y$, except for the last vertex of this subcurve. This subcurve $\sigma^+(z, y)$ is therefore a uniquely defined $(4\Delta, 2\ell)$-simplification of $P[t_z, t_y]$ that ends in a point of the edge of $P_z^+$ corresponding to $P(t_y)$. Analogously we generate the curve $\sigma^\circ(z, z+1)$ by running the algorithm for the curve $P[t_z, t_{z+1}]$ and the $\sigma^-(x, z)$ by running the algorithm for the direction-inverted curve $P[t_z, 0]$. We define $P[t_z, 0]$ to be the curve $Q : [0, 1] \to \mathbb{R}^d$ with $Q(t) = P((1-t)t_z)$. Note that it is possible that the algorithm does not find a simplification at all for a specific subcurve. In this case we say the simplification is empty (and we denote this with $\perp$). See also Figure 4.3 for an example of the generated simplifications.

We summarize crucial properties of the generated simplifications in the following two lemmata. These properties will help to construct an efficient oracle for our range space later.

**Lemma 4.4.1.** *Let $i, j \in \{1, \ldots, m\}$ with $i < j$. The curve $\sigma^+(i, j)$ is either a uniquely defined $(4\Delta, 2\ell)$-simplification of $P[t_i, t_i]$, or it is $\sigma^+(i, j) = \perp$. In the latter case, there exists no $Q \in \mathbb{X}_\ell^d$ such that $d_F(Q, P[t_i, t_j]) \leq \Delta$. Moreover, for any non-empty simplification $\sigma^+(i, j)$ and for any $i < j' < j$, the simplification $\sigma^+(i, j')$ is non-empty and is a subcurve of $\sigma^+(i, j)$.*

We get symmetric lemmas for the other simplifications. We will see in the next section why it is convenient to have these properties in both directions, forwards and backwards along the curve.

**Lemma 4.4.2.** *Let $i, j \in \{1, \ldots, m\}$ with $i < j$. The curve $\sigma^-(i, j)$ is either a uniquely defined $(4\Delta, 2\ell)$-simplification of $P[t_i, t_j]$, or it is $\sigma^-(i, j) = \perp$. In the latter case there exists no $Q \in \mathbb{X}_\ell^d$ such that $d_F(Q, P[t_i, t_j]) \leq \Delta$. Moreover, for any non-empty simplification $\sigma^-(i, j)$ and for any $i < i' < j$ it holds that the simplification $\sigma^-(i', j)$ is non-empty and is a subcurve of $\sigma^-(i, j)$.*

**Lemma 4.4.3.** *Let $z \in \{1, \ldots, m-1\}$. The curve $\sigma^\circ(z, z+1)$ is either a uniquely defined $(4\Delta, 2\ell)$-simplification of $P[t_z, t_{z+1}]$, or it is $\sigma^\circ(z, z+1) = \perp$. In the latter case there exists no $Q \in \mathbb{X}_\ell^d$ such that $d_F(Q, P[t_z, t_{z+1}]) \leq \Delta$.*

Lemma 4.4.1 follows directly from the following lemma. Lemma 4.4.2 and Lemma 4.4.3 follow by using symmetric arguments.

**Lemma 4.4.4.** *Consider the generating process described in Section 4.4.1. Let $y$ be a breakpoint of $P$ with $t_y > s_{z_{2\ell}}$. There exists no $Q \in \mathbb{X}_\ell^d$ such that $d_F(Q, P[t_z, t_y]) \leq \Delta$.*

*Proof.* Let $1 \leq v \leq n$ such that $s_{v-1} \leq t_y \leq s_v$. So $P(s_v)$ is the first vertex of $P$ after the breakpoint $y$. Assume there exists a $Q \in \mathbb{X}_l^d$ such that $d_F(Q, P[t_z, t_y]) \leq \Delta$.

To get a contradiction we will show that, with this assumption, we can construct a vertex-restricted $(2\Delta, 2\ell - 1)$-simplification of $P[t_z, s_v]$. Let $f(2\Delta)$ denote the minimum number of vertices in a vertex-restricted $(2\Delta, n)$-simplification of $P[t_z, s_v]$. Note that $v > z_{2\ell}$. So the vertex-restricted $(4\Delta, f(2\Delta))$-simplification $P'$ of the subcurve $P[t_z, s_v]$ computed with the algorithm of [4] has a complexity of at least $2\ell + 1$. This follows by the definition of $P(s_{z_{2\ell}})$. Therefore we have $f(2\Delta) \geq 2\ell + 1$. But our constructed vertex-restricted $(2\Delta, 2\ell - 1)$-simplification then would directly contradictict $f(2\Delta) \geq 2\ell + 1$.

For the construction of the $(2\Delta, 2\ell - 1)$-simplification let $\tilde{P} = P[t_z, s_{v-1}]$. Since $Q$ is a $(\Delta, \ell)$-simplification of $P[t_z, t_y]$, there exists a subcurve $\tilde{Q}$ of $Q$ with $d_F(\tilde{Q}, \tilde{P}) \leq \Delta$. Let $e_1, \ldots, e_k$ be the edges of $\tilde{Q}$ and $\tilde{p}_1, \ldots, \tilde{p}_j$ be the vertices of $\tilde{P}$. It is $k \leq l - 1$ and $j \leq n$. Let $\gamma$ be a strictly monotone-increasing function such that

$$d_F(\tilde{P}, \tilde{Q}) = \sup_{t \in [0,1]} \|\tilde{P}(t) - \tilde{Q}(\gamma(t))\| \leq \Delta.$$

Let further

$$t_{i_1} = \min\{t \in [0,1] \mid \tilde{Q}(\gamma(t)) \in e_i, \tilde{P}(t) \in \{\tilde{p}_1, \ldots, \tilde{p}_j\}\}$$

be the first vertex of $\tilde{P}$ that gets mapped to $e_i$ and

$$t_{i_2} = \max\{t \in [0,1] \mid \tilde{Q}(\gamma(t)) \in e_i, \tilde{P}(t) \in \{\tilde{p}_1, \ldots, \tilde{p}_j\}\}$$

be the last vertex of $\tilde{P}$ that gets mapped to $e_i$. By construction we have

$$d_F(\overline{\tilde{P}(t_{i_1})\tilde{P}(t_{i_2})}, \overline{\tilde{Q}(\gamma(t_{i_1}))\tilde{Q}(\gamma(t_{i_2}))}) \leq \Delta$$

and therefore with the use of triangle inequality

$$\begin{aligned} & d_F(\overline{\tilde{P}(t_{i_1})\tilde{P}(t_{i_2})}, \tilde{P}[t_{i_1}, t_{i_2}]) \\ \leq\ & d_F(\overline{\tilde{P}(t_{i_1})\tilde{P}(t_{i_2})}, \overline{\tilde{Q}(\gamma(t_{i_1}))\tilde{Q}(\gamma(t_{i_2}))}) + d_F(\overline{\tilde{Q}(\gamma(t_{i_1}))\tilde{Q}(\gamma(t_{i_2}))}, \tilde{P}[t_{i_1}, t_{i_2}]) \\ \leq\ & \Delta + \Delta \\ =\ & 2\Delta \end{aligned}$$

Since $\tilde{P}(t_{i_2})$ and $\tilde{P}(t_{(i+1)_1})$ are consecutive vertices of $\tilde{P}$, we also have

$$d_F(\overline{\tilde{P}(t_{i_2})\tilde{P}(t_{(i+1)_1})}, \tilde{P}[t_{i_2}, t_{(i+1)_1}]) = 0.$$

So we can construct a $(2\Delta, 2\ell - 1)$-simplification of $P[t_z, s_v]$ by concatenating the vertices

$$\tilde{P}(t_{1_1}), \tilde{P}(t_{1_2}), \tilde{P}(t_{2_1}), \tilde{P}(t_{2_2}), \ldots, \tilde{P}(t_{k_1}), \tilde{P}(t_{k_2}), P(s_v).$$

To see that the resulting curve is indeed a vertex-restricted simplification, we observe that $\tilde{P}(t_{1_1}) = \tilde{P}(0) = P(t_z)$ and that the edge from $\tilde{P}(t_{k_2}) = P(s_{v-1})$ to $P(s_v)$ is entirely included in $P$. □

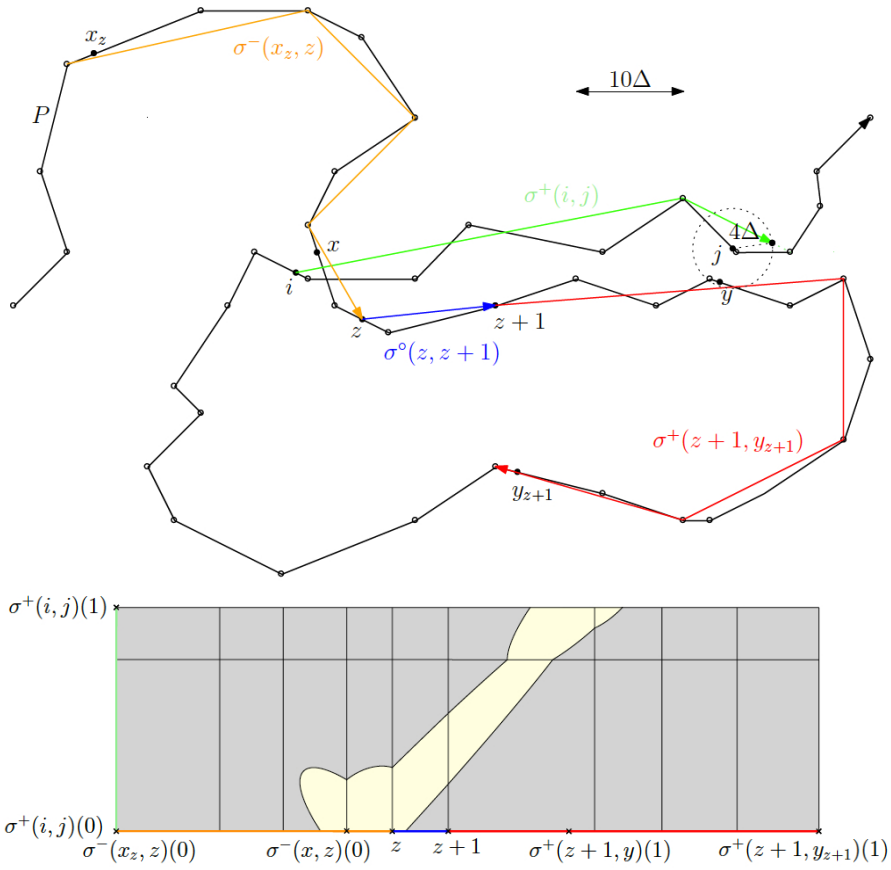


Figure 4.4: Example of a curve $P$ such that $z \in r_{i,j}$ for some $r_{i,j} \in \widetilde{\mathcal{R}}_3$. Also shown is the $10\Delta$-free space diagram of $\kappa_z(x, y)$ and $\sigma^+(i, j)$. Simplification $\sigma^+(i, j)$ demonstrates that the simplifications do not have to be vertex-restricted.

### 4.4.2 The range space

We are now ready to define the new range space $\widetilde{\mathcal{R}}_3$ with ground set $Z = \{1, \ldots, m-1\}$. The range space depends on the simplifications of subcurves of $P$ defined in the previous section. Let $(i, j)$ be a tuple with $1 \leq i \leq j \leq m$. We say $r_{i,j} = \emptyset$ if there is no $Q \in \mathbb{X}_\ell^d$ such that $d_F(Q, P[t_i, t_j]) \leq \Delta$. Otherwise, we define a set $r_{i,j} \in \widetilde{\mathcal{R}}_3$ as follows

$$r_{i,j} = \{z \in Z \mid \exists x \in [x_z, z], y \in [z+1, y_{z+1}] \text{ with } d_F(\kappa_z(x, y), \sigma^+(i, j)) \leq 10\Delta\},$$

where

$$\kappa_z(x, y) = \sigma^-(x, z) \oplus \sigma^\circ(z, z+1) \oplus \sigma^+(z+1, y)$$

and $x_z \leq z$ is the smallest index such that $\sigma^-(x, z) \neq \perp$ for all $x_z \leq x \leq z$ and $y_{z+1} \geq z+1$ is the highest index such that $\sigma^+(z+1, y) \neq \perp$ for all $z+1 \leq y \leq y_{z+1}$. For an example of a curve $P$ with breakpoints $z, i, j$ such that $z \in r_{i,j}$ see Figure 4.4. Note that, by Lemma 4.4.3 the curve $\sigma^\circ(z, z+1)$ is non-empty for all $z \in \{1, \ldots, m-1\}$ if there exists a set of cluster centers $C \subset \mathbb{X}_\ell^d$ such that $\Phi(P, C) \leq \Delta$. So in this case the range space is well-defined as implied by the Lemmas 4.4.1, 4.4.2 and 4.4.3.

### 4.4.3 Analysis of the approximation error

We show correctness in the same schema as in Section 4.3.2. In particular, we prove Lemma 4.4.5 and Lemma 4.4.6.

**Lemma 4.4.5.** *Let $S$ be a set cover of size $k$ for $\widetilde{\mathcal{R}}_3$. We can derive from $S$ a set of $3k$ cluster centers $C \subseteq \mathbb{X}_\ell^d$ and such that $\phi(P, C) \leq 14\Delta$.*

*Proof.* To construct $C$ from $S$ we take for each tuple $r_{i,j} \in S$ the center curve $\sigma^+(i,j)$. Let $z \in r_{i,j}$. By the definition of $r_{i,j}$ there are $x \in [x_z, z]$ and $y \in [z+1, y_z]$ such that $d_F(\kappa_z(x,y), \sigma^+(i,j)) \leq 10\Delta$. In the following we show that $d_F(\sigma^+(i,j), P[t_x, t_y]) \leq 14\Delta$. With the triangle inequality, we get

$$\begin{aligned} d_F(\sigma^+(i,j), P[t_x,t_y]) &\leq d_F(\sigma^+(i,j), \kappa_z(x,y)) + d_F(\kappa_z(x,y), P[t_x,t_y]) \\ &\leq 10\Delta + d_F(\kappa_z(x,y), P[t_x,t_y]) \end{aligned}$$

Here, the distance $d_F(\kappa_z(x,y), P[t_x,t_y])$ is at most the maximum of the distances $d_F(\sigma^-(x,z), P[t_x,t_z])$, $d_F(\sigma^\circ(z,z+1), P[t_z,t_{z+1}])$ and $d_F(\sigma^+(z+1,y), P[t_{z+1},t_y]))$. Since $\sigma^-(x,z)$, $\sigma^\circ(z,z+1)$ and $\sigma^+(z+1,y)$ are $(4\Delta, 2\ell)$-simplifications of the corresponding subcurves, we get

$$d_F(\kappa_z(x,y), P[t_x,t_y]) \leq 4\Delta.$$

and in total $d_F(\sigma^+(i,j), P[t_x,t_y]) \leq 14\Delta$.

Since $S$ is a set cover, it holds for the ground set $Z = \{1, \ldots, m-1\}$, that $Z = \bigcup_{(i,j)\in S} r_{i,j}$. Therefore, if we choose $C' = \{\sigma^+(i,j) \mid r_{i,j} \in S\}$, we get $\phi(P, C') \leq 14\Delta$. Note that $C' \subseteq \mathbb{X}_{2\ell}^d$. Let $c \in C'$ with vertices $c_1, \ldots, c_N$ where $N \leq 2\ell$. We can arbitrarily split $c$ into 3 curves $c^{(1)}, c^{(2)}, c^{(3)}$ of complexity at most $\ell$. Those 3 subcurves can cover the same parts of $P$, that $c$ can cover since each subcurve $P'$ of $P$ with $d_F(P', c) \leq 14\Delta$ can be split into 3 subcurves $P^{(1)}, P^{(2)}, P^{(3)}$ such that $d_F(P^{(1)}, c^{(1)})$, $d_F(P^{(2)}, c^{(2)})$ and $d_F(P^{(3)}, c^{(3)})$ are each at most $14\Delta$. So, if we split each curve $c \in C'$ as described above, we obtain a set $C \subseteq \mathbb{X}_\ell^d$ with $|C| = 3|C'|$ and $\phi(P, C) \leq 14\Delta$. □

**Lemma 4.4.6.** *If there exists a set cover $S$ of $\mathcal{R}$, then there exists a set cover of the same size for $\widetilde{\mathcal{R}}_3$.*

*Proof.* We claim that for any set $r_Q \in \mathcal{R}$ there exists a set $r \in \widetilde{\mathcal{R}}_3$, such that $r_Q \subseteq r$. This claim implies the lemma statement. It remains to prove the claim.

Let $Y$ be the set of tuples $(i,j) \in \mathbb{N}^2$ with $1 \leq i < j \leq m$ and $d_F(Q, P[t_i,t_j]) \leq \Delta$. We have that $r_Q = \bigcup_{(i,j)\in Y} [i,j) \cap \mathbb{N}$.

Let $(i,j) \in Y$. We show that $r_Q \subseteq r_{i,j} \in \widetilde{\mathcal{R}}_3$. Let $z \in r_Q$. By the definition of $r_Q$ we have

$$\exists\ x \leq z < y \text{ s.t. } d_F(Q, P[t_x,t_y]) \leq \Delta$$

To show that $z \in r_{i,j}$, we prove that the following two conditions hold:

(i) $x \in [x_z, z]$ and $y \in [z+1, y_z]$,

(ii) $d_F(\kappa_z(x,y), \sigma^+(i,j)) \leq 10\Delta$.

As stated above, we have $d_F(Q, P[t_x, t_y]) \leq \Delta$. Therefore we can subdivide $Q$ into 3 subcurves $Q_x, Q_z, Q_y$ such that

$$\max(d_F(Q_x, P[t_x, t_z]), d_F(Q_z, P[t_z, t_{z+1}]), d_F(Q_y, P[t_{z+1}, t_y])) \leq \Delta$$

Each of the subcurves has complexity at most $\ell$ since $Q$ has complexity at most $\ell$. By the Lemmas 4.4.2 and 4.4.1, we have $\sigma^-(x', z) \neq \perp$ for all $x \leq x' \leq z$ and $\sigma^+(z+1, y') \neq \perp$ for all $z+1 \leq y' \leq y_{z+1}$. We can conclude that $x \in [x_z, z]$ and $y \in [z+1, y_z]$ and therefore condition (i) is fulfilled.

To prove condition (ii) we can use the triangle inequality to get

$$d_F(\kappa_z(x, y), \sigma^+(i, j)) \leq d_F(\kappa_z(x, y), Q) + d_F(Q, \sigma^+(i, j))$$

Since we have

$$\begin{aligned} d_F(\kappa_z(x, y), Q) &\leq d_F(\kappa_z(x, y), P[t_x, t_y]) + d_F(P[t_x, t_y], Q) \\ &\leq 4\Delta + \Delta \\ &= 5\Delta \end{aligned}$$

and

$$\begin{aligned} d_F(Q, \sigma^+(i, j)) &\leq d_F(Q, P[t_i, t_j]) + d_F(P[t_i, t_j], \sigma^+(i, j)) \\ &\leq \Delta + 4\Delta \\ &= 5\Delta \end{aligned}$$

we get in total

$$d_F(\kappa_z(x, y), \sigma^+(i, j)) \leq 10\Delta$$

Together, the above implies that $z \in r_{i,j}$ and therefore $r_Q \subseteq r_{i,j}$. □

### 4.4.4 The approximation oracle

To find a set cover of the range space $\widetilde{\mathcal{R}}_3$ we want to use the adapted version of the multiplicative weight update method described in Section 4.2.2. But to apply Theorem 4.2.4 directly we would need to implement an oracle that answers for an element of the ground set $Z = \{1, \ldots, m-1\}$ and a set of $\widetilde{\mathcal{R}}_3$, whether this element is contained in the set. In this section, we describe how to answer such queries approximately. In the next section (Section 4.4.5) we then show how to apply Theorem 4.2.4.

The approximation oracle will have the following properties. Given a set $r_{i,j} \in \widetilde{\mathcal{R}}_3$ and an element $z \in Z$ this approximation oracle returns either one of the following answers:

(i) "Yes", in this case, there exists a breakpoint $x \in [x_z, z]$ and a breakpoint $y \in [z+1, y_{z+1}]$ with $d_F(\kappa_z(x, y), \sigma^+(i, j)) \leq 46\Delta$

(ii) "No", in this case $z \notin r_{i,j}$.

In both cases the answer is correct.

To construct the approximation oracle we build a data structure that answers a query, given indices $i$,$j$ and $z$, for the predicate $z \in r_{i,j}$ in $O(\ell^2)$ time. In particular we need a data structure that can build a free space diagram of the curves $\kappa_z(x_z, y_{z+1})$ and $\sigma^+(i, j)$

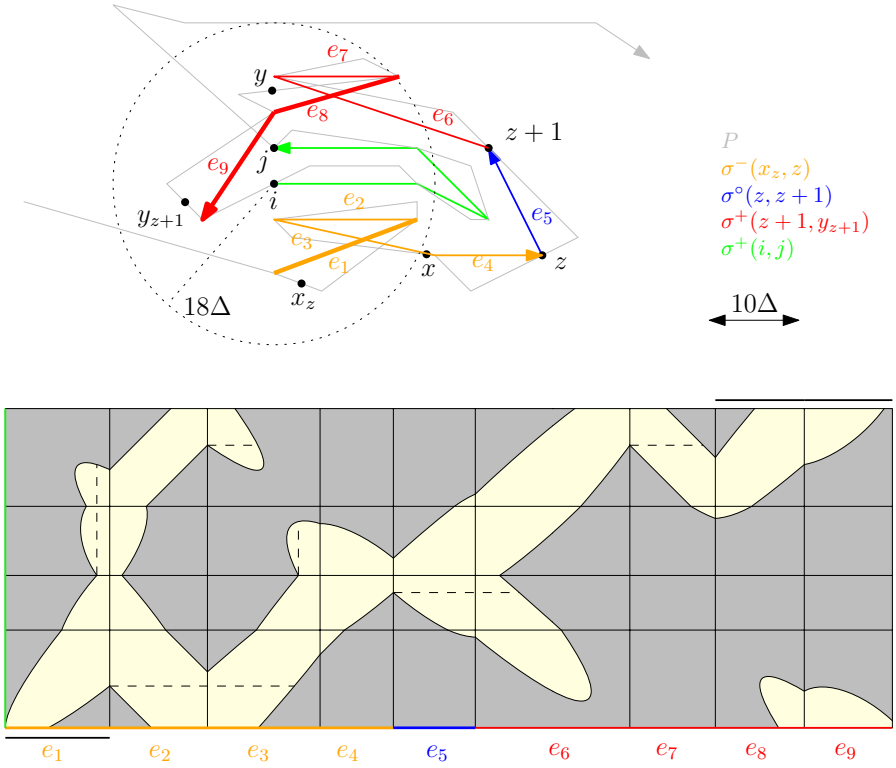


Figure 4.5: Example of a curve $P$ and breakpoints $x_z$, $x$, $z$, $z+1$, $y$, $y_{z+1}$, $i$ and $j$. The active edges are $e_1$, $e_8$ and $e_9$ since there are breakpoints corresponding to these edges within distance $18\Delta$ to $i$ or $j$ respectively. There is, however, no strictly monotone path from $e_1$ on the bottom to $e_8$ or $e_9$ on the top in the $10\Delta$-free space of $\sigma^+(i,j)$ and $\kappa_z(x_z, y_{z+1})$. So we have $z \notin r_{i,j}$.

to bound the distance $d_F(\kappa_z(x,y), \sigma^+(i,j))$ for every $x \in [x_z, z]$ and $y \in [z+1, y_{z+1}]$. In this context, we define active edges of the simplifications $\sigma^-(x_z, z)$ and $\sigma^+(z+1, y_{z+1})$ with respect to $r_{i,j}$ since the data structure needs to be able to find these efficiently to answer the query. Recall that a point of $P$ is said to correspond to an edge $e$ of a vertex-restricted simplification of $P$ if it lies in between the two endpoints of $e$ in $P$.

**Definition 4.4.7.** *Let $z, i, j$ be breakpoints of $P$. An edge $e$ of the simplification $\sigma^-(x_z, z)$ is **active** with respect to $r_{i,j}$ if there is a breakpoint $x \in [x_z, z]$ corresponding to $e$ with $d(P(t_x), P(t_i)) \leq 18\Delta$. An edge $e$ of the simplification $\sigma^+(z+1, y_{z+1})$ is **active** with respect to $r_{i,j}$ if there is a breakpoint $y \in [z+1, y_{z+1}]$ corresponding to $e$ with $d(P(t_y), P(t_j)) \leq 18\Delta$.*

So an active edge is an edge of the simplification that contains the image of a breakpoint that is close to $i$ or $j$ respectively. The active edges will become relevant for answering a query since in the case that $z \in r_{i,j}$ there exist breakpoints $x$ and $y$ on active edges such that $d_F(\kappa_z(x,y), \sigma^+(i,j)) \leq 10\Delta$. For an approximate solution, it will suffice to check the existence of a strictly monotone path in the free space diagram that starts on an active edge of $\sigma^-(x_z, z)$ and ends in an active edge of $\sigma^+(z+1, y_{z+1})$. The advantage is that this can be done faster than checking if $d_F(\kappa_z(x,y), \sigma^+(i,j)) \leq 10\Delta$ for each $x \in [x_z, z]$ and $y \in [z+1, y_{z+1}]$. See Figure 4.5 for an example.

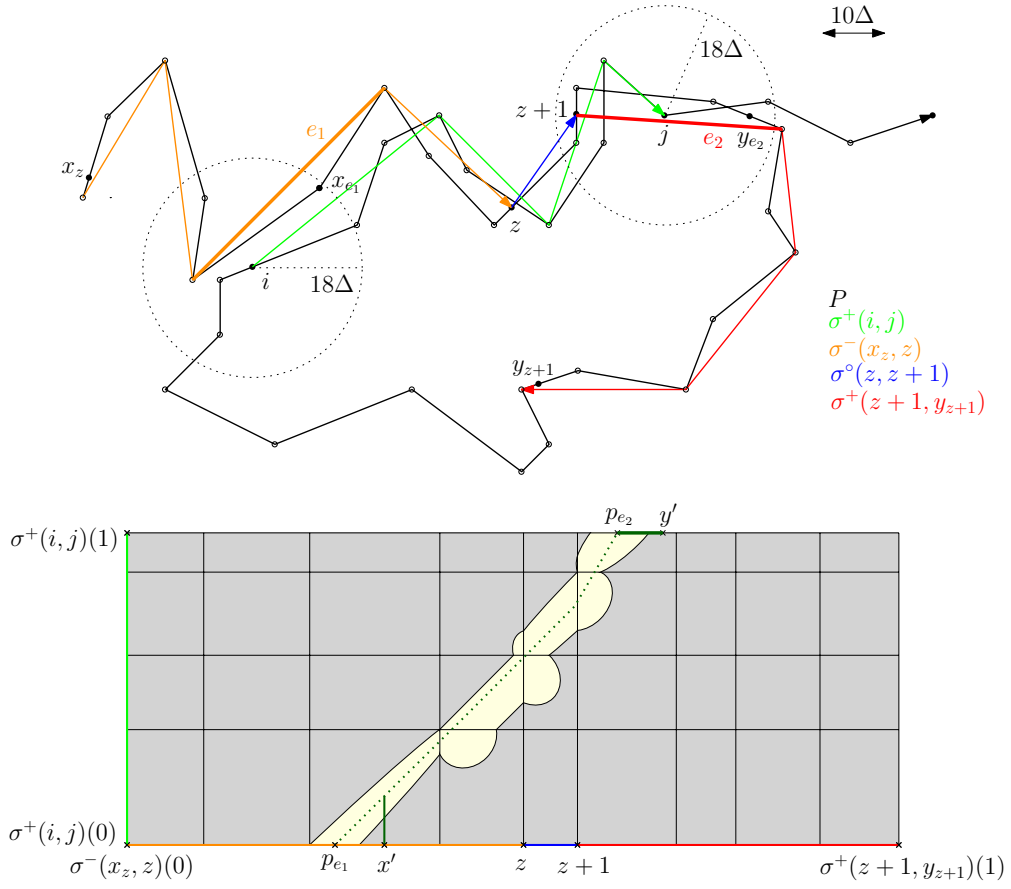


Figure 4.6: Example for a curve $P$ and breakpoints $z, i, j$ such that the approximation oracle returns "Yes" for the query $z \in r_{i,j}$. The path from $p_{e_1}$ to $p_{e_2}$ in the 10Δ-free space diagram is a monotone increasing path from the active edge $e_1$ to the active edge $e_2$. The edges are active since $d(P(t_i), P(t_{x_{e_1}})) \leq 18\Delta$ and $d(P(t_j), P(t_{y_{e_2}})) \leq 18\Delta$. The path from $x' = \sigma^-(x_{e_1}, z)(0)$ to $y' = \sigma^+(z+1, y_{e_2})(1)$ in the free space diagram gives a parametrization of $\kappa_z(x_{e_1}, y_{e_2})$ and $\sigma^+(i,j)$ yielding $d_F(\kappa_z(x_{e_1}, y_{e_2}), \sigma^+(i,j)) \leq 46\Delta$ as proven in Lemma 4.4.8.

**The query.** Given $z, i, j \in \{1, \ldots, m\}$ the oracle is therefore checking if $z \in r_{i,j}$ the following way:

First it builds a free space diagram of $\sigma^+(i,j)$ and $\kappa_z(x_z, y_{z+1})$ for the distance 10Δ. Then it checks for each edge on $\sigma^-(x_z, z)$ and on $\sigma^+(z+1, y_{z+1})$ if it is active. In the end, the oracle checks if there is a monotone increasing path in the 10Δ-free space that starts on an active edge of $\sigma^-(x_z, z)$ in one coordinate and $\sigma^+(i,j)(0)$ in the other coordinate and ends on an active edge of $\sigma^+(z+1, y_{z+1})$ in one coordinate and $\sigma^+(i,j)(1)$ in the other coordinate. The oracle returns "Yes" if such a path exists. See Figure 4.6 for an example of a "Yes" answer.

To do the above steps efficiently an underlying data structure for the oracle has to be built in the preprocessing. We will first show how the data structure is built and then prove the correctness of the oracle and analyze its running time.

**The data structure.** The data structure is built in two steps. The first step is to compute the simplifications. The second step consists of constructing a data structure for the breakpoints that can be used to determine active edges.

We compute the simplifications $\sigma^-(x_z, z)$, $\sigma^\circ(z, z+1)$ and $\sigma^+(z, y_z)$ for every breakpoint $z \in \{1, \ldots, m\}$ by running the algorithm of [4] up to complexity $2\ell$. For each edge $e$ of $\sigma^-(x_z, z)$ and $\sigma^+(z, y_z)$, we save the first breakpoint $x_e$ and the last breakpoint $y_e$ that corresponds to $e$.

In addition to these simplifications, the oracle also needs the simplification $\sigma^+(i, j)$ to build the free space diagram. Note that $\sigma^+(i, j)$ does not need to be stored in the data structure since for all $i, j \in \{1, \ldots, m\}$, the simplification $\sigma^+(i, j)$ can be constructed using $\sigma^+(i, j_i)$. To do so, the oracle does binary search to find the edge $e$ of $\sigma^+(i, j_i)$ such that $j$ corresponds to $e$. Then, the oracle computes the last point of $e$ that intersects the ball $B(t_j, 4\Delta)$. The subcurve of $\sigma^+(i, j_i)$ up to this point is $\sigma^+(i, j)$.

The oracle needs to determine which edges are active. For this, we construct a data structure in the same way as described for the case $\ell = 2$ in Section 4.3.3. We build an $m \times m$ matrix $M$ which stores the following information. For each breakpoint $i$ we go through the sorted list of breakpoints and check if $d(P(t_i), P(t_j)) \leq 18\Delta$ for each $1 \leq j \leq m$. While doing that, we determine for each $j$ which is the first breakpoint $z_{i,j} \geq j$ with $d(P(t_i), P(t_j)) \leq 18\Delta$. The entries $z_{i,j}$ are then stored in the matrix $M$.

Let $x_e(y_e)$ be the first (last) breakpoint corresponding to the edge $e$. To check if there is one breakpoint $z$ on an edge $e$ of a simplification such that $d(P(t_i), P(t_z)) \leq 18\Delta$ for some other breakpoint $i$, we only have to check if $z_{i,x_e} \geq y_e$. This is exactly what we need to check to decide if an edge is active and can be done in constant time given the matrix $M$.

Overall, the data structure therefore consists of $O(m)$ simplifications with pointers to the first (last) element of each edge and the matrix $M$ of size $O(m^2)$ containing the $z_{i,j}$-entries. This data structure is then used for each query to build a free space diagram and to find the active edges. The existence of a monotone increasing path is then tested by computing the reachability of active edges from active edges in the free space diagram. This can be done using the standard methods described by Alt and Godau [13] in the following way.

The free space diagram of the $10\Delta$-free space $F$ can be divided into cells that each correspond to a pair of edges, one from each curve $\kappa_z(x_z, y_{z+1})$ and $\sigma^+(i, j)$. Let us denote with $C_{s,t}$ the cell of the free space diagram corresponding to the $s$-th edge of $\sigma^+(i, j)$ and the $t$-th edge $e_t$ of $\kappa_z(x_z, y_{z+1})$. We further denote with $L_{s,t}$ and $B_{s,t}$ the left and bottom line segment bounding the cell $C_{s,t}$. We also define $L^F_{s,t} = L_{s,t} \cap F$ and $B^F_{s,t} = B_{s,t} \cap F$.

We need to calculate the reachable space $R \subseteq F$ where a point $p \in F$ is in $R$ if and only if there exists an active edge $e_t$ of $\sigma^-(x_z, z)$ such that there exists a monotone increasing path within $F$ from $B^F_{1,t}$ to $p$. We further define $L^R_{s,t} = L_{s,t} \cap R$ and $B^R_{s,t} = B_{s,t} \cap R$.

Note that given $L^R_{s,t}$, $B^R_{s,t}$,$L^F_{s+1,t}$ and $B^F_{s,t+1}$, we can construct $L^R_{s+1,t}$ and $L^R_{s,t+1}$ in constant time. So, given that we know for each edge $e_t$ of $\sigma^-(x_z, z)$, whether it is active or not, we can compute $L^R_{1,t}$ and $B^R_{1,t}$ for all edges $e_t$. With these we can iteratively construct all $L^R_{s,t}$ and $B^R_{s,t}$, proceeding row by row in the free space diagram.

Let $s^* \leq 2\ell$ be the number of edges of $\sigma^+(i, j)$. We get the following directly from the definition of $R$. There exists an active edge $e_t$ of $\sigma^+(z+1, y_{z+1})$ such that $B^R_{s^*+1,t} \neq \emptyset$ if and only if there is a monotone increasing path starting and ending in an active edge. So we only have to check for all active edges $e_t$ of $\sigma^+(z+1, y_{z+1})$ if $B^R_{s^*+1,t} \neq \emptyset$.

**Correctness.** To show the correctness of the oracle we show the following lemma.

**Lemma 4.4.8.** *Let $z, i, j \in \{1, \dots, m\}$. Consider the query $z \in r_{i,j}$. If the approximation oracle returns the answer*

*(i) "Yes", then there exists $x \in [x_z, z]$ and $y \in [z+1, y_{z+1}]$ with $d_F(\kappa_z(x,y), \sigma^+(i,j)) \leq 46\Delta$*

*(ii) "No", then we have $z \notin r_{i,j}$.*

*Proof.* i) Consider the $10\Delta$-free space diagram of $\kappa_z(x,y)$ and $\sigma^+(i,j)$. If the oracle returns the answer "Yes" then there is a monotone increasing path in the $10\Delta$-free space that starts on an active edge $e$ and ends on an active edge $e'$.

We show that this path implicitly gives two breakpoints $x_e \in [x_z, z]$ and $y_{e'} \in [z+1, y_{z+1}]$ as well as a monotone increasing path from $\sigma^-(x_e, z)(0)$ to $\sigma^+(z+1, y_{e'})(1)$ in the $46\Delta$-free space of $\kappa_z(x,y)$ and $\sigma^+(i,j)$.

Let $x_e$ be the first breakpoint corresponding to $e$ such that $d(P(t_{x_e}), P(t_i)) \leq 18\Delta$. Since $e$ is active, $x_e$ has to exist. We distinguish between the cases that the path starts in a point $p_e$ before or after $\sigma^-(x_e, z)(0)$ on $e$:

(I) The path starts in a point $p_e$ after $\sigma^-(x_e, z)(0)$ on $e$:

We have

$$\begin{aligned} & d(\sigma^+(i,j)(0), \sigma^-(x_e,z)(0)) \\ \leq\ & d(\sigma^+(i,j)(0), P(t_i)) + d(P(t_i), P(t_{x_e})) + d(P(t_{x_e}), \sigma^-(x_e,z)(0)) \\ \leq\ & 4\Delta + 18\Delta + 4\Delta \\ \leq\ & 26\Delta \end{aligned}$$

The second inequality above follows by the properties of $x_e$ and the fact that $\sigma^+(i,j)$ and $\sigma^-(x_e,z)$ are $(4\Delta, 2\ell)$-simplifications of $P[t_i, t_j]$ and $P[t_{x_e}, t_z]$. Since the path starts in a reachable area of the free space diagram, we have

$$d(\sigma^+(i,j)(0), p_e) \leq 10\Delta$$

Since $p_e$ and $\sigma^-(x_e,z)(0)$ lie both on the same edge of $\sigma^-(x_z, z)$, the segment $\overline{p_e, \sigma^-(x_e,z)(0)}$ is a subcurve of $\sigma^-(x_z,z)$. The Fréchet distance

$$d_F(\overline{p_e, \sigma^-(x_e,z)(0)}, \sigma^+(i,j)(0))$$

is at most

$$\max(d(\sigma^+(i,j)(0), \sigma^-(x_e,z)(0)), d(\sigma^+(i,j)(0), p_e)) \leq 26\Delta$$

since the Fréchet distance of a line segment and a point is attained at the start or end point of the line segment. The horizontal line segment from the point $(p_e, \sigma^+(i,j)(0))$ to the point $(\sigma^-(x_e,z)(0), \sigma^+(i,j)(0))$ is therefore contained in the $46\Delta$-free space of $\kappa_z(x,y)$ and $\sigma^+(i,j)$.

(II) The path starts in a point $p_e$ before $\sigma^-(x_e, z)(0)$ on $e$:

We again have

$$d(\sigma^+(i,j)(0), \sigma^-(x_e,z)(0)) \leq 26\Delta$$

and

$$d(\sigma^+(i,j)(0), p_e) \leq 10\Delta$$

Therefore we have

$$\begin{aligned} d(p_e, \sigma^-(x_e,z)(0)) &\leq d(p_e, \sigma^+(i,j)(0)) + d(\sigma^+(i,j)(0), \sigma^-(x_e,z)(0)) \\ &\leq 10\Delta + 26\Delta \\ &\leq 36\Delta \end{aligned}$$

The path has to pass the vertical line in the free space diagram through $\sigma^-(x_e,z)(0)$ at some height $h$. Note that the path is totally included in the $10\Delta$-free space. So for each point $p$ on $\sigma^+(i,j)[0,h]$ there is a point $q$ on $\sigma^-(x_z,z)$ between $p_e$ and $\sigma^-(x_e,z)(0)$ such that

$$d(p,q) \leq 10\Delta.$$

Because $q$ lies on the same edge of $\sigma^-(x_z,z)$ as $\sigma^-(x_e,z)(0)$ and $p_e$ we have

$$d(\sigma^-(x_e,z)(0), q) \leq d(\sigma^-(x_e,z)(0), p_e) \leq 36\Delta$$

and therefore

$$\begin{aligned} d(\sigma^-(x_e,z)(0), p) &\leq d(\sigma^-(x_e,z)(0), q) + d(q,p) \\ &\leq 36\Delta + 10\Delta \\ &\leq 46\Delta \end{aligned}$$

So we can replace the path in the $10\Delta$-free space starting at $p_e$ up to height $h$ with a vertical line segment from $(\sigma^-(x_e,z)(0), \sigma^+(i,j)(0))$ up to height $h$. This line segment is then fully contained in the $46\Delta$-free space.

By symmetry, we can apply the same arguments for changing the path in the free space diagram, so that the path ends in $\sigma^-(z+1,y)(1)$ for some breakpoint $y$. Therefore we can always find a monotone increasing path from $\sigma^-(x_e,z)(0)$ to $\sigma^+(z+1,y_{e'})(1)$ in the $46\Delta$-free space of $\kappa_z(x,y)$ and $\sigma^+(i,j)$. For an example of such a path see Figure 4.6. The vertical path starting in $x'$ is an example for Case II and the horizontal path from $p_{e_2}$ to $y'$ is an example for Case I (by symmetry for the end of the path).

ii) We prove that the oracle returns the answer "Yes" if $z \in r_{i,j}$:

So let $z \in r_{i,j}$ Then we have $d_F(\kappa_z(x,y), \sigma^+(i,j)) \leq 10\Delta$ for some $x_z \leq x \leq z$ and $z+1 \leq y \leq y_{z+1}$. Therefore there is a path in the free space diagram from $(\sigma^+(i,j)(0), \sigma^-(x,z)(0)))$ to $((\sigma^+(i,j)(1), \sigma^+(z+1,y)(1)))$. It remains to show that the edges corresponding to $x$ and $y$ are active. This follows by triangle inequality. In particular we have that $d(P(t_i), P(t_x))$ is at most

$$d(P(t_i), \sigma^+(i,j)(0)) + d(\sigma^+(i,j)(0), \sigma^-(x,z)(0)) + d(\sigma^-(x,z)(0), P(t_x))$$

and by the above this is at most $18\Delta$, and analogously $d(P(t_j), P(t_y)) \leq 18\Delta$. □

**Running time.** First, we analyze the preprocessing time needed to build the data structure for the oracle then we analyze the query time of the oracle.

Since one application of the algorithm of [4] needs $O(n\log(n))$ time and $O(n)$ space, we need $O(mn\log(n))$ time and $O(n+m\ell)$ space to construct the simplifications $\sigma^-(x_z,z)$, $\sigma^\circ(z,z+1)$ and $\sigma^+(z,y_z)$ for every $z \in \{1,\ldots,m\}$. To construct the pointers from each

edge to the first and last breakpoint on the edge we need an additional $O(m+\ell)$ time for each simplification. In total this needs at most $O(mn\log(n)+m^2)$ time and $O(n+m\ell)$ space.

To construct the matrix $M$ with the $O(m^2)$ entries of $z_{i,j}$ we need for each breakpoint $i$ a time of $O(m)$ and a space of $O(m)$ to go through the list of all $m$ breakpoints and save the entries of $z_{i,j}$. So in total, we need $O(m^2)$ time and $O(m^2)$ space for all entries. Combined with the time and space requirement for the simplifications we need $O(m(n\log(n)+m+\ell))$ time and $O(m\ell+m^2)$ space for the whole preprocessing.

To answer a query the oracle builds a free space diagram of $\sigma^+(i,j)$ and $\kappa_z(x_z,y_{z+1})$. To do that, it needs the simplifications $\sigma^+(i,j)$, $\sigma^-(x_z,z)$, $\sigma^\circ(z,z+1)$ and $\sigma^+(z+1,y_z)$. The simplifications $\sigma^-(x_z,z)$, $\sigma^\circ(z,z+1)$ and $\sigma^+(z+1,y_z)$ were already computed during preprocessing. The simplification $\sigma^+(i,j)$ can be computed in $O(\log(l))$ time with binary searches on $\sigma^+(i,y_i)$ and $\sigma^+(z,y_z)$. With the matrix $M$, it can be checked if an edge of $\sigma^-(x_z,z)$ or $\sigma^+(z+1,y_z)$ is active in $O(1)$ time. Therefore all active edges can be found in $O(\ell)$ time. The construction of the free space diagram of two curves with complexity $O(\ell)$ can then be done with standard methods as described earlier in $O(\ell^2)$ time. Testing the existence of a monotone increasing path from any of the active edges is then done as described above in the paragraph about the data structure. Note that given $L^R_{s,t}$, $B^R_{s,t}$,$L^F_{s+1,t}$ and $B^F_{s,t+1}$, we can construct $L^R_{s+1,t}$ and $L^R_{s,t+1}$ in $O(1)$ time. Therefore, given that we know for each edge $e_t$ of $\sigma^-(x_z,z)$ if it is active, we can compute $L^R_{1,t}$ and $B^R_{1,t}$ for all edges $e_t$ in $O(\ell)$ time. So we can compute all $L^R_{s,t}$ and $B^R_{s,t}$ in $O(\ell^2)$ time. Since $\sigma^+(z+1,y_{z+1})$ has at most $2\ell$ edges, the check for each of the active edges $e_t$ of $\sigma^+(z+1,y_{z+1})$ if $B^R_{s^*+1,t}\neq\emptyset$ can then be done in $O(\ell)$ time. This implies that testing if there exists a monotone increasing path with the described properties can be done in $O(\ell^2)$ time. Therefore the total query time is $O(\ell^2)$, as well. These results for the running time imply the following theorem.

**Theorem 4.4.9.** *One can build a data structure for the approximation oracle of size $O(m\ell+m^2)$ in time $O\left(m^2+mn\log(n)\right)$ and space $O(n+m\ell+m^2)$ that has a query time of $O(\ell^2)$.*

### 4.4.5 Applying the framework for computing a set cover

In order to apply Theorem 4.2.4 directly, we technically need to define a range space based on our data structure. Concretely, we define a new range space that is implicitly given by the approximation oracle. Let $I(z,(i,j))$ be the output of the approximation oracle for $z\in Z$ and $(i,j)\in T$ with

$$I(z,(i,j))=1 \quad \text{if the oracle answers "Yes"}$$
$$I(z,(i,j))=0 \quad \text{if the oracle answers "No"}$$

Let $\widetilde{\mathcal{R}}_4$ be the range space consisting of sets of the form

$$\tilde{r}_{i,j}=\{z\in Z\mid I(z,(i,j))=1\}$$

With Theorem 4.4.9 we immediately get

**Theorem 4.4.10.** *One can build a data structure in $O\left(m^2+mn\log(n)\right)$ time and $O(n+m\ell+m^2)$ space that answers for an element of the ground set $Z$ and a set of $\widetilde{\mathcal{R}}_4$, whether this element is contained in the set in $O(\ell^2)$ time. The data structure has a size of $O(m\ell+m^2)$.*

Since for all $(i, j)$ we have that $r_{i,j} \subseteq \tilde{r}_{i,j}$ it holds that for each set cover of $\widetilde{\mathcal{R}}_3$, there is also a set cover of the same size for $\widetilde{\mathcal{R}}_4$. Together with Lemma 4.4.6 this directly implies

**Lemma 4.4.11.** *If there exists a set cover $S$ of $\mathcal{R}$, then there exists a set cover of the same size for $\widetilde{\mathcal{R}}_4$.*

For the range space $\widetilde{\mathcal{R}}_4$ we further can derive a lemma corresponding to Lemma 4.4.5 using that for $z \in \tilde{r}_{i,j}$ we have $d_F(\kappa_z(x, y), \sigma^+(i, j)) \leq 46\Delta$. The proof is in all other parts completely analogous.

**Lemma 4.4.12.** *Assume there exists a set cover for $\mathcal{R}$ with parameter $\Delta$. Let $S$ be a set cover of size $k$ for $\widetilde{\mathcal{R}}_4$. We can derive from $S$ a set of $k$ cluster centers $C \subseteq \mathbb{X}_l^d$ and such that $\phi(P, C) \leq 50\Delta$.*

So if we apply Theorem 4.2.4 to the range space $\widetilde{\mathcal{R}}_4$ given by the approximation oracle we merely lose a constant approximation factor for our clustering problem in comparison to the direct application on the range space $\widetilde{\mathcal{R}}_3$. This leads to the following result.

### 4.4.6 The result

**Lemma 4.4.13.** *Let $k$ be the minimum size of a set cover for $\widetilde{\mathcal{R}}_4$. There exists an algorithm that computes a set cover for $\widetilde{\mathcal{R}}_4$ of size $O(k \log^2(m))$ with expected running time in $\tilde{O}\left(k\ell^2m^2 + mn\right)$ and using space in $O(n + m\ell + m^2)$.*

*Proof.* Note that we must have $k < m$ if such a set $C^*$ exists. Indeed, this is the case since for each $i \in \{1, \ldots, m-1\}$ the subcurve $P[t_i, t_{i+1}]$ has to be covered by only one element of $C^*$. So if we had $k > m - 1$ then we would have more center curves in $C^*$ than elements to cover. We apply Theorem 4.2.4 to compute a set cover of $(Z, \widetilde{\mathcal{R}}_4)$. For Theorem 4.2.4, we use Theorem 4.4.10, $|Z| = m - 1$ and $|\widetilde{\mathcal{R}}_4| = O(m^2)$. Again, the VC-dimension of the dual range space is bounded by $O(\log m)$. □

**Theorem 4.1.2.** *Let $P : [0, 1] \to \mathbb{R}^d$ be a polygonal curve of complexity $n$ with breakpoints $0 \leq t_1, \ldots, t_m \leq 1$. Assume there exists a set $C^* \subset \mathbb{X}_\ell^d$ of size $k \leq m$, such that $\phi(P, C^*) \leq \Delta$. Then there exists an algorithm that computes a set $C \subset \mathbb{X}_\ell^d$ of size $O(k \log^2(m))$ such that $\phi(P, C) \leq 50\Delta$. The algorithm has expected running time in $\tilde{O}\left(k\ell^2m^2 + mn\right)$ and uses space in $O(n + m\ell + m^2)$.*

*Proof.* The theorem follows directly by the combination of Lemma 4.4.13, Lemma 4.4.11 and Lemma 4.4.12. □

## 4.5 Improving the algorithm in the continuous case

In the previous sections, we considered the discrete variant of the subtrajectory clustering problem, assuming we are given breakpoints that denote the possible start and end points of subcurves that cover $P$. In the continuous case, we do not restrict the subcurves of $P$ to start and end at breakpoints. Recall that a point of $P$ is covered by a center curve $c$ if there is any subcurve $S$ of $P$ that contains $p$ and is in Fréchet distance at most $\Delta$ to $c$. In the continuous case we do not restrict $S$ to start and end at a breakpoint of $P$. The exact problem statement is given in Section 2.5.

In this section, we present an approximation algorithm that applies the algorithmic ideas developed in the previous sections to the discretization described in Section 2.5.1. A direct application of Theorem 4.1.2 using Lemma 2.5.2, however, leads to a high dependency on the arclength of the input curve, see also the discussion in Section 4.1.2. We will see that some steps of the algorithm can be simplified for this particular choice of breakpoints, ultimately leading to an improvement in the running time. Again, the crucial step is to choose the range space and the range space oracle wisely.

### 4.5.1 The range space

We will again use the range space $\widetilde{\mathcal{R}}_3$ that was defined in Section 4.4.2. Here we choose $m = \lceil \frac{\lambda}{\varepsilon\Delta} \rceil$ breakpoints to ensure that two consecutive breakpoints have a distance of at most $\varepsilon\Delta$. The explicit choice of breakpoints was already described in Section 2.5.1. For the construction of the approximation oracle we can take advantage of the fact that two consecutive breakpoints are close to each other. This will result in better running times based on the simpler structure of the oracle. A key factor is the low VC-dimension of the range space that is dual to the range space which is implicitly given by the oracle.

### 4.5.2 The approximation oracle

The new approximation oracle will have the following properties. Given a set $r_{i,j} \in \widetilde{\mathcal{R}}_3$ and an element $z \in Z$ this approximation oracle returns either one of the answers below:

(i) "Yes", in this case there exists a breakpoint $x \in [x_z, z]$ and a breakpoint $y \in [z+1, y_{z+1}]$ with $d_F(P[t_x, t_y], \sigma^+(i,j)) \leq (14+\varepsilon)\Delta$

(ii) "No", in this case $z \notin r_{i,j}$.

In both cases the answer is correct. Furthermore, we say that the new approximation oracle answers the query in the same way as the approximation oracle introduced in section 4.4.4 and therefore also needs the same data structures as before. There is only one exception. The oracle does not need to check if any edge is active and only needs to check if there is a monotone increasing path in the $10\Delta$-free space of $\sigma^+(i,j)$ and $\kappa_z(x_z, y_{z+1})$ that starts before or at $z$ and ends after or at $z+1$. So it also does not need to build the data structure for determining active edges. Neither does it have to save the first and last breakpoint on the edge of each simplification. As a direct consequence, we get the following running time result for the new approximation oracle.

**Theorem 4.5.1.** *One can build a data structure for the approximation oracle of size $O(m\ell)$ in time $O(mn\log(n))$ and space $O(n+m\ell)$ that has a query time of $O(\ell^2)$.*

**Correctness.** We want to show that the oracle is still correct, even though it does not check for active edges. To do so, we prove the following lemma.

**Lemma 4.5.2.** *Let $z, i, j \in \{1, \ldots, m\}$. Consider the query $z \in r_{i,j}$. If the approximation oracle returns the answer*

*(i) "Yes", then there exists $x \in [x_z, z]$ and $y \in [z+1, y_{z+1}]$ with $d_F(P[t_x, t_y], \sigma^+(i,j)) \leq (14+\varepsilon)\Delta$*

*(ii) "No", then we have $z \notin r_{i,j}$.*

*Proof.* (i) If the oracle returns "Yes", then there exists a monotone increasing path in the $10\Delta$-free space of $\sigma^+(i,j)$ and $\kappa_z(x_z, y_{z+1})$ that starts before or at $z$ and ends after or at $z+1$. Let $p$ be the start of the path on $\sigma^-(x_z, z)$. Let $q$ be a point of $P$ that gets mapped to $p$ by a strictly monotone increasing function from $P[t_{x_z}, t_z]$ to $\sigma^-(x_z, z)$ that realises the Fréchet distance $d_F(P[t_{x_z}, t_z], \sigma^-(x_z, z))$. So the last breakpoint $q_\varepsilon$ before $q$ has distance at most $\varepsilon\Delta$ to $q$. Therefore we have by triangle inequality

$$d(p, q_\varepsilon) \leq d(p, q) + d(q, q_\varepsilon) \leq (4+\varepsilon)\Delta$$

Since $d(p, q_\varepsilon) \leq (4+\varepsilon)\Delta$ and $d(p, q) \leq 4\Delta$, we also have for the line segment $\overline{q_\varepsilon q}$ that

$$d_F(p, \overline{q_\varepsilon q}) \leq (4+\varepsilon)\Delta$$

An analogous argument can be made for the end point $v$ of the path. So let $v$ get mapped to a point $u$ on $P$ by a strictly monotone increasing function from $\sigma^-(x_z, z)$ to $P[t_{x_z}, t_z]$ that realises the Fréchet distance $d_F(P[t_{x_z}, t_z], \sigma^-(x_z, z))$. For the first breakpoint $u_\varepsilon$ after $u$, we therefore get

$$d_F(v, \overline{uu_\varepsilon}) \leq (4+\varepsilon)\Delta$$

Let $\widetilde{\kappa}$ be the subcurve of $\kappa_z(x_z, y_{z+1})$ starting at $p$ and ending at $v$ and $\widetilde{P}$ be the subcurve of $P$ starting at $q$ and ending at $u$. By the definition of $\kappa_z(x_z, y_{z+1})$ as a $(4\Delta, 2\ell)$-simplification and the choices of $p, q, u$ and $v$, we get

$$d_F(\widetilde{\kappa}, \widetilde{P}) \leq 4\Delta$$

So by concatenation, we can get the curve

$$\widetilde{P}_\varepsilon = \overline{q_\varepsilon q} \oplus \widetilde{P} \oplus \overline{uu_\varepsilon}$$

which is a subcurve of $P$ with

$$d_F(\widetilde{\kappa}, \widetilde{P}_\varepsilon) \leq (4+\varepsilon)\Delta$$

By the use of triangle inequality, we now get

$$d_F(\sigma^+(i,j), \widetilde{P}_\varepsilon) \leq d_F(\sigma^+(i,j), \widetilde{\kappa}) + d_F(\widetilde{\kappa}, \widetilde{P}_\varepsilon) \leq (14+\varepsilon)\Delta$$

(ii) We prove that the oracle returns the answer "Yes" if $z \in r_{i,j}$:

So let $z \in r_{i,j}$ Then we have $d_F(\kappa_z(x,y), \sigma^+(i,j)) \leq 10\Delta$ for some $x_z \leq x \leq z$ and $z+1 \leq y \leq y_{z+1}$. Therefore there is a path in the free space diagram that starts before or at $z$ and ends after or at $z+1$. □

Now that we have shown that the oracle works correctly, we describe how we can use the oracle to approximate our problem. Analogous to the approach in the discrete case, we define a range space that is implicitly given by the new approximation oracle. Let $\widetilde{I}(z,(i,j))$ be the output of the approximation oracle for $z, i, j \in \{1, \ldots, m\}$ with

$$\widetilde{I}(z,(i,j)) = 1 \quad \text{if the oracle answers "Yes"}$$
$$\widetilde{I}(z,(i,j)) = 0 \quad \text{if the oracle answers "No"}$$

Let $\widetilde{\mathcal{R}}_5$ be the range space consisting of sets of the form

$$\tilde{r}_{i,j} = \{z \in Z \mid \widetilde{I}(z,(i,j)) = 1\}$$

With Theorem 4.5.1 we immediately get

**Theorem 4.5.3.** *One can build a data structure of size $O(m\ell)$ in time $O(mn\log(n))$ and $O(n+m\ell)$ space that answers for a breakpoint $z \in \{1,\dots,m\}$ and a set of $\widetilde{\mathcal{R}}_5$, whether $z$ is contained in the set in $O(\ell^2)$ time.*

We can also get the following results for the range space $\widetilde{\mathcal{R}}_5$ in the same way as before. We use that each range in $\widetilde{\mathcal{R}}_3$ is contained in a range of $\widetilde{\mathcal{R}}_5$. Together with Lemma 4.4.6 this directly implies

**Lemma 4.5.4.** *If there exists a set cover $S$ of $\mathcal{R}$, then there exists a set cover of the same size for $\widetilde{\mathcal{R}}_5$.*

To get the next result, we use that for $z \in \tilde{r}_{i,j}$ we have $d_F(\kappa_z(x,y),\sigma^+(i,j)) \leq (14+\varepsilon)\Delta$. Imitating the proof of Lemma 4.4.5 we then get

**Lemma 4.5.5.** *Assume there exists a set cover for $\mathcal{R}$ with parameter $\Delta$. Let $S$ be a set cover of size $k$ for $\widetilde{\mathcal{R}}_4$. We can derive from $S$ a set of $3k$ cluster centers $C \subseteq \mathbb{X}_l^d$ and such that $\phi(P,C) \leq (18+\varepsilon)\Delta$.*

These results imply that a minimum set cover of $\widetilde{\mathcal{R}}_5$ can be used to find an approximate solution for our clustering problem. But to apply Theorem 4.2.4 for finding a good set cover, we first need to bound the VC-dimension of the dual of $\widetilde{\mathcal{R}}_5$.

### 4.5.3 The VC-dimension

In our proof to bound the VC-dimension of the range space $\widetilde{\mathcal{R}}_5$ and its dual range space, we apply the techniques established in Chapter 3. We use the predicates defined in Section 3.3.2 and show that the output $\widetilde{I}(z,(i,j))$ of the approximation oracle can be determined by the truth value of these predicates. A bound for the VC-dimension then directly follows from the simplicity of the predicates. In particular, we consider the predicates $\mathcal{P}_1,\mathcal{P}_2,\mathcal{P}_7,\mathcal{P}_8$ for $V=\sigma^+(i,j)$, $W=\kappa_z(x_z,y_{z+1})$ and radius $10\Delta$. Let $v_1,\dots,v_{\ell_1}$ be the vertices of a polygonal curve $V$ and $w_1,\dots,w_{\ell_2}$ be the vertices of a polygonal curve $W$. Note that we have $\ell_1=O(\ell)$ and $\ell_2=O(\ell)$ for the case $V=\sigma^+(i,j)$ and $W=\kappa_z(x_z,y_{z+1})$. We restate the predicates and add a subscript to enable targeting specific predicates directly by using the indices $i,j$ and $t$.

i) $(\mathcal{P}_1)_{(i,j)}$: Given an edge of $V$, $\overline{v_jv_{j+1}}$ and a vertex $w_i$ of $W$, this predicate returns true iff there exists a point $p \in \overline{v_jv_{j+1}}$, such that $\|p-w_i\| \leq 10\Delta$.

ii) $(\mathcal{P}_2)_{(i,j)}$: Given an edge of $W$, $\overline{w_iw_{i+1}}$ and a vertex $v_j$ of $V$, this predicate returns true iff there exists a point $p \in \overline{w_iw_{i+1}}$, such that $\|p-v_j\| \leq 10\Delta$.

iii) $(\mathcal{P}_7)_{(i,j,t)}$: Given two vertices of $V$, $v_j$ and $v_t$ with $j<t$ and an edge of $W$, $\overline{w_iw_{i+1}}$, this predicate returns true if there exists two points $p_1$ and $p_2$ on the line supporting the directed edge, such that $p_1$ appears before $p_2$ on this line, and such that $\|p_1-v_t\| \leq 10\Delta$ and $\|p_2-v_j\| \leq 10\Delta$.

iv) $(\mathcal{P}_8)_{(i,j,t)}$: Given two vertices of $W$, $w_i$ and $w_t$ with $i<t$ and an edge of $V$, $\overline{v_jv_{j+1}}$, this predicate returns true if there exists two points $p_1$ and $p_2$ on the line supporting the directed edge, such that $p_1$ appears before $p_2$ on this line, and such that $\|p_1-w_t\| \leq 10\Delta$ and $\|p_2-w_i\| \leq 10\Delta$.

The proof of the following lemma is analogous to Lemma 9 in the full version [1] of the paper [2] by Afshani and Driemel. Much of the argumentation can be applied verbatim. We need to adapt the proof slightly since the range space is defined based on a partial alignment instead of a complete alignment. We include the proof here for the sake of completeness since there are some subtle differences.

**Lemma 4.5.6.** *Let $V$ and $W$ be two polygonal curves with vertices $v_1, \ldots, v_{\ell_1}$ and $w_1, \ldots, w_{\ell_2}$. Let further $1 \leq a \leq b \leq \ell_2$. Given the truth value of all predicates $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_7, \mathcal{P}_8$, one can determine if there exists a monotone increasing path in the $10\Delta$-free space of $V$ and $W$ that starts in $\overline{w_a w_{a+1}}$ at the bottom of the free space diagram and ends in $\overline{w_b w_{b+1}}$ at the top of the free space diagram.*

*Proof.* As in the proof of Lemma 9 in [1], we first introduce the notion of a valid sequence of cells in the free space diagram. We as well denote the cell corresponding to the edges $\overline{w_i w_{i+1}}$ and $\overline{v_j v_{j+1}}$ with $C_{i,j}$. The definition of a valid sequence, however, changes slightly for our application. We call a sequence of cells $\mathcal{C} = ((i_1, j_1), (i_2, j_2), \ldots, (i_k, j_k))$ valid if $i_1 = a, j_1 = 1, i_k = b, j_k = \ell_1 - 1$ and if for any two consecutive cells $(i_m, j_m)$ and $(i_{m+1}, j_{m+1})$ it holds that either $i_m = i_{m+1}$ and $j_{m+1} = j_m + 1$ or $j_m = j_{m+1}$ and $i_{m+1} = i_m + 1$. Here each tuple $(i, j)$ represents a cell $C_{i,j}$. The only difference to the definition in [1] is that we require $i_1 = a$ and $i_k = b$.

In our application, we say that a monotone increasing path in the $10\Delta$-free space of $V$ and $W$ is feasible if it starts in $\overline{w_a w_{a+1}}$ at the bottom of the free space diagram and ends in $\overline{w_b w_{b+1}}$ at the top of the free space diagram. It is easy to see that for any valid sequence, there exists a feasible path which passes the cells in the order of the sequence. On the other hand, it is also true that for each feasible path, there exists a valid sequence such that the path passes the cells in the order of the sequence. In the following, we identify with each sequence of cells $\mathcal{C}$ a set of predicates $\mathcal{P}$. The set of predicates is different from the predicates in [1] and consists of the following predicates.

i) $(\mathcal{P}_1)_{(i,j)} \in \mathcal{P}$ iff $(i, j-1), (i, j) \in \mathcal{C}$.

ii) $(\mathcal{P}_2)_{(i,j)} \in \mathcal{P}$ iff $(i-1, j), (i, j) \in \mathcal{C}$.

iii) $(\mathcal{P}_2)_{(a,1)} \in \mathcal{P}$ and $(\mathcal{P}_2)_{(b,\ell_1)} \in \mathcal{P}$

iv) $(\mathcal{P}_7)_{(i,j,k)} \in \mathcal{P}$ iff $(i, j-1), (i, k) \in \mathcal{C}$ and $j < k$.

v) $(\mathcal{P}_7)_{(a,1,k)} \in \mathcal{P}$ iff $(a, k) \in \mathcal{C}$ and $1 < k$.

vi) $(\mathcal{P}_7)_{(b,j,\ell_1-1)} \in \mathcal{P}$ iff $(b, j) \in \mathcal{C}$ and $j < \ell_1 - 1$.

vii) $(\mathcal{P}_7)_{(i,j,k)} \in \mathcal{P}$ iff $(i-1, j), (k, j) \in \mathcal{C}$ and $i < k$.

As in [1], we say that a valid sequence of cells is feasible if the conjunction of its induced predicates is true. We claim that any feasible path through the free space induces a feasible sequence of cells and vice versa. To prove the claim we use the following helper lemma from [1].

**Lemma 4.5.7** ([1], Lemma 10)**.** *Let $\mathcal{C}$ be a feasible sequence of cells and consider a monotonicity predicate $\mathcal{P}$ of the set of predicates $\mathcal{P}$ induced by $\mathcal{C}$. Let $a_1$ and $a_2$ be the vertices and let $e$ be the directed edge associated with $\mathcal{P}$. There exist two points $p_1$ and $p_2$ on $e$, such that $p_1$ appears before $p_2$ on $e$, and such that $\|p_1 - a_1\| \leq 10\Delta$ and $\|p_2 - a_2\| \leq 10\Delta$.*

Lemma 4.5.7 holds for our definition of feasible sequences of cells in the same way as in the original work. For the proof, we refer to [1]. To continue the proof of Lemma 4.5.6, we claim that any feasible path induces a feasible sequence of cells and vice versa. Assume there exists a feasible path $\pi$ that passes through the sequence of cells $\mathcal{C}$. The truth value of the predicates $(\mathcal{P}_2)_{(a,1)}$ and $(\mathcal{P}_2)_{(b,\ell_1)}$ follows directly by the starting and ending conditions of a feasible path. The truth value of the other predicates can be derived in the following way (which is exactly the same as in [1]).

Consider a horizontal vertex-edge predicate $(\mathcal{P}_2)_{(i,j)}$ for consecutive pairs of cells $C_{(i,j-1)}$, $C_{(i,j)}$ in the sequence $\mathcal{C}$. The path $\pi$ is a feasible path that passes through the cell boundary between these two cells. This implies that there exists a point on the edge $\overline{w_i w_{i+1}}$ which lies within distance $10\Delta$ to the vertex $v_j$. This implies that the predicate is true. A similar argument can be made for each vertex-edge predicate.

Next, we will discuss the monotonicity predicates. Consider a subsequence of cells of $\mathcal{C}$ that lies in a fixed column $i$ and consider the set of predicates $\mathcal{P}' \subseteq \mathcal{P}$ that consists of vertical monotonicity predicates $(\mathcal{P}_7)_{(i,j,k)}$ for fixed $i$. Let $p_j, p_{j+1}, \ldots, p_k$ be the sequence of points along $W$ that correspond to the vertical coordinates where the path $\pi$ passes through the corresponding cell boundaries corresponding to vertices $v_j, v_{j+1}, \ldots, v_k$. The sequence of points lies on the directed line supporting the edge $\overline{w_i w_{i+1}}$ and the points appear in their order along this line in the sequence due to the monotonicity of $\pi$. Since $\pi$ is a feasible path it lies in the free space and therefore we have $\|p_{k'} - v_{k'}\| \leq 10\Delta$ for every $j \leq k' \leq k$. This implies that all predicates in $\mathcal{P}$ are true. We can make a similar argument for the horizontal monotonicity predicates $(\mathcal{P}_8)_{(i,j,k)}$ for a fixed row $j$. This shows that a feasible path $\pi$ that passes through the cells of $\mathcal{C}$ implies that the conjunction of induced predicates $\mathcal{P}$ is true.

It remains to show the other direction. Since each cell of the free space is convex, it is clear that the vertex edge predicates give us the existence of a continuous (not necessarily monotone) path $\pi$ that stays inside the free space and connects the edges $\overline{w_a w_{a+1}}$ and $\overline{w_b w_{b+1}}$. To show that there always exists such a path that is also $(x, y)$-monotone we again use the argumentation of [1].

Assume for the sake of contradiction that the conjunction of predicates in $\mathcal{P}$ is true, but there exists no feasible path through the sequence of cells $\mathcal{C}$. In this case, it must be that either a horizontal passage or a vertical passage is not possible. Concretely, in the first case, there must be two vertices $v_j$ and $v_k$ and a directed edge $e = \overline{w_i w_{i+1}}$, such that there exist no two points $p_1$ and $p_2$ on $e$, such that $p_1$ appears before $p_2$ on $e$, and such that $\|p_1 - v_j\| \leq 10\Delta$ and $\|p_2 - v_k\| \leq 10\Delta$. However, $(P_3)_{(i,j,k)}$ is contained in $\mathcal{P}$ and by Lemma 4.5.7 two such points $p_1$ and $p_2$ must exist. We obtain a contradiction. In the second case, the argument is similar. Therefore, a feasible sequence of cells implies a feasible path, as claimed. □

Lemma 4.5.6 now directly implies the following theorem.

**Theorem 4.5.8.** *Given the truth values of all predicates $\mathcal{P}_1, \ldots, \mathcal{P}_8$ for two fixed curves $V = \sigma^+(i, j)$ and $W = \kappa_z(x_z, y_{z+1})$, one can determine the value of $\widetilde{I}(z, (i, j))$.*

Applying Theorem 4.5.8 and the techniques of Chapter 3 directly yields the following bound on the VC-dimension of $(Z, \widetilde{\mathcal{R}}_5)$ and its dual range space.

**Theorem 4.5.9.** *Let $Z = \{1, \ldots, m\}$. The VC-dimension of $(Z, \widetilde{\mathcal{R}}_5)$ and its dual range space are both in $O(d\ell \log(\ell))$.*

*Proof.* By Lemma 3.4.11 and 3.4.12 the predicates $\mathcal{P}_1, \ldots, \mathcal{P}_8$ are simple. Therefore it follows from Corollary 3.3.2 that the VC-dimension of $(Z, \widetilde{\mathcal{R}}_5)$ and its dual range space are both in $O(d\ell \log(\ell))$. □

### 4.5.4 The result

We apply Theorem 4.2.4 on the dual of $(\{1, \ldots, m\}, \widetilde{\mathcal{R}}_5)$ to get the following result for computing a set cover. We use here that $|\widetilde{\mathcal{R}}_5| = O(m^2)$ and that the result of Theorem 4.5.9 that the VC-dimension of $\widetilde{\mathcal{R}}_5^*$ is in $O(d\ell \log(\ell))$.

**Lemma 4.5.10.** *Let $k$ be the minimum size of a set cover for $\widetilde{\mathcal{R}}_5$. Let further $m = \lceil \frac{\lambda}{\varepsilon\Delta} \rceil$ and $\delta = O(d\ell \log(\ell))$, there exists an algorithm that computes a set cover for $\widetilde{\mathcal{R}}_5$ of size $O(\delta k \log(k))$ with expected running time in $\widetilde{O}(\delta k \ell^2 m^2 + mn)$ and using space in $O(n + m\ell)$.*

This lemma finally implies our main result for the clustering problem in the continuous case.

**Theorem 4.1.3** (Main Theorem)**.** *Let $P : [0,1] \to \mathbb{R}^d$ be a polygonal curve of complexity $n$, let $\ell \in \mathbb{N}$ and $\Delta > 0$ be parameters. Let $k$ be the minimum size of a solution to the $(\Delta, \ell)$-covering problem on $P$. Let further $\lambda(P)$ be the arc length of the curve $P$. There exists an algorithm that outputs a $(19, O(\ell \log(k) \log(\ell)))$-approximate solution. Let $m = \left\lceil \frac{\lambda(P)}{\Delta} \right\rceil$. The algorithm has expected running time in $\widetilde{O}\left(k\ell^3 m^2 + mn\right)$ and uses space in $O(n + m\ell)$.*

*Proof.* The theorem follows immediately by the combination of Lemma 4.5.10, 4.5.4 and 4.5.5. □

## 4.6 Additional lower bounds for the VC-dimension

In this section, we derive bounds on the VC-dimension of the dual range spaces in the discrete and continuous case. Consider the range space $\mathcal{R}$ from Section 2.5.1. The dual range space of $\mathcal{R}$ is the range space $\mathcal{R}^*$ with ground set $\mathbb{X}_\ell^d$ where each set $r_z \in \mathcal{R}^*$ is defined by a breakpoint $z \in \{1, \ldots, m-1\}$ as follows

$$r_z = \left\{ Q \in \mathbb{X}_\ell^d \mid \exists i \leq z < j \text{ with } d_F(Q, P[t_i, t_j]) \leq \Delta \right\}$$

In the continuous case, the dual range space is the range space $\mathcal{R}_0^*$ with ground set $\mathbb{X}_\ell^d$ where each set $r_t \in \mathcal{R}_0^*$ is defined by a parameter $t \in [0,1]$ as follows

$$r_t = \left\{ Q \in \mathbb{X}_\ell^d \mid \exists t' \leq t < t'' \text{ with } d_F(Q, P[t', t'']) \leq \Delta \right\}$$

Before deriving bounds on the VC-dimension of $\mathcal{R}^*$ and $\mathcal{R}_0^*$ in the general case, we observe that in the special case where cluster centers are points $\ell = 1$, there is a simple upper bound to the VC-dimension. In this chapter, we use the notation $b(p, \rho) = \{q \in \mathbb{R}^d \mid \|p - q\| \leq \rho\}$ for the Euclidean ball of radius $\rho \geq 0$ centered at $p \in \mathbb{R}^d$.

**Lemma 4.6.1.** *For $\ell = 1$, the VC-dimension of $(\mathbb{X}_\ell^d, \mathcal{R}_0^*)$ and $(\mathbb{X}_\ell^d, \mathcal{R}^*)$ are both at most $d + 1$.*

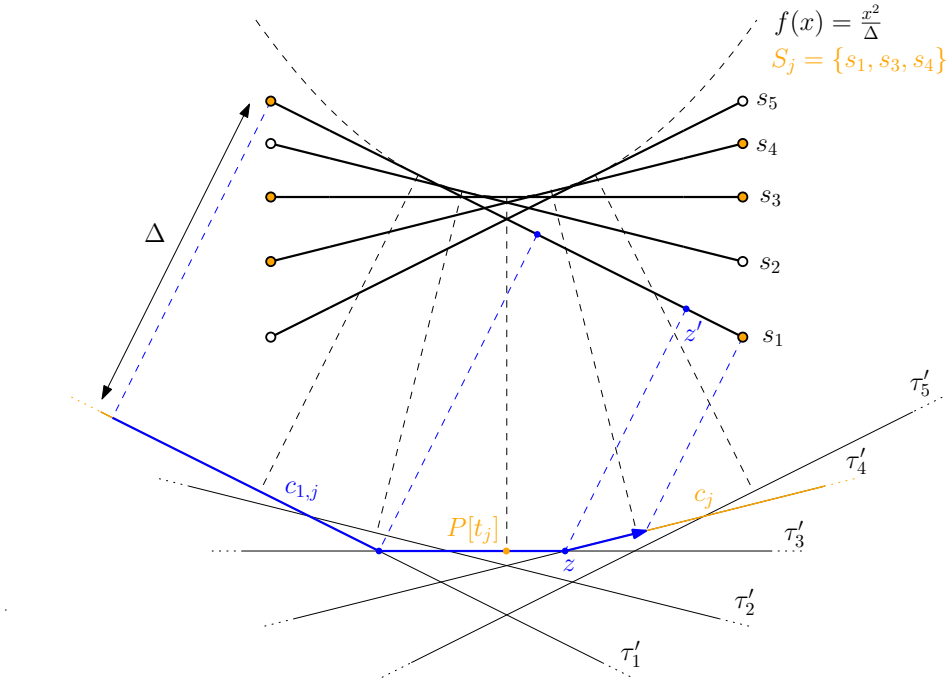


Figure 4.7: Construction for the case $\ell = 2$ such that the VC-dimension of $(\mathbb{X}_\ell^d, \mathcal{R}_0^*)$ is high.

*Proof.* We prove the bound for $(\mathbb{X}_1^d, \mathcal{R}_0^*)$ here. The proof works verbatim for $(\mathbb{X}_1^d, \mathcal{R}^*)$. The ground set of the range space is $\mathbb{X}_1^d = \mathbb{R}^d$. Now, consider a fixed $t \in [0, 1]$ and radius $\Delta > 0$. We claim that

$$r_t = b(P(t), \Delta).$$

Indeed, for any $0 \leq c \leq t \leq d \leq 1$, we can write for the set

$$R_{[c,d]} = \{p \in \mathbb{R}^d \mid d(p, P[c,d]) \leq \Delta\} = \bigcap_{s \in [c,d]} \{p \in \mathbb{R}^d \mid \|p - P(s)\| \leq \Delta\} \subseteq b(P(t), \Delta).$$

Thus, by the definition of $\mathcal{R}^*$,

$$r_t = \bigcup_{0 \leq c \leq t \leq d \leq 1} R_{[c,d]} = b(P(t), \Delta).$$

The claim now follows since the VC-dimension of Euclidean balls in $\mathbb{R}^d$ is equal to $d + 1$. □

### 4.6.1 Continuous case

We derive a lower bound on the VC-dimension of the dual range space $(\mathbb{X}_\ell^d, \mathcal{R}_0^*)$ in the general case.

**Theorem 4.6.2.** *For $\ell \geq 2$ and $d \geq 2$, the VC-dimension of $(\mathbb{X}_\ell^d, \mathcal{R}_0^*)$ is in $\Omega(\log(n))$.*

*Proof.* We show the lower bound for $\ell = 2$ and $d = 2$; this implies the bound for larger values of $\ell$ and $d$. Let $m \in \mathbb{N}$. We construct a curve $P$ with at most $O(4^m)$ vertices such

that the range space $(\mathbb{X}_2^2, \mathcal{R}_0^*)$ defined on $P$ shatters a set $S \subset \mathbb{X}_2^2$ of $m$ line segments. For the construction of $S = \{s_1, \ldots, s_m\}$ we choose line segments that are tangent to the parabola $f(x) = \frac{x^2}{\Delta}$. More specifically, let $\tau_i$ be the tangent that passes through $(x_i, y_i) = (\frac{\Delta(i-1)}{2(m-1)} - \frac{\Delta}{4}, \frac{(\frac{\Delta(i-1)}{2(m-1)} - \frac{\Delta}{4})^2}{\Delta})$. Then $s_i$ is the intersection of $\tau_i$ with the rectangle $[-\frac{2\Delta}{3}, \frac{2\Delta}{3}] \times [-\frac{\Delta}{2}, \frac{\Delta}{2}]$. The construction is visualized in Figure 4.7.

Consider the power set $2^S$. We show that for each subset $S_j \in 2^A$ there exists a curve $c_j \in \mathbb{X}_{m+1}^d$ such that for each $s_i \in S_j$ there exists a subcurve $c_{i,j}$ of $c_j$ with $d_F(s_i, c_{i,j}) \leq \Delta$ and for each $s_i \in S \setminus S_j$ there exists no subcurve $c_{i,j}$ of $c_j$ with $d_F(s_i, c_{i,j}) \leq \Delta$. The curve $P$ defining the range space instance will later be defined as a concatenation of these curves $c_j$. This will allow us to find a point $t_j$ on $c_j$ such that $r_{t_j} \cap S = S_j$ for each $j$, which then implies that $S$ can be shattered.

The curve $c_j$ can be generated as follows. Let $\tau_i'$ be the line parallel to $\tau_i$ that lies below $\tau_i$ and has distance $\Delta$ to $\tau_i$. For $S_j \in 2^S$ we define with $o_j$ the upper contour set of the lines $\tau_i'$ such that $s_i \in S_j$. We further define $c_j$ to be the intersection of $o_j$ with $[-2\Delta, 2\Delta] \times (-\infty, \infty)$. We observe that for $s_i \in S \setminus S_j$ the intersection of $b((x_i, y_i), \Delta)$ and $c_j$ is empty. Therefore there exists no subcurve $c_{i,j}$ of $c_j$ with $d_F(s_i, c_{i,j}) \leq \Delta$. For $s_i = \overline{p_i q_i} \in S_j$ let $l_{p_i}$ (resp. $l_{q_i}$) be the line perpendicular to $s_i$ that contains $p_i$ (resp. $q_i$). We define $c_{i,j}$ to be the subcurve of $c_j$ starting at the intersection of $l_{p_i}$ and $c_j$ and ending at the intersection of $l_{q_i}$ and $c_j$. To show that $d_F(c_{i,j}, s_i) \leq \Delta$, we divide $s_i$ into edges by projecting each vertex $z$ of $c_{i,j}$ orthogonal onto $s_i$. Since the slope of each edge of $c_j$ is between $-\frac{1}{2}$ and $\frac{1}{2}$ and also the slope of $s_i$ is between $-\frac{1}{2}$ and $\frac{1}{2}$, the projected vertices appear in the same order on $s_i$ as the corresponding vertices appear on $c_j$.

So to conclude that $d_F(c_{i,j}, s_i) \leq \Delta$, it remains to show that each vertex $z$ of $c_{i,j}$ has distance at most $\Delta$ to its projection $z'$ on $s_i$. This is enough because the Fréchet distance of two edges is attained at the distances of the start points or the end points of the edges. So let $z$ be a vertex of $c_{i,j}$. By construction, $c_{i,j}$ is part of the upper contour set $o_j$. We observe that the rectangle $[-\frac{2\Delta}{3}, \frac{2\Delta}{3}] \times [-\frac{\Delta}{2}, \frac{\Delta}{2}]$ that contains all line segments $S$ lies in the connected component of $\mathbb{R}^2 \setminus o_j$ that does not contain $\tau_i'$. Therefore the ray starting at $z'$ and containing $\overline{z'z}$ hits $z$ before or at the same time as it hits $\tau_i'$. So we have

$$d(z', z) \leq d(z', \tau_i') = \Delta$$

Note that the intersection $\bigcap_{i:s_i \in S_j} c_{i,j}$ always contains the intersection of $c_j$ with the vertical axis through $(0, 0)$. This is the case because the $x$-coordinate of the start point of each curve $c_{i,j}$ is smaller than 0 and the $x$-coordinate of the end point of each curve $c_{i,j}$ is greater than 0.

Let

$$P = \bigoplus_{j=1}^{2^m} c_j.$$

Since each curve $c_j$ has at most $m+1$ vertices, we get that $P$ has at most $n = m2^m = O(4^m)$ vertices and thus $m$ is in $\Omega(\log_4(n))$.

So, it remains to show that the set $S$ is shattered by $(\mathbb{X}_2^2, \mathcal{R}_0^*)$ defined on $P$. Indeed, for any $S_j \in 2^S$, let $t_j \in [0, 1]$ be the parameter such that $P[t_j]$ is the intersection of $c_j$ with the vertical axis through $(0, 0)$. We claim

$$r_{t_j} \cap S = S_j.$$

Since $P[t_j] \in \bigcap_{i:s_i \in S_j} c_{i,j}$, we get by the analysis above that $S_j \subseteq r_{t_j} \cap S$.

On the other hand, for each $s_i \in S \setminus S_j$ there exists no subcurve $c_{i,j}$ of $c_j$ with $d_F(s_i, c_{i,j}) \leq \Delta$. Note that the start points and end points of $c_j$ are by construction more than $\Delta$ away from any point on $s_i$. Therefore $d_F(s_i, Q) \leq \Delta$ for each subcurve $Q$ of $P$ that contains either the start point or the end point of $c_j$. So in total, we get that $s_i \in r_{t_j} \cap S$. □

### 4.6.2 Discrete case

Now we consider the range space $(\mathbb{X}_\ell^d, \mathcal{R}^*)$ that is dual to the range space $\mathcal{R}$, which was introduced in Section 2.5.1 to discretize our clustering problem through the addition of breakpoints.

We show that the VC-dimension of $(\mathbb{X}_\ell^d, \mathcal{R}^*)$ is in $\Theta(\log m)$ in the worst-case for any reasonable values of $d$ and $\ell$. Interestingly, our bounds on the VC-dimension are independent of $d$ and $n$. In fact, quite surprisingly, they also hold if $P$ is non-polygonal. The upper bound that the VC-dimension of $\mathcal{R}^*$ is at most $\log(m)$ follows directly from the upper bound on the size of the range space. It remains to show the lower bound.

**Theorem 4.6.3.** *For $d \geq 2$ and $\ell \geq 1$ the VC-dimension of $(\mathbb{X}_\ell^d, \mathcal{R}^*)$ is in $\Omega(\log m)$ in the worst-case.*

*Proof.* We show the lower bound for $\ell = 1$ and $d = 2$; this implies the bound for larger values of $\ell$ and $d$. To show the lower bound, we need to construct a set $A \subseteq \mathbb{R}^2$ with $|A| = t$ for $t \in \Omega(\log m)$, and a $P$ with breakpoints $t_1, \ldots, t_m$, such that $A$ is shattered by $\mathcal{R}^*$ as defined by $P$.

We use the lower bound construction of [57] for the VC-dimension of the range space of metric balls under the Fréchet distance centered at curves of complexity $t$ on the ground set $\mathbb{R}^2$. According to this result, we can find a set $A$ of $t$ points in $\mathbb{R}^2$, such that for every subset $A' \subseteq A$ we can find a curve $P_{A'} \in \mathbb{X}_t^d$, such that

$$A' = A \cap \{x \in \mathbb{R}^2 | d_F(x, P_{A'}) \leq \Delta\} \tag{4.2}$$

We will now construct $P$ as the concatenation of these curves with breakpoints at the start and endpoints of these curves, where to concatenate them we linearly interpolate between the endpoints of consecutive curves.

In order to show the correctness of the resulting construction we observe that the definition of the Fréchet distance can be simplified if one of the curves is a point. Let $x \in \mathbb{R}^2$ and let $P' = P[t_i, t_j]$, then

$$d_F(x, P') = \max_{t \in [0,1]} (x, P'(t)) \tag{4.3}$$

This implies that for the case $\ell = 1$ our range space $\mathcal{R}^*$ actually has a simpler structure. In particular, any $r_z \in \mathcal{R}^*$ defined by an index $z \in Z$ can be rewritten as follows

$$\begin{aligned} r_z &= \left\{x \in \mathbb{R}^d \mid \exists i \leq z \leq j \text{ with } d_F(x, P[t_i, t_j]) \leq \Delta\right\} && (4.4)\\ &= \bigcup_{i \leq z < j} \left\{x \in \mathbb{R}^d \mid d_F(x, P[t_i, t_j]) \leq \Delta\right\} && (4.5)\\ &= \left\{x \in \mathbb{R}^d \mid d_F(x, P[t_z, t_{z+1}]) \leq \Delta\right\} && (4.6) \end{aligned}$$

Thus, with our choice of $P$ and breakpoints $t_1 < \cdots < t_m$, we have that for any $A' \subseteq A$ there exists an index $z$ with $1 \leq z < m$, such that $A' = A \cap r_z$ holds as required

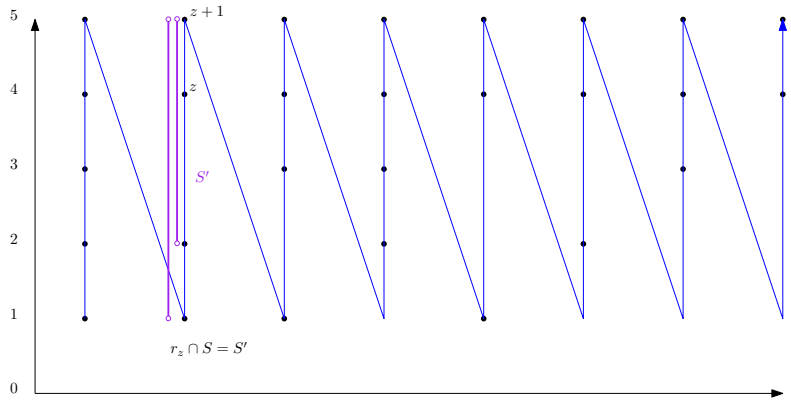


Figure 4.8: Schematic drawing of $P : [0, 1] \to \mathbb{R}$ in the construction for the lower bound to the VC-dimension. Parameters of the construction are $\Delta = \frac{1}{3}$, $\ell = 2$ and $t = 3$. The shattered set of line segments in $\mathbb{R}$ is $S = \{\overline{1,5}\,\overline{2,5}, \overline{3,5}\}$ with $|S| = t$. The subset encoder segments are shown vertically upwards, the connector segments are shown diagonally downwards. The horizontal axis shows the parametrization of the curve. The figure also shows the subset $S' = \{\overline{1,5}\,\overline{2,5}\}$ and indicates the breakpoint at index $z$, such that $r_z \cap S = S'$.

by (4.2). Finally, the number of breakpoints we used is $m = 2^{t+1}$ (two breakpoints for each subset of $A$). Therefore, we have $t \geq \log(m) - 1$. □

**Theorem 4.6.4.** *For $d \geq 1$ and $\ell \geq 2$ the VC-dimension of $\mathcal{R}^*$ is in $\Omega(\log m)$ in the worst-case.*

*Proof.* We construct a curve $P$ with breakpoints as follows. Let $t \in \mathbb{N}$ be a parameter of the construction. Let $\Delta = \frac{1}{3}$. The curve $P$ is constructed from a series of $2^t$ line segments starting at 0 and ending at $t + 2$ with certain breakpoints along these line segments to be specified later. We call these segments ***subset encoder segment***. These line segments are connected by $2^t - 1$ line segments starting at $t + 2$ and ending at 0. Those line segments will not contain any breakpoints and we call them ***connector segments***. Let $A = \{1, \ldots, t\}$. For each subset $A' \subseteq A$, we create one subset encoder segment with breakpoints at the values of $A'$, in addition, we put two breakpoints at the values $t + 1$ and at $t + 2$. The curve $P$ is defined by concatenating all $2^t$ subset encoder segments with the connector segments in between. Figure 4.8 shows an example of this construction for $t = 3$. Now, consider the following set of line segments in $\mathbb{R}$. $S = \{\overline{s_1 s_2} \mid s_1 \in A, s_2 = t + 2\}$. We claim that $S$ is shattered by $\mathcal{R}^*$ defined on $P$ and $\Delta$. Therefore, the VC-dimension is $t$. The number of breakpoints $m$ we used is upper-bounded by $(t + 2)2^t$ and therefore $t \geq \Omega(\log m)$. □

# Chapter 5

# Faster Subtrajectory Clustering

The main content of this chapter previously appeared as the paper *Faster Approximate Covering of Subcurves Under the Fréchet Distance* [25] by Frederik Brüning, Jacobus Conradi and Anne Driemel which was published in the proceedings of the *30th Annual European Symposium on Algorithms (ESA 2022)*. A full version of the paper is available on arXiv [26]. This chapter considers the same subtrajectory problem as the previous chapter, i.e. the $(\Delta, \ell)$-covering problem. The approximation algorithms in this chapter improve upon the ones from Chapter 4 in approximation factors as well as expected running time and space requirement. The chapter extends [25] with an additional analysis of the VC-dimension of the underlying range space in Sections 5.3.4 and 5.3.5. The additional analysis is based on the techniques developed in Chapter 3 and improves the constants of the already constant VC-dimension which then directly implies better constants in the solution size.

## 5.1 Introduction

In this chapter, we study subtrajectory clustering under the Fréchet distance. As in the previous chapter, we conduct this study by designing a bicriterial approximation algorithm to the $(\Delta, \ell)$-covering problem (see Section 2.5 for the problem definition). The resulting algorithm improves upon the algorithm from Chapter 4 in the dependency of the running time and space requirement on the arclength of the input curve and also has slightly better approximation factors. A key factor for this improvement is a different approach for the discretization of the underlying set cover problem. Instead of a direct discretization of the input curve, we first simplify the input curve and then discretize it by distributing points evenly along the simplification. We restrict subcurves to start and end at these points and also restrict the selectable center curves to be subedges of the simplification that start and end at these points. Another important factor is an improvement in the adaptation of the multiplicative weight update method that enables us to keep track of the weights implicitly. Our approach leads to the following results.

### 5.1.1 Results

Our main result in Theorem 5.4.9 is an algorithm that computes an $(\alpha, \beta)$-approximate solution to the $(\Delta, \ell)$-covering problem with $\alpha \in O(1)$ and $\beta = O(\ell \log(k\ell))$, where $k$ is the size of an optimal solution. The algorithm needs $O\left(n(k\ell)^3 \log^4\left(\frac{n\lambda}{\Delta k\ell}\right) + n\log^2(n)\right)$

time in expectation and $O\left(n(k\ell)^2 \log^2\left(\frac{n\lambda}{\Delta k\ell}\right)\right)$ space, where $\lambda$ is the arclength of the input curve. Here, we stated our results for general $\ell$ using the reduction described at the end of this section.

In our algorithm, we again use the variant of the multiplicative weights update method described in Section 2.4.2. As shown in Theorem 4.6.2, the range space underlying the $(\Delta, \ell)$-covering problem initially has high VC-dimension, —namely $\Theta(\log n)$ in the worst-case. Similar to the approach in the last chapter, we circumvent the high VC-dimension by defining an intermediate set cover problem where the VC-dimension is significantly reduced. A key idea that enables our results is a curve simplification that requires the curve to be locally maximally simplified, a notion that is borrowed from de Berg, Cook, and Gudmundsson [50]. The algorithm improves the dependency on the relative arclength from quadratic to polylogarithmic as compared to the previous result in Theorem 4.1.3.

**Reduction to line segments** In the remainder of the paper, we will focus on finding a $\Delta$-covering with line segments, that is $\ell = 2$. The following lemma provides the reduction for general $\ell$ at the expense of an increased approximation factor.

**Lemma 5.1.1.** *Let $P \in \mathbb{X}_n^d$ be a polygonal curve, $\Delta \in \mathbb{R}_+$ and $\ell \in \mathbb{N}$. Let $C \subseteq \mathbb{X}_\ell^d$ be a $\Delta$-covering of $P$ of minimum cardinality. There exists a set of line segments $C' \subseteq \mathbb{X}_2^d$ that is a $\Delta$-covering of $P$ with $|C'| \leq (\ell - 1)|C|$.*

*Proof.* Choose as set $C'$ the union of the set of edges of the polygonal curves of $C$. Clearly, this set has the claimed cardinality and is a $\Delta$-covering of $P$. □

**Remark** Note that using Theorem 4.1.3 for line segments and afterwards applying Lemma 5.1.1 to get a result for general $\ell$ would directly improve the previous result of Theorem 4.1.3 to an $(\alpha, \beta)$-approximate solution with $\alpha \in O(1)$ and $\beta = O(\ell \log(k\ell))$ that needs in expectation $\widetilde{O}\left(k\ell(\frac{\lambda}{\Delta})^2 + \frac{\lambda n}{\Delta}\right)$ time and $O\left(n + \frac{\lambda}{\Delta}\right)$ space.

### 5.1.2 Roadmap

In Section 5.2 we develop a structured variant of our problem that allows us to apply the multiplicative weight update method (see Section 2.4.2) in a more efficient way than in Chapter 4. Our intermediate goal in this section is to obtain a structured set of candidates for a modified covering problem that is on the one hand easy to compute and on the other hand sufficient to obtain good approximation bounds for the original problem. In Section 5.2.1, we define a notion of curve simplification that is inspired by the work of de Berg, Gudmundsson and Cook [50]. A crucial property of this simplification is that subcurves of the input are within small Fréchet distance to subcurves of constant complexity of the simplification. In Section 5.3, we specify the range space that can be used in combination with the multiplicative weights update method to achieve approximate solutions. Crucially, we show that the VC-dimension of the induced range space which is implicitly used by our algorithm is small by design (see Sections 5.3.1 to 5.3.5). The Sections 5.3.2 and 5.3.3 present a VC-dimension bound that is derived with the techniques from [26] and the Sections 5.3.4 and 5.3.5 present an improved bound that is derived with techniques from Chapter 3. In Section 5.4, we analyze in how we can adapt the multiplicative weights update method to get an approximation algorithm. The adaptation mainly consists of specifying how we compute simplifications (Section 5.4.1), how we implement the verifier (Section 5.4.2) and how we

implement the data structure for maintaining the probability distribution (Section 5.4.3). Based on this adaptation, we finally prove our main results in Section 5.4.4.

## 5.2 Structuring the solution space

In this section, we introduce key concepts that allow us to transfer the problem to a set cover problem on a finite range space with small VC-dimension and still obtain good approximation bounds.

### 5.2.1 Simplifications and containers

We start by defining the notion of curve-simplification that we will use throughout the paper.

**Definition 5.2.1** (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 an index set $1 \leq i_1 < \ldots < i_k \leq n$ that defines 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 the simplification is $\Delta$-**good**, if the following properties hold:*

*(i)* $\|p_{i_j} - p_{i_{j+1}}\| \geq \frac{\Delta}{3}$ *for* $1 \leq j < k$

*(ii)* $d_F(P[t_{i_j}, t_{i_{j+1}}], \overline{p_{i_j}\, p_{i_{j+1}}}) \leq 3\Delta$ *for all* $1 \leq j < k$.

*(iii)* $d_F(P[t_1, t_{i_1}], \overline{p_{i_1}\, p_{i_1}}) \leq 3\Delta$ *and* $d_F(P[t_{i_k}, t_n], \overline{p_{i_k}\, p_{i_k}}) \leq 3\Delta$

*(iv)* $d_F(P[t_{i_j}, t_{i_{j+2}}], \overline{p_{i_j}\, p_{i_{j+2}}}) > 2\Delta$ *for all* $1 \leq j < k-1$

Our intuition is the following. Property $(i)$ guarantees that $S$ does not have short edges. Property $(ii)$ and $(iii)$ together tell us, that the simplification error is small. Property $(iv)$ tells us, that the simplification is (approximately) maximally simplified, that is, we cannot remove a vertex, and hope to stay within Fréchet distance $2\Delta$ to $P$.

**Definition 5.2.2** (Container)**.** *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 $\pi$ on $S$ as $S[t_a, t_b]$, with $a = \max(\{i_1\} \cup \{i \in I \mid t_i \leq s\})$ and $b = \min(\{i \in I \mid t_i \geq t\} \cup \{i_k\})$.*

The following lemma has been proven by de Berg et al. [50]. We restate and reprove it here with respect to our notion of simplification.

**Lemma 5.2.3.** *[50] Let $P$ be a polygonal curve in $\mathbb{R}^d$, and let $S$ be a $\Delta$-good simplification of $P$. Let $Q$ be an edge in $\mathbb{R}^d$ and let $\pi$ be a subcurve of $P$ with $d_F(Q, \pi) \leq \Delta$. Then $c_S(\pi)$ consists of at most 3 edges and $c_S(\pi)$ can not contain 3 vertices of $S$.*

*Proof.* Assume for the sake of contradiction, that $c_S(\pi)$ contains 3 vertices $s_1, s_2, s_3$ of $S$. Note that in the case that $c_S(\pi)$ has 4 edges, it has three internal vertices. By Definition 5.2.2 these three vertices are also interior vertices of $\pi$. As the Fréchet distance $d_F(Q, \pi) \leq \Delta$, there are points $q_1, q_2, q_3 \in Q$, that get matched to $s_1, s_2$ and $s_3$ respectively during the traversal, with $\|s_i - q_i\| \leq \Delta$. This implies $d_F(\pi[s_1, s_3], \overline{q_1\, q_3}) \leq \Delta$. It also implies, that $d_F(\overline{s_1\, s_3}, \overline{q_1\, q_3}) \leq \Delta$. But then

$$d_F(\overline{s_1\, s_3}, P[s_1, s_3]) = d_F(\overline{s_1, s_3}, \pi[s_1, s_3]) \leq d_F(\overline{s_1\, s_3}, \overline{q_1\, q_3}) + d_F(\pi[s_1, s_3], \overline{q_1\, q_3}) \leq 2\Delta,$$

contradicting the assumption that $S$ is a $\Delta$-good simplification. □

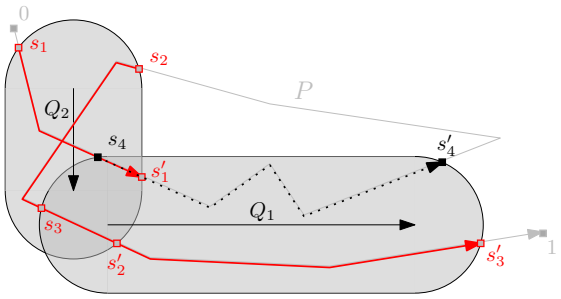


Figure 5.1: Example of the structured $\Delta$-coverage of a set $C = \{Q_1, Q_2\}$ and a curve $P$. Here we have $\Psi'_\Delta(P, C) = [s_1, s'_1] \cup [s_2, s'_3]$ since the subcurves $P[s_1, s'_1]$ and $P[s_2, s'_2]$ have Fréchet distance $\Delta$ to $Q_1$ and $P[s_3, s'_3]$ has Fréchet distance $\Delta$ to $Q_2$. Note that $[s_4, s'_4]$ is not part of the coverage since the subcurve $P[s_4, s'_4]$ consists of 4 edges.

### 5.2.2 Structured coverage and candidate space

We want to make use of the property of $\Delta$-good simplifications shown in Lemma 5.2.3. For this, we adapt the notion of $\Delta$-coverage from Section 2.5 as follows.

**Definition 5.2.4.** *Let $S$ be a polygonal curve in $\mathbb{R}^d$. Let $(t_1, \ldots, t_n)$ be the vertex-parameters of $S$. Let $\ell \in \mathbb{N}$ and $\Delta \in \mathbb{R}$ be fixed parameters. Define the **structured $\Delta$-coverage** of a set of center curves $C \subset \mathbb{X}^d_\ell$ as*

$$\Psi'_\Delta(S, C) = \bigcup_{q \in C} \bigcup_{(i,j) \in J} \Psi^{(i,j)}_\Delta(S, q)$$

*where*

$$\Psi^{(i,j)}_\Delta(S, q) = \{s \in [t, t'] \mid t_i \leq t \leq t_{i+1};\ t \leq t';\ t_{j-1} \leq t' \leq t_j;\ d_F(S[t, t'], q) \leq \Delta\},$$

*and where $J = \{1 \leq i < j \leq n \mid 1 \leq j - i \leq 3\}$.*

*If it holds that $\Psi'_\Delta(S, C) = [0, 1]$, then we call $C$ a **structured $\Delta$-covering** of $S$.*

**Observation 5.2.5.** *In general for any polygonal curve $S$ and set of center curves $C$ it holds that $\Psi'_\Delta(S, C) \subseteq \Psi_\Delta(S, C)$.*

We now want to restrict the candidate set to subedges of a simplification of the input curve, thereby imposing more structure on the solution space. For this, we begin by defining a more structured parametrization of the set of edges of a polygonal curve.

**Definition 5.2.6** (Edge space)**.** *We define the **edge space** $\mathbb{T}_n = \{1, \ldots, n-1\} \times [0, 1]$. We denote the set of edges of $P$ with $E(P)$.*

**Definition 5.2.7** (Candidate space)**.** *Let $E = \{e_1, \ldots, e_{n-1}\}$ be an ordered set of edges in $\mathbb{R}^d$. We define the **candidate space** induced by $E$ as the set $\mathcal{Z}_E = \{(i_1, t_1, i_2, t_2) \in \mathbb{T}_n \times \mathbb{T}_n \mid i_1 = i_2\}$. We associate an element $(i, t_1, i, t_2) \in \mathcal{Z}_E$ with the subedge $\overline{e_i(t_1)\, e_i(t_2)}$. We may abuse notation by denoting the associated edge to an element $t \in \mathcal{Z}_E$ simply with $t$.*

The following theorem summarizes and motivates the above definitions of structured coverage and candidate space. Namely, we can restrict the search space to subedges of the simplification $S$ and still obtain a good covering of $P$. Moreover, we can evaluate the coverage of our solution solely based on $S$. The structured coverage only allows subcurves of $S$ that consist of at most three edges to contribute to the coverage. This technical restriction is necessary to obtain a small VC-dimension in our main algorithm later on, and it is well-motivated by Lemma 5.2.3. The proof of the theorem is rather technical and we divert it to Section 5.2.3.

**Theorem 5.2.8.** *Let $S$ be a $\Delta$-good simplification of a curve $P$. Let $C$ be a set of subedges of edges of $S$. If $C$ is a structured $8\Delta$-covering of $S$, then $C$ is an $11\Delta$-covering of $P$. Moreover, if $k$ is the size of an optimal $\Delta$-covering of $P$, then there exists such a set $C$ of size at most $3k$.*

To get a finite set of candidates, we can approximate the candidate space $\mathcal{Z}_E$.

**Definition 5.2.9** ($\varepsilon$-approximate candidate set)**.** *Let $E$ be a set of edges in $\mathbb{R}^d$ and let $\varepsilon > 0$ be a parameter. We can approximate the candidate space induced by $Q$ by subsampling the edges as follows. Define $G_\delta := \{i \cdot \delta \mid i \in \mathbb{Z}\}$. For each edge $e_i \in E$ consider the set*

$$X_i = ([0,1] \cap G_{\varepsilon/\lambda_i}) \cup \{1\}$$

*where $\lambda_i$ is the length of the edge $e_i$. The $\varepsilon$-approximate candidate set induced by $E$ is the set*

$$Z_{\varepsilon,E} = \bigcup_{i=1}^{m} \{(i,x,i,y) \mid x \in X_i, y \in X_i\}$$

*Assuming $E$ fixed, observe that for any edge $p \in \mathcal{Z}_E$, there exists an edge $p' \in Z_{\varepsilon,E}$, such that $d_F(e,e') \leq \varepsilon$.*

By applying triangle inequality, we directly get the following corollary to Theorem 5.2.8.

**Corollary 5.2.10.** *Let $S$ be a $\Delta$-good simplification of a curve $P$. Let $C \subseteq Z_{\Delta,E}$ . If $C$ is a structured $9\Delta$-covering of $S$, then $C$ is an $12\Delta$-covering of $P$. Moreover, if $k$ is the size of an optimal $\Delta$-covering of $P$, then there exists such a set $C$ of size at most $3k$.*

### 5.2.3 Proof of Theorem 5.2.8

We want to prove Theorems 5.2.8. We will use the following observation on the Fréchet distance of a curve and its simplifications.

**Observation 5.2.11.** *Let $P$ be a polygonal curve, and let $S$ be a $\Delta$-good simplification of $P$, defined by the index set $I = (i_1, \ldots, i_k)$. Then $d_F(P,S) \leq 3\Delta$. Moreover, there is a traversal $(f_S, g_S)$ with $0 \leq t_1 \leq \ldots \leq t_k \leq 1$, such that $P(f_S(t_j)) = S(g_S(t_j)) = p_{i_j}$, with associated distance at most $3\Delta$. This can be seen by concatenating the traversals induced by conditions $(ii)$ and $(iii)$ on the respective subcurves.*

The following Lemma motivates the use of the simplification $S$. It shows that for any covering of $P$ there exists a suitable structured covering of $S$. Moreover, we can transfer a structured cover of $S$ back to $P$.

**Lemma 5.2.12.** *Let $P$ be polygonal curve, and let $S$ be a $\Delta$-good simplification of $P$. Let $P'$ be a polygonal curve, with $d_F(P, P') \leq \Delta'$. Assume there exists a set $C \subset \mathbb{X}_2^d$ of cardinality $k$, such that $\Psi_\Delta(P, C) = [0, 1]$. Then*

*(i)* $\Psi_{\Delta+\Delta'}(P', C) = [0, 1]$ *and*

*(ii)* $\Psi'_{4\Delta}(S, C) = [0, 1]$.

*Proof.* We start by proving $(i)$. Let $(f, g)$ be a traversal of $P$ and $P'$, with associated cost at most $\Delta'$. 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 $\Delta'$. 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

$$[0, 1] = \mu_P([0, 1]) = \mu_P\left(\Psi_\Delta(P, C)\right) \subseteq \Psi_{\Delta+\Delta'}(P', C)$$

This would imply the set inclusion $[0, 1] \subseteq \Psi_{\Delta+\Delta'}(P', C)$, which then also implies equality, since by definition $[0, 1] \supseteq \Psi_{\Delta+\Delta'}(P', C)$.

We argue as follows. Observe that by triangle inequality it holds for any $t, t' \in [0, 1]$ and any $Q \in C$ with $d_F(P[t, t'], Q) \leq \Delta$, and for any $s \in \mu_P(t)$ and $s' \in \mu_P(t')$ that

$$d_F(P'[s, s'], Q) \leq d_F(P'[s, s'], P[t, t']) + d_F(P[t, t'], Q) \leq \Delta + \Delta'$$

Therefore we can write

$$\begin{aligned}
\mu_P\left(\Psi_\Delta(P, C)\right) &= \bigcup_{Q \in C} \bigcup_{0 \leq t \leq t' \leq 1} \{x \in \mu_P([t, t']) \mid d_F(P[t, t'], Q) \leq \Delta\} \\
&\subseteq \bigcup_{Q \in C} \bigcup_{0 \leq s \leq s' \leq 1} \{x \in [s, s'] \mid d_F(P'[s, s'], Q) \leq \Delta + \Delta'\} \\
&= \Psi_{\Delta+\Delta'}(S, C)
\end{aligned}$$

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

Now for $(ii)$ notice, that when $P' = S$ is a $\Delta$-good simplification of $P$, and $\Delta' = 3\Delta$, $S[s, s']$ is contained in $c_S(P[t, t'])$ for $\mu_P([t, t']) = [s, s']$. This follows from the traversal given in Observation 5.2.11 together with the definition of $\mu_P$. Thus, because $c_S(P[t, t'])$ consists of at most three edges by Lemma 5.2.3, it holds that

$$\bigcup_{Q \in C} \bigcup_{0 \leq s \leq s' \leq 1} \{x \in [s, s'] \mid d_F(S[s, s'], Q) \leq 4\Delta\} \subset \Psi'_{4\Delta}(S, C),$$

implying the claim. □

The following lemma shows that we can restrict the search for a covering to the subedges of the simplification.

**Lemma 5.2.13.** *Let $S$ be a $\Delta$-good simplification of some curve $P$. Assume there exists a set $C \subset \mathbb{X}_2^d$ of size $k$, such that $\Psi_\Delta(P, C) = [0, 1]$. Then there exists a set $C_S \subset \mathcal{Z}_{E(S)}$ of cardinality at most $3k$ such that $\Psi'_{8\Delta}(S, C_S) = [0, 1]$.*

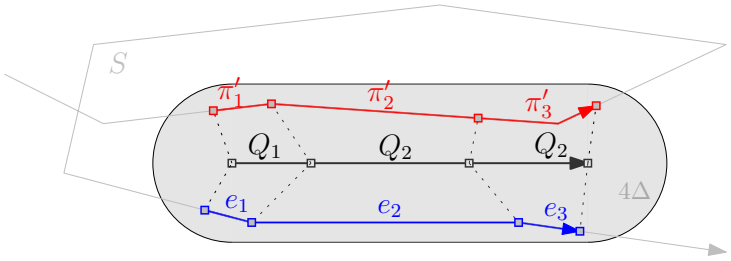


Figure 5.2: Illustration to the proof of Lemma 5.2.13. The example illustrates that the $8\Delta$-coverage of the edges $e_1$, $e_2$ and $e_3$ is a superset of the $4\Delta$-coverage of $Q = Q_1 \oplus Q_2 \oplus Q_3$.

*Proof.* We start by applying Lemma 5.2.12 $(ii)$ to $P$ and its $\Delta$-good simplification. Thus $\Psi'_{4\Delta}(S, C) = [0, 1]$. We show that for each center curve in $Q \in C$ there exist 3 subcurves of edges of $S$ that cover all the parts of $S$ that were covered by $Q$. An illustration of the proof is given in Figure 5.2. Let $Q \in C$ with $\Psi'_{4\Delta}(S, \{Q\}) \neq \emptyset$. We fix a subcurve $\pi$ of $S$ such that $d_F(\pi, Q) \leq 4\Delta$, that consists of at most 3 edges $e_1$, $e_2$, $e_3 \in \mathcal{Z}_{E(S)}$. $\pi$ exists by Lemma 5.2.3. The curve $Q$ can be split into 3 subcurves $Q_1$, $Q_2$, $Q_3$ such that $Q = Q_1 \oplus Q_2 \oplus Q_3$ with $d_F(Q_1, e_1) \leq 4\Delta$, $d_F(Q_2, e_2) \leq 4\Delta$ and $d_F(Q_3, e_3) \leq 4\Delta$.

Now consider an arbitrary subcurve $\pi'$ of $S$ such that $d_F(\pi', Q) \leq 4\Delta$. The curve $\pi'$ can be split into 3 subcurves $\pi'_1$, $\pi'_2$, $\pi'_3$ such that $\pi' = \pi'_1 \oplus \pi'_2 \oplus \pi'_3$ with $d_F(\pi'_1, Q_1) \leq 4\Delta$, $d_F(\pi'_2, Q_2) \leq 4\Delta$ and $d_F(\pi'_3, Q_3) \leq 4\Delta$. By triangle inequality we get

$$d_F(\pi'_1, e_1) \leq d_F(\pi'_1, Q_1) + d_F(Q_1, e_1) \leq 8\Delta.$$

In the same way we obtain $d_F(\pi'_2, e_2) \leq 8\Delta$ and $d_F(\pi'_3, e_3) \leq 8\Delta$. It follows that the entire curve $\pi'$ is covered by $e_1$, $e_2$ and $e_3$. By applying this argument to every subcurve $\pi'$ of $S$ with $d_F(\pi', Q) \leq 4\Delta$ with $\pi'$ consisting of at most three edges, we get

$$\Psi'_{4\Delta}(S, \{Q\}) \subseteq \Psi'_{8\Delta}(S, \{e_1, e_2, e_3\}).$$

Applying this to every $Q \in C$ and using the fact that $\Psi'_{8\Delta}(S, C) = \bigcup_{Q \in C} \Psi'_{8\Delta}(S, \{Q\})$, the lemma is implied by constructing the set $C_S$ out of the edges $e_1, e_2, e_3$ for each $Q \in C$. □

Using the above two lemmas, we can prove Theorem 5.2.8, which was the main theorem in Section 5.2.2.

*Proof of Theorem 5.2.8.* Lemma 5.2.13 implies that there exists a set $C$ of subedges of edges of $S$ of size $3k$ which is a structured $8\Delta$-covering of $S$. By Observation 5.2.5, $C$ is also an $8\Delta$-covering of $S$. Now, Observation 5.2.11 implies that $d_F(P, S) \leq 3\Delta$ and thus we can apply Lemma 5.2.12 $(i)$ with $P' = S$ and $\Delta' = 3\Delta$ to conclude that $C$ is an $11\Delta$-covering of $P$. □

## 5.3 A new range space for approximation

In this section, we present a new range space for which the multiplicative weight update method described in Section 2.4.2 can be applied to find an approximate solution to the $(\Delta, 2)$-covering problem for a polygonal curve $P$. The range space is built with respect to a $\Delta$-good simplification $S$ of $P$. To formally describe the range space, we introduce the notion of feasible sets.

**Definition 5.3.1** (Feasible set)**.** *Let $S : \mathbb{T}_n \to \mathbb{R}^d$ be a polygonal curve and let $B \subset \mathbb{X}_2^d$ be a candidate set of edges and let $\Delta \geq 0$ be a real value. For any point $t \in \mathbb{T}_n$, we define the **feasible set** of $t$ as the set of all elements $Q \in B$ with the following property: There exist $i, j$ with $1 \leq j - i \leq 3$ and $a, b$ with $t_i \leq a \leq t \leq b \leq t_j$ such that $d_F(S[a, b], Q) \leq \Delta$. We denote the feasible set of $t$ with $F_\Delta(t)$.*

Note that for any fixed $\Delta$-good simplification $S$ and $B = Z_{\varepsilon,E(S)}$ the feasible set $F_{\Delta,\varepsilon}(t)$ contains all center curves in $Z_{\varepsilon,E(S)}$ that can cover $t$ in a structured $\Delta$-covering of $S$. We study the range space

$$(X, \mathcal{R}) = (Z_{\Delta,E(S)}, \{F_{9\Delta}(t) \mid t \in \mathbb{T}_n\}).$$

A hitting set of $\{F_{9\Delta}(t) \mid t \in \mathbb{T}_n\}$ is a structured $9\Delta$-covering of $S$ and therefore by Corollary 5.2.10 a $12\Delta$-covering of $P$. The bounds in Section 5.3.4 and 5.3.5 are improvements of the bounds in Section 5.3.2 and 5.3.3.

techniques from [26] and the Sections 5.3.4 and 5.3.5 present an improved bound that is derived with techniques from Chapter 3.

To find small hitting sets, we want to apply the multiplicative weight for hitting sets described in Section 2.4.2 on $(Z_{\Delta,E(S)}, \{F_{9\Delta}(t) \mid t \in \mathbb{T}_n\})$. To do so, we need to bound the VC-dimension of the range space. In the following sections, we will analyze the structure of the range space and derive a VC-dimension bounds based on it and techniques from Section 2.4.3 and Chapter 3.

### 5.3.1 On the structure of feasible sets

We claim that any feasible set can be split into sets corresponding to the edges of the simplification, where each set consists of a constant union of rectangles in the candidate space restricted to the respective edge. Figure 5.3 illustrates one of those rectangles. The following lemma provides the formal statement.

**Lemma 5.3.2.** *Let $P$ be a polygonal curve in $\mathbb{R}^d$ and let $e \in \mathbb{X}_2^d$ be an edge. Let $(t_1, \ldots, t_n)$ be the vertex-parameters of $P$. For any integer values $1 \leq i < j \leq \min(i + 3, n)$ and real value $t \in [0, 1]$ with $t_i \leq t \leq t_j$, either there exist $\alpha_1, \alpha_2, \beta_1, \beta_2$ such that*

$$R := \{(\alpha, \beta) \in [0, 1]^2 \mid t \in \Psi_\Delta^{i,j}(P, e[\alpha, \beta])\} = [\alpha_1, \alpha_2] \times [\beta_1, \beta_2],$$

*or the set $R$ is empty. Moreover, each $\alpha_v$ (respectively $\beta_v$) for $v \in \{1, 2\}$ can be written as $\alpha_v = c_v + \sqrt{d_v}$ (respectively $\beta_v = e_v + \sqrt{f_v}$), where the parameters $c_v$ and $d_v$ (respectively $e_v$ and $f_v$) can be computed by an algorithm that takes $(i, j), t$ and $e$ as input and needs $O(d)$ simple operations.*

*Proof.* Let $R = \{(\alpha, \beta) \in [0, 1]^2 \mid t \in \Psi_\Delta^{i,j}(P, e[\alpha, \beta])\}$. We first show that either $R$ corresponds to a rectangle $[\alpha_1, \alpha_2] \times [\beta_1, \beta_2]$ in the parameter space $[0, 1]^2$ of $e$ or $R = \emptyset$. In Figure 5.3, we give an example for the construction of the rectangle $R$. Let $i < v < j$. The intersection of the $\Delta$-free space $\mathcal{D}_\Delta(P, e)$ with the edge $t_v \times [0, 1]$ is either a free space interval of the form $t_v \times [l_v, u_v]$ or is empty. Also the intersection with $t \times [0, 1]$ has the form $t \times [l_t, u_t]$ or is empty. If either of the intersections is empty for $t$ or some $i < v < j$ then $R$ is empty, since no point on $e$ is within distance $\Delta$ of $P(t)$ respectively $P(t_v)$. Otherwise for all $i < v < j$ the parameters $l_v$,$u_v$, as well as $l_t$ and $u_t$ are well

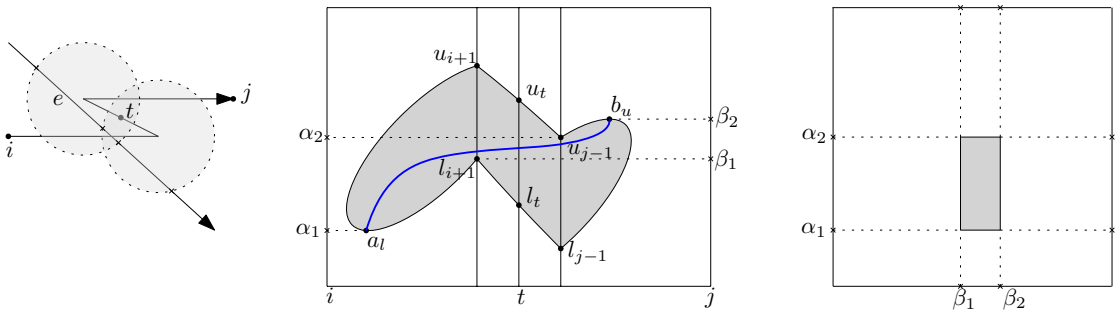


Figure 5.3: Example for the construction of the rectangle $R = [\alpha_1, \alpha_2] \times [\beta_1, \beta_2]$ for fixed $P, i, j, t, \Delta$ and $e$. The left image shows the curves $P[t_i, t_j]$ and $e$ with two circles of radius $\Delta$ around $P(t_{i+1})$ and $P(t_{j-1})$. The middle image shows the corresponding $\Delta$-free space diagram with a continuous monotone increasing path from $a_l$ to $b_u$ and the right image shows the rectangle $R$ in the parameter space $[0, 1]^2$ of $e$.

defined. In the case that $j - i = 3$ and $l_{i+1} > u_{i+2}$ we have $R$ is empty since there is no bi-monotone path in the free space that first passes $t_{i+1} \times [l_{i+1}, u_{i+1}]$ and then $t_{i+2} \times [l_{i+2}, u_{i+2}]$. In the following we therefore assume $l_{i+1} \leq u_{i+2}$ for $j - i = 3$. Let further $a = (a_i, a_l)$ be the lowest point in the cell of the $\Delta$-free space corresponding to the edge $P[t_i, t_{i+1}]$ and $b = (b_j, b_u)$ be the highest point in the cell of the $\Delta$-free space corresponding to the edge $P[t_{j-1}, t_j]$. We define the following parameters

$$\alpha_1 = \begin{cases} l_t & \text{for } a_i \geq t \\ a_l & \text{else} \end{cases}$$
$$\alpha_2 = \min(u_{i+1}, \ldots, u_{j-1}, u_t)$$
$$\beta_1 = \max(l_{i+1}, \ldots, l_{j-1}, l_t)$$
$$\beta_2 = \begin{cases} u_t & \text{for } b_j \leq t \\ b_u & \text{else} \end{cases}$$

and show that $R$ corresponds to a rectangle $[\alpha_1, \alpha_2] \times [\beta_1, \beta_2]$ in the parameter space $[0, 1]^2$ of $e$. To do so we first show that for each $\alpha \in [\alpha_1, \alpha_2]$ and $\beta \in [\beta_1, \beta_2]$ with $\alpha \leq \beta$ there is a $t_\alpha \in [t_i, t_{i+1}]$ and a $t_\beta \in [t_{j-1}, t_j]$ such that

$$d_F(P[t_\alpha, t_\beta], e) \leq \Delta.$$

The case $\beta < \alpha$ is analogous. So let $\alpha \in [\alpha_1, \alpha_2]$, $\beta \in [\beta_1, \beta_2]$ with $\alpha \leq \beta$. By the definition of $\alpha_1, \alpha_2$ it directly follows that there is a $t_\alpha \in [t_i, \min(t_{i+1}, t)]$ such that $(t_\alpha, \alpha)$ is in the free space $\mathcal{D}_\Delta(P, e)$. By the definition of $\beta_1, \beta_2$ it also follows that there is a $t_\beta \in [\max(t_{j-1}, t), t_j]$ such that $(t_\beta, \beta)$ is in the free space $\mathcal{D}_\Delta(P, e)$. We split the analysis on how to construct a monotone increasing path in the free space from $(t_\alpha, \alpha)$ to $(t_\beta, \beta)$ in three cases depending on $j - i$.

Case $j - i = 1$: Since the free space is convex in each cell and $\alpha \leq \beta$, the path $(t_\alpha, \alpha) \oplus (t_\beta, \beta)$ is a bi-monotone path in $\mathcal{D}_\Delta(P, e)$.

Case $j - i = 2$: Let $\gamma = \min(u_{i+1}, \beta)$. Since $\beta \geq \beta_1 \geq l_{i+1}$ and therefore $u_{i+1} \geq \gamma \geq l_{i+1}$, we have $(t_{i+1}, \gamma) \in \mathcal{D}_\Delta(P, e)$. By the convexity of each cell in the free space and the fact that $\alpha \leq \gamma \leq \beta$, we get that

$$(t_\alpha, \alpha) \oplus (t_{i+1}, \gamma) \oplus (t_\beta, \beta)$$

is a monotone increasing path in $\mathcal{D}_\Delta(P, e)$.

Case $j - i = 3$: Let $\gamma_1 = \min(u_{i+1}, u_{i+2}, \beta)$ and $\gamma_2 = \min(u_{i+2}, \beta)$. By $l_{i+1} \leq \beta_{i+1} \leq \beta$ and $l_{i+1} \leq u_{i+2}$, we get $(t_{i+1}, \gamma_1) \in \mathcal{D}_\Delta(P, e)$. Further by $l_{i+2} \leq \beta_1 \leq \beta$, we get $(t_{i+2}, \gamma_2) \in \mathcal{D}_\Delta(P, e)$. By the convexity of each cell in the free space diagram and the fact $\alpha \leq \gamma_1 \leq \gamma_2 \leq \beta$, we get similar to the last case that

$$(t_\alpha, \alpha) \oplus (t_{i+1}, \gamma_1) \oplus (t_{i+2}, \gamma_2) \oplus (t_\beta, \beta)$$

is a bi-monotone path in $\mathcal{D}_\Delta(P, e)$.

It remains to show that no other $(\alpha, \beta) \in [0, 1]^2$ with $\alpha \leq \beta$ and $e[\alpha, \beta] \in R$ exist. So we show that for $\alpha \notin [\alpha_1, \alpha_2]$ there is no $t_\alpha \in [t_i, t_{i+1}]$, $t_\beta \in [t_{j-1}, t_j]$, $\beta \geq \alpha$ such that

$$d_F(P[t_\alpha, t_\beta], e[\alpha, \beta]) \leq \Delta.$$

Case $\alpha < \alpha_1$: By the definition of $\alpha_1$ there exists no $t_\alpha \in [t_i, t_{i+1}]$ such that $\|P(t_\alpha) - e(\alpha)\| \leq \Delta$.

Case $\alpha > \alpha_2$: Assume there exists a $i < v < j$ such that $\alpha_2 = u_v$. Then by the definition of $u_v$ there exists no $\gamma \in e[\alpha, 1]$ such that $\|P(t_v) - e(\gamma)\| \leq \Delta'$. Otherwise, we have $\alpha_2 = u_t$. Then by the definition of $u_t$, there exists no $\gamma \in e[\alpha, 1]$ such that $\|P(t_v) - e(\gamma)\| \leq \Delta$. The combination of the two cases directly implies the claim.

Now we analyze the running time to compute the parameters $c_1, c_2, d_1, d_2, e_1, e_2, f_1, f_2$ that define $\alpha_1, \alpha_2, \beta_1, \beta_2$. Each of the parameters $\alpha_1, \alpha_2, \beta_1, \beta_2$ is equal to one parameter of the set $K = \{u_{i+1}, \ldots, u_{j-1}, l_{i+1}, \ldots, l_{j-1}, a_l, b_u, l_t, u_t\}$ with $|K| \leq 8$. Each of the intervals $[l_v, u_v]$ as well as $[l_t, u_t]$ correspond to the intersections of a line with a ball in $\mathbb{R}^d$. The points $a = (a_i, a_l)$ and $b = (b_i, b_j)$ are defined by the intersections of a line with the union of a cylinder and two balls in $\mathbb{R}^d$. Each extreme point $\kappa \in K$ of such an intersection can be written as $\kappa = \kappa_1 + \sqrt{\kappa_2}$ with parameters $\kappa_1$ and $\kappa_2$ that can be computed with $O(d)$ simple operations (see Lemma 17 and 18 in [57]). For each parameter $\gamma \in \{\alpha_1, \alpha_2, \beta_1, \beta_2\}$, we can find $\kappa^* \in K$ with $\kappa^* = \gamma$ by comparisons of $O(1)$ elements of $K$ with each other. For each two element of $\kappa, \kappa' \in K$ this can be done with $O(1)$ simple operations, given the parameters $\kappa_1, \kappa_2, \kappa_1', \kappa_2'$ with $\kappa = \kappa_1 + \sqrt{\kappa_2}$ and $\kappa' = \kappa_1' + \sqrt{\kappa_2'}$. So in total we need $O(d)$ simple operations to compute the parameters $c_1, c_2, d_1, d_2, e_1, e_2, f_1, f_2$ that define $\alpha_1, \alpha_2, \beta_1, \beta_2$. □

### 5.3.2 Analysis of the VC-dimension

To prove a VC-dimension bound of $O(d^2)$ for the range space $\{F_\Delta(t) \mid t \in \mathbb{T}_n\}$ with ground set $\mathbb{X}_2^d$, we use the above lemma and bound the VC-dimension with the help of Theorem 2.4.16 based on the number of simple operations that are needed to answer a range query. A more detailed bound on the VC-dimension of our range space with an analysis of the constant factors can be found in Lemma 5.3.8.

**Lemma 5.3.3.** *Let $S : \mathbb{T}_n \to \mathbb{R}^d$ be a polygonal curve and let $\Delta \in \mathbb{R}_+$. The VC-dimension of the range space $\{F_\Delta(t) \mid t \in \mathbb{T}_n\}$ with ground set $\mathbb{X}_2^d$ is in $O(d^2)$.*

*Proof.* Let $t = (t', i') \in \mathbb{T}_n$. We define the parameter vector $v_t$ of $t$ as

$$v_t = (t', i', S(t_{i'-3}), S(t_{i'-2}), \ldots, S(t_{i'+4})) \in \mathbb{R}^{8d+2}$$

where $t_j \coloneqq t_1$ for all $j < 1$ and $t_j \coloneqq t_n$ for all $j > n$. We further define a function $h : \mathbb{R}^{8d+2} \times \mathbb{X}_2^d \to \{0, 1\}$ with $h(v_t, Q) = 1$ if and only if $Q \in F_\Delta(t)$. In order to show the

lemma, we first argue that for any given $v_t \in \mathbb{R}^{8d+2}$ and $Q \in \mathbb{X}_d^2$ the expression $h(t, Q)$ can be evaluated with $O(d)$ simple operations.

Let $(t', i') = t$ and let $J = \{(i, j) \mid t_i \leq t \leq t_j;\ 1 \leq j - i \leq 3\}$. Note that $|J| \leq 9$ and $i' - 3 \leq i \leq j \leq i' + 4$ for each $(i, j) \in J$ since $t_i \leq t \leq t_{i+1}$. Furthermore, $J$ can be determined by $O(1)$ simple operations from $i'$. Note that $Q \in F_\Delta(t)$ if and only if $t \in \Psi_\Delta^{i,j}(S, Q)$ for some $(i, j) \in J$. So, for fixed $(i, j)$, consider the set

$$R = \{(\alpha, \beta) \in [0, 1]^2 \mid t \in \Psi_\Delta^{i,j}(S, Q[\alpha, \beta])\}.$$

Lemma 5.3.2 implies that $R$ is either empty or can be written as a rectangle $[\alpha_1, \alpha_2] \times [\beta_1, \beta_2]$. Note that $t \in \Psi_\Delta^{i,j}(S, Q)$ if and only if $R$ is non-empty and $(0, 1) \in R$. By Lemma 5.3.2, this test can be performed using $O(d)$ simple operations. Thus, we can apply Theorem 2.4.16 and conclude that the VC-dimension of $(\{F_\Delta(t) \mid t \in \mathbb{T}_n\}, \mathbb{X}_2^d)$ is in $O(d^2)$. □

### 5.3.3 Detailed analysis of the VC-dimension

In Lemma 5.3.3, it was shown that the VC-dimension of the range space $\{F_\Delta(t) \mid t \in \mathbb{T}_n\}$ with ground set $\mathbb{X}_2^d$ is in $O(d)$. In this section, we explicitly derive constants $a$ and $b$ such that this VC-dimension is at most $ad + b$. This is necessary to obtain exact bounds on the sample size to be used in the main algorithm. To do so, we give a detailed pseudocode (Algorithm 2) for the check if an element $e \in \mathbb{X}_2^d$ is in the set $F_\Delta(t)$ and analyze the simple operations that are needed for each step of the algorithm. In the algorithm, we refer to specific free space intervals by using the following notation. Let $C_i$ be the cell in the $\Delta$-free space of a curve $S$ and an edge $e$ corresponding to the $i$th edge of $S$ and $e$. We denote with $I_{i,0}^h = [a_{i,0}, b_{i,0}]$ the horizontal free space interval that bounds $C_i$ from below and with $I_{i,1}^h = [a_{i,1}, b_{i,1}]$ the horizontal free space interval that bounds $C_i$ from above. We further denote with $I_{i,0}^v = [c_{i,0}, d_{i,0}]$ the vertical free space interval that bounds $C_i$ from the left and with $I_{i,1}^v = [c_{i,1}, d_{i,1}]$ the vertical free space interval that bounds $C_i$ from the right.

The following lemmata give us bounds on the number of iterations for specific operations of the algorithm. Figure 5.4 shows an example of the relevant free space intervals that are used by the algorithm for checking if $t$ is contained in the structured coverage of some $(i, j) \in \{(i, j) \mid i' - 3 \leq i \leq i' \leq j \leq i + 3\}$.

**Lemma 5.3.4.** *Let $a, b, c \in \mathbb{R}$ with $b \geq 0$. The truth value of $a + \sqrt{b} \leq c$ can be computed by an algorithm that takes $a, b$ and $c$ as input and needs at most 4 simple operations.*

*Proof.* To check if $a + \sqrt{b} \leq c$ we can first check if $a > c$ with one simple operation. If this is the case we can return false. If this is not the case, the statement is equivalent to $b \leq (c - a)^2$. We need one subtraction and one multiplication to calculate $(c - a)^2$. Together with the comparison we get a total of 4 simple operations. □

**Lemma 5.3.5.** *Let $a, b, c, d \in \mathbb{R}$ with $b, d \geq 0$. The truth value of $a + \sqrt{b} \leq c + \sqrt{d}$ can be computed by an algorithm that takes $a, b, c$ and $d$ as input and needs at most 11 simple operations.*

*Proof.* Consider Algorithm 3. It is easy to check that the procedure CheckInequality outputs the truth value of $a + \sqrt{b} \leq c + \sqrt{d}$. We show that the algorithm can be implemented such that it needs at most 11 simple operations. The checks in line 2, 3

**Algorithm 2** Alternative definition of feasibility test

```
1:  procedure IsFeasible(e ∈ X_2^d, t = (t', i') ∈ T_n, (S(t_{i'-3}), S(t_{i'-2}), ..., S(t_{i'+4})) ∈ R^{8d}, Δ ∈ R)
2:      Denote with I^h_{i,0}, I^h_{j,1}, I^v_{i,1}, I^v_{j-1,1} free space intervals of D_Δ(e, S)
3:      J = {(i, j) | i' − 3 ≤ i ≤ i' ≤ j ≤ i + 3}
4:      for (i, j) ∈ J do
5:          if I^h_{i,0} ≠ ∅ and I^h_{j,1} ≠ ∅ then
6:              if a_{i,0} ≤ t and t ≤ b_{j,1} then
7:                  if i = j then
8:                      return true
9:                  if I^v_{i,1} ≠ ∅ then
10:                     if j = i + 1 then
11:                         return true
12:                     if I^v_{j-1,1} ≠ ∅ then
13:                         if c_{i,1} ≤ d_{j-1,1} then
14:                             return true
15:     return false
```

and 11 need one simple operation each. The computation of $z = (c-a)^2 - b - d$ in line 5 or 13 can be done by 3 subtractions and one multiplication. The check for $z \geq 0$ ( respectively $>$) in line 14 ( respectively 6) is one simple operation. The check $z^2 \leq 4bd$ (respectively $<$) in the same line needs 3 multiplications and one comparison. Counting the number of simple operations together, we see that the algorithm needs at most 11 simple operations. □

**Algorithm 3** Check for $a + \sqrt{b} \leq c + \sqrt{d}$

```
1:  procedure CheckInequality(a, b, c, d ∈ R)
2:      if c ≥ a then
3:          if d ≥ b then
4:              return true
5:          Let z = (c − a)^2 − b − d
6:          if z ≥ 0 or z^2 ≤ 4bd then
7:              return true
8:          else
9:              return false
10:     else
11:         if d < b then
12:             return false
13:         Let z = (c − a)^2 − b − d
14:         if z > 0 or z^2 < 4bd then
15:             return false
16:         else
17:             return true
```

**Lemma 5.3.6.** *Let $p, q, r \in \mathbb{R}^d$ and let $\Delta \in \mathbb{R}_+$. The intersection of the line through $p$ and $q$ and the ball with radius $\Delta$ around $r$ is either $\{p + t(q-p) \mid t \in [a - \sqrt{b}, a + \sqrt{b}] \subset \mathbb{R}\}$*

*for some $a, b \in \mathbb{R}$ or is empty. There exists an algorithm that needs at most $10d + 5$ simple operations that takes $p, q, r$ and $\Delta$ as input and outputs the values $a$ and $b$ or decides that the intersection is empty.*

*Proof.* The algorithm needs to solve the quadratic equation

$$\|p + t(q - p) - r\|^2 \leq \Delta^2$$

which is equivalent to $t^2 + \alpha t + \beta = 0$ where

$$\alpha = \frac{\sum_{i=1}^{d} 2(p_i - q_i)r_i}{\sum_{i=1}^{d}(q_i - p_i)} \text{ and } \beta = \frac{-\Delta^2 + \sum_{i=1}^{d}(p_i^2 + r_i^2)}{\sum_{i=1}^{d}(q_i - p_i)}.$$

The sum $\sum_{i=1}^{d}(q_i - p_i)$ can be computed using $2d - 1$ simple operations. For the sum $\sum_{i=1}^{d} 2(p_i - q_i)r_i$ needs $4d - 1$ simple operations and the term $-\Delta^2 + \sum_{i=1}^{d}(p_i^2 + r_i^2)$ needs $4d + 1$ operations. So with the two extra divisions, we need a total of $10d + 1$ operations to compute $\alpha$ and $\beta$. The solutions of the quadratic equation are

$$t_{1,2} = -\frac{\alpha}{2} \pm \sqrt{\left(\frac{\alpha}{2}\right)^2 - \beta}.$$

So we need an extra operation to calculate $a = -\frac{\alpha}{2}$ and an extra two operations to calculate $b = a^2 - \beta$. Also, we have to check if $b < 0$ to decide if the solution is feasible or if the intersection is empty. This sums up to a total of $10d + 5$ □

With the help of the above lemmata, we can get the following result.

**Lemma 5.3.7.** *The algorithm IsFeasible (Algorithm 2) can be implemented such that it needs at most $110d + 412$ simple operations.*

*Proof.* In total, the algorithm needs to compute at most 6 horizontal and 5 vertical free space intervals. Each of them corresponds to the intersection of an edge with a ball. By Lemma 5.3.6, the parameters $a$ and $b$ of the intersection of the line containing this edge with the ball can be computed with an algorithm that needs at most $10d + 5$ simple operations. To get explicit representations of the borders of the free space interval, we have to additionally check if $a + \sqrt{b} \leq 1$, $a - \sqrt{b} \geq 0$ and $a - \sqrt{b} > 1$. By Lemma 5.3.4, each of these checks needs at most 4 simple operations. So we need at most $10d + 17$ operations to compute the parameters $c, d, e, f$ that describe a free space interval $[c - \sqrt{d}, e + \sqrt{f}]$ implicitly. In total, we therefore need $110d + 187$ simple operations for all 11 free space intervals. After computing the intervals, the 4 checks in line 5, 9 and 12 only need one simple operation each. The checks $i = j$ in line 7 and $j = i + 1$ in line 10 are also only one simple operation each. By Lemma 5.3.4, the two checks in line 6 need 4 simple operations each. The remaining check in line 13 needs 11 simple operations, as shown in Lemma 5.3.5. So each iteration of the for loop (line 5-15) needs at most 25 simple operations. Since $J$ has at most 9 elements, the whole for loop needs at most 225 simple operations. By adding the simple operations, that we need to compute the free space intervals, we get that the whole algorithm needs at most $110d + 412$ simple operations. □

Based on Lemma 5.3.7, we can finally bound the VC-dimension.

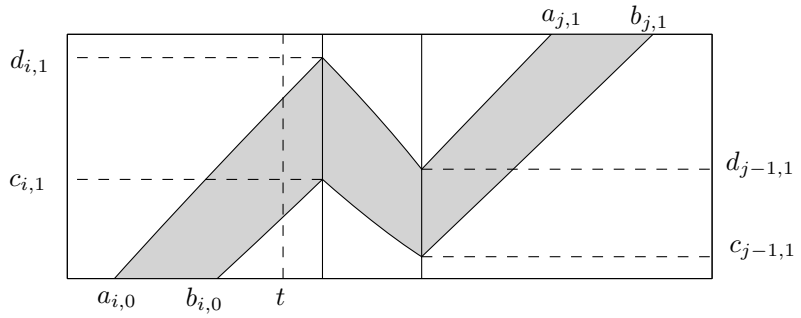


Figure 5.4: Example of the free space intervals $I^h_{i,0} = [a_{i,0}, b_{i,0}]$, $I^h_{j,1} = [a_{j,1}, b_{j,1}]$, $I^v_{i,1} = [c_{i,1}, d_{i,1}]$ and $I^v_{j-1,1} = [c_{j-1,1}, d_{j-1,1}]$ for some $t = (t', i')$ and some $(i, j) \in \{(i, j) \mid i' - 3 \leq i \leq i' \leq j \leq i + 3\}$

**Lemma 5.3.8.** *Let $S : \mathbb{T}_n \to \mathbb{R}^d$ be a polygonal curve and let $\Delta \in \mathbb{R}_+$. The VC-dimension of the range space $\{F_\Delta(t) \mid t \in \mathbb{T}_n\}$ with ground set $\mathbb{X}_2^d$ is at most $3520d^2 + 14128d + 3312$.*

*Proof.* Since $\Delta$ is fixed for the whole range space, each range $F_\Delta(t)$ is uniquely identified by $v_t = (t', i', S(t_{i'-3}), S(t_{i'-2}), \ldots, S(t_{i'+4})) \in \mathbb{R}^{8d+2}$. Applying Theorem 2.4.16 with Lemma 5.3.7 on $\{F_\Delta(t) \mid t \in \mathbb{T}_n\}$ yields a bound of

$$4(8d + 2)(110d + 412 + 2) = 3520d^2 + 14128d + 3312$$

on the VC-dimension. □

### 5.3.4 Improved analysis of the VC-dimension

In this section, we show an improved bound on the VC-dimension of $O(d)$ for the range space $\{F_\Delta(t) \mid t \in \mathbb{T}_n\}$ with ground set $\mathbb{X}_2^d$. We use the same arguments as in Section 4.5.3 that are based on our techniques from Chapter 3. A more detailed bound on the VC-dimension with an analysis of the constant factor can be found in Lemma 5.3.12.

**Lemma 5.3.9.** *Let $S : \mathbb{T}_n \to \mathbb{R}^d$ be a polygonal curve and let $\Delta \in \mathbb{R}_+$. The VC-dimension of the range space $\{F_\Delta(t) \mid t \in \mathbb{T}_n\}$ with ground set $\mathbb{X}_2^d$ is in $O(d)$.*

*Proof.* Let $t = (i', t') \in \mathbb{T}_n$ and $Q \in \mathbb{X}_2^d$. In the following let $t_j = t_1$ for $j < 1$ and $t_j = t_n$ for $j \geq n$. Evaluating if $Q$ is in the feasible set $F_\Delta(t)$ is based on deciding if there exist $i, j$ with $1 \leq j - i \leq 3$ and $a, b$ with $t_i \leq a \leq t \leq b \leq t_j$ such that $d_F(S[a, b], Q) \leq \Delta$. So we only have to consider subcurves of $S[t_{i'-3}, t_{i'+4}]$ when checking if there are subcurves of $S$ containing $t$ that are close to $Q$. Note that the vertices $t_{i'+4}$ and $t_{i'-3}$ are only needed for the cases that $t' = 1$ or $t' = 0$, otherwise the subcurve $S[t_{i'-2}, t_{i'+3}]$ would be sufficient.

We want to show that the range query $Q \in F_\Delta(t)$ can be evaluated using predicates of the types $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_7, \mathcal{P}_8$ for $S[t_{i-3}, t_{i+4}]$ and $Q$ (as in Section 4.5.3). Since we also need predicates for $S(t)$, we define $S'$ to be the polygonal curve with vertices $S(t_{i'-3}), \ldots, S(t_{i'}), S(t), S(t_{i'+1}), \ldots, S(t_{i'+4})$, i.e. $S'$ is the subcurve $S[t_{i'-3}, t_{i'+4}]$ that contains $S(t)$ as an additional vertex between $S(t_{i'})$ and $S(t_{i'+1})$. By applying Lemma 4.5.6 for $S'$, $Q$ and $\Delta$ instead of $10\Delta$, we immediately get that $F_\Delta(t)$ can be determined based on the predicates $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_7, \mathcal{P}_8$ for $S'$ and $Q$. By Lemma 3.4.11 and 3.4.12 the predicates $\mathcal{P}_1, \ldots, \mathcal{P}_8$ are simple. Therefore it follows from Corollary 3.3.2 that the VC-dimension of $(\{F_\Delta(t) \mid t \in \mathbb{T}_n\}, \mathbb{X}_2^d)$ is in $O(d)$.

□

### 5.3.5 Improved detailed analysis of the VC-dimension

In the following, we analyze the constants in our VC-dimension bound of $O(d)$ for the range space $\{F_\Delta(t) \mid t \in \mathbb{T}_n\}$ with ground set $\mathbb{X}_2^d$. Let $t = (i', t') \in \mathbb{T}_n$ and $Q \in \mathbb{X}_2^d$. Let further $S'$ to be the polygonal curve with vertices $S(t_{i'-3}), \dots, S(t_{i'}), S(t), S(t_{i'+1}), \dots, S(t_{i'+4})$ where $t_j = t_1$ for $j < 1$ and $t_j = t_n$ for $j \geq n$. The bound mainly results from an application of Corollary 3.3.2 (as a version of Theorem 2.4.10) with the Predicates $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_7, \mathcal{P}_8$ for $Q \in \mathbb{X}_2^d, \Delta \in \mathbb{R}_+$ and $S' \in \mathbb{X}_9^d$. See Section 3.3.2 for a definition of the predicates. The predicates are functions mapping from $\mathbb{R}^{2d+1} \times \mathbb{R}^{9d}$ to $\{0, 1\}$. To calculate the constant in our VC-dimension bound, we analyze how many predicates of the types $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_7, \mathcal{P}_8$ are required to answer the range query $Q \in F_\Delta(t)$. We further analyze how many sign values of polynomials are required to determine each of the individual predicates and we determine the maximum degree of these polynomials.

**Number of predicates** The curve $S'$ has 8 edges and $Q$ has 2 vertices. Therefore there are 16 predicates of type $\mathcal{P}_1$. For type $\mathcal{P}_2$, we only require the 7 interior vertices of $S'$ together with the edge $Q$. So in total, we have 23 predicates of type $\mathcal{P}_1$ or $\mathcal{P}_2$. For the monotonicity predicates $\mathcal{P}_7, \mathcal{P}_8$, we have no predicate of type $\mathcal{P}_8$ because $Q$ has just one edge. Since we only consider subcurves with at most 3 edges of $S'$, we also only have to consider monotonicity predicates for consecutive interior vertices of $S'$. These are 6 predicates in total of type $\mathcal{P}_7$.

**Lemma 5.3.10.** *For any two polygonal curves $Q \in \mathbb{X}_m^d, S \in \mathbb{X}_k^d$ and a radius $\Delta \in \mathbb{R}_+$, each of the predicates of type $\mathcal{P}_1$ or $\mathcal{P}_2$ is a 6-combination of $\operatorname{sgn}(F)$, where $F$ is a class of functions mapping from $\mathbb{R}^{dm+1} \times \mathbb{R}^{dk}$ to $\mathbb{R}$ so that, for all $x \in \mathbb{X}_m^d$ and $f \in F$ the function $y \to f(x, y)$ is a polynomial on $\mathbb{R}^{dk}$ of degree no more than 4.*

*Proof.* Note that this proof is an in-depth version of the proof of Lemma 3.4.6 that also considers the exact degree and number of the involved polynomials. Let $\mathcal{P}$ be a predicate of type $\mathcal{P}_1$ or $\mathcal{P}_2$. The value of $\mathcal{P}$ can be determined by checking if a vertex $v \in \mathbb{R}^d$ is in the stadium $D_\Delta(\overline{pq})$ for some vertices $p, q \in \mathbb{R}^d$. For this check, it suffices to check if $v$ is in at least one of $B_\Delta(p)$, $B_\Delta(q)$ and $R_\Delta(\overline{pq})$. For $B_\Delta(p)$ ,we have to check the inequality

$$\Delta^2 - \sum_{i=1}^{d} (v_i - p_i)^2 \geq 0$$

which is the sign value of a polynomial of degree 2. The check for $B_\Delta(q)$ is analogous.

To check if $v \in R_\Delta(\overline{pq})$ consider the closest point $s$ to $v$ on the line $\ell(\overline{pq})$. The truth value of

$$\Delta^2 - \|s - v\|^2 \geq 0 \tag{5.1}$$

uniquely determines if $v$ is in the cylinder $C_\Delta(\overline{pq})$. The truth values of

$$\|p - q\|^2 - \|p - s\|^2 \geq 0, \tag{5.2}$$

$$\|p - q\|^2 - \|q - s\|^2 \geq 0 \tag{5.3}$$

further determine if $s$ is on the edge $\overline{pq}$. So the truth values of the inequalities (5.1), (5.2) and (5.3) determine the truth value of $v \in R_\Delta(\overline{pq})$. The closest point to $v$ on the line $\ell(\overline{pq})$ is

$$s = p + \frac{(p - q)\langle (p - q), v \rangle}{\|p - q\|^2}.$$

For each coordinate of $s$, we have

$$s_j = p_j + (p_j - q_j)\frac{\sum_{i=1}^{d}(p_i - q_i)v_i}{\sum_{i=1}^{d}(p_i - q_i)^2}.$$

All three inequalities (5.1), (5.2) and (5.3) are of the type

$$h(p, q, v) - \|x - s\|^2 \geq 0$$

where $x \in \{p, q, v\}$ and $h(p, q, v)$ is a polynomial of degree 2. Let $z \coloneqq \sum_{i=1}^{d}(p_i - q_i)^2$. We have

$$\|x - s\|^2 = \sum_{j=1}^{d}\left((p_j - x_j) + (p_j - q_j)\frac{\sum_{i=1}^{d}(p_i - q_i)v_i}{z}\right)^2.$$

We can check if $z > 0$ with the sign value of one polynomial of degree 2. If this is not the case, then $p = q$ and $R_\Delta(\overline{pq})$ does not need to be considered. Otherwise, we can multiply both sides of inequality 5.3.5 by $z$ and get

$$zh(p, q, v)\sum_{j=1}^{d}\left((p_j - x_j)^2 + 2(p_j - x_j)(p_j - q_j)\sum_{i=1}^{d}(p_i - q_i)v_i\right) + \sum_{j=1}^{d}(p_i - q_i)^2 v_i^2 \geq 0$$

which is the sign value of a polynomial of degree 4. Therefore, we require the sign values of 6 polynomials of degree at most 4 in total. □

**Lemma 5.3.11.** *For any two polygonal curves $Q \in \mathbb{X}_m^d, S \in \mathbb{X}_k^d$ and a radius $\Delta \in \mathbb{R}_+$, each of the predicates of type $\mathcal{P}_7$ is a 9-combination of* $\operatorname{sgn}(F)$*, where $F$ is a class of functions mapping from $\mathbb{R}^{dm+1} \times \mathbb{R}^{dk}$ to $\mathbb{R}$ so that, for all $x \in \mathbb{X}_m^d$ and $f \in F$ the function $y \to f(x, y)$ is a polynomial on $\mathbb{R}^{dk}$ of degree no more than 4.*

*Proof.* Let $\mathcal{P}$ be a predicate of type $\mathcal{P}_7$. The truth value of $\mathcal{P}$ can be determined by checking if there is an intersection of a line segment $\overline{pq}$ with the intersection of two balls $B_\Delta(u)$ and $B_\Delta(v)$ for some vertices $p, q, u, v \in \mathbb{R}^d$. Let $z \coloneqq \sum_{i=1}^{d}(p_i - q_i)^2$. We can check if $z > 0$ with the sign value of one polynomial of degree 2. If this is not the case, then $p = q$ and $\mathcal{P}$ is true if and only if $p \in B_\Delta(u)$ and $p \in B_\Delta(v)$. This can be determined by the two inequalities

$$\Delta^2 - \sum_{i=1}^{d}(u_i - p_i)^2 \geq 0 \quad \text{and}$$
$$\Delta^2 - \sum_{i=1}^{d}(v_i - p_i)^2 \geq 0.$$

Both are sign values of polynomials of degree 2. If $z > 0$ then we precede with analysing the intersection of $\overline{pq}$ with $B_\Delta(u)$ and $B_\Delta(v)$. If the intersection of $\ell(\overline{pq})$ and the $B_\Delta(u)$ exists, the first and the last point of the intersection in direction $(q - p)$ are uniquely defined by

$$s_{1,2} = p + t_{1,2}(u)(q - p)$$

with

$$t_1(u) = -\frac{a}{2} + \sqrt{\frac{a^2}{4} - b}$$

and

$$t_2(u) = -\frac{a}{2} - \sqrt{\frac{a^2}{4} - b}$$

where

$$a = \frac{2\sum_{i=1}^{d}(p_i - u_i)(q_i - p_i)}{\sum_{i=1}^{d}(q_i - p_i)^2} \quad \text{and} \quad b = \frac{\sum_{i=1}^{d}(p_i - u_i)^2 - \Delta^2}{\sum_{i=1}^{d}(q_i - p_i)^2}.$$

Analogously, we have for $B_\Delta(v)$ the values

$$t_1(v) = -\frac{c}{2} + \sqrt{\frac{c^2}{4} - d}$$

and

$$t_2(v) = -\frac{c}{2} - \sqrt{\frac{c^2}{4} - d}$$

where

$$c = \frac{2\sum_{i=1}^{d}(p_i - v_i)(q_i - p_i)}{\sum_{i=1}^{d}(q_i - p_i)^2} \quad \text{and} \quad d = \frac{\sum_{i=1}^{d}(p_i - v_i)^2 - \Delta^2}{\sum_{i=1}^{d}(q_i - p_i)^2}.$$

To check if the intersections exist, we have to check if $\frac{a^2}{4} - b \geq 0$ and $\frac{c^2}{4} - d \geq 0$. We can multiply both sides of the inequalities by $z^2$ and get sign values of polynomials of degree 4. If the intersections exist, we can determine if they overlap by checking the 4 comparisons $t_1(u) \leq t_1(v)$, $t_1(u) \leq t_2(v)$, $t_2(u) \leq t_1(v)$ and $t_2(u) \leq t_2(v)$. We analyze the comparison $t_1(u) \leq t_1(v)$. The other comparisons are analogous. The comparison is equivalent to

$$\frac{c}{2} - \frac{a}{2} \leq \sqrt{\frac{a^2}{4} - b} - \sqrt{\frac{c^2}{4} - d}.$$

To check the truth of this inequality, we first check the sign values of each individual side. The left side is the sign value of a polynomial of degree 2 after multiplying with $z$. Note that the left side does not change for the other comparisons $t_1(u) \leq t_2(v)$, $t_2(u) \leq t_1(v)$ and $t_2(u) \leq t_2(v)$. For the right side, we have to check if $\frac{a^2}{4} - b \geq \frac{c^2}{4} - d$ which is a polynomial of degree 4 after multiplying with $z^2$. The check for this right side is the same for $t_2(u) \leq t_2(v)$ and the sign is trivially given for $t_1(u) \leq t_2(v)$ and $t_2(u) \leq t_1(v)$. If the signs of the left and the right side differ, then the truth of the inequality can already be determined by these signs. Otherwise, we square both sides and get

$$b + d - \frac{ac}{2} \leq -2\sqrt{\frac{a^2}{4} - b}\sqrt{\frac{c^2}{4} - d}.$$

We check the sign of $b + d - \frac{ac}{2}$ which is a polynomial of degree 4 after multiplying with $z^2$. This is again the same check for all comparisons. Depending on the sign, the truth of the whole inequality is either directly given or we can square both sides again to get

$$b^2 + d^2 - abc - acd + 2bd \leq 4db - a^2d - bc^2.$$

This is an sign value of a polynomial of degree 4 with respect to the coordinates of $u$ and $v$. In total we have 9 sign values of polynomials with a degree of at most 4 with respect to the coordinates of $u$ and $v$. □

With the help of Lemma 5.3.10 and Lemma 5.3.11 and a careful analysis of reoccurring sign values of polynomials, we can achieve the following bound on the VC-dimension.

**Lemma 5.3.12.** *Let $S : \mathbb{T}_n \to \mathbb{R}^d$ be a polygonal curve and let $\Delta \in \mathbb{R}_+$. The VC-dimension of the range space $\{F_\Delta(t) \mid t \in \mathbb{T}_n\}$ with ground set $\mathbb{X}_2^d$ is at most $229d$.*

*Proof.* Let $t = (i', t') \in \mathbb{T}_n$ and $Q \in \mathbb{X}_2^d$. Let further $S'$ to be the polygonal curve with vertices $S(t_{i'-3}), \dots, S(t_{i'}), S(t), S(t_{i'+1}), \dots, S(t_{i'+4})$ where $t_j = t_1$ for $j < 1$ and $t_j = t_n$ for $j \geq n$. By applying Lemma 4.5.6 for $S'$, $Q$ and $\Delta$ instead of $10\Delta$, we immediately get that $F_\Delta(t)$ can be determined based on the predicates $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_7, \mathcal{P}_8$ for $S'$ and $Q$. As argued at the beginning of this section, there are 23 relevant predicates of type $\mathcal{P}_1$ or $\mathcal{P}_2$, 7 relevant predicates of type $\mathcal{P}_7$ and no relevant predicates of type $\mathcal{P}_8$. The predicates are functions mapping from $\mathbb{R}^{2d+1} \times \mathbb{R}^{9d}$ to $\{0, 1\}$. By Lemma 5.3.10 and Lemma 5.3.11, the range space $(\{F_\Delta(t) \mid t \in \mathbb{T}_n\}, \mathbb{X}_2^d)$ is therefore a 201-combination of $\operatorname{sgn}(F)$, where $F$ is a class of functions mapping from $\mathbb{R}^{dm+1} \times \mathbb{R}^{dk}$ to $\mathbb{R}$ so that, for all $x \in \mathbb{X}_m^d$ and $f \in F$ the function $y \to f(x, y)$ is a polynomial on $\mathbb{R}^{dk}$ of degree no more than 4. In the following, we show that it is even a 139-combination by careful identification of repeated sign values of polynomials. First of all, note that the checks for $v \in B_\Delta(p)$ and $p \in B_\Delta(v)$ are the same for any pair of vertices $v, p \in \mathbb{R}^d$ and are answered by

$$\Delta^2 - \sum_{i=1}^{d} (v_i - p_i)^2 \geq 0.$$

We have 9 vertices of $S'$ and 2 vertices of $Q$ for a total of 18 combinations to check. However, the predicates contain many repetitions of these checks. Each predicate of type $\mathcal{P}_1, \mathcal{P}_2$ or $\mathcal{P}_7$ contains 2 such checks, resulting in 60 checks in total. This is already a reduction of 42 sign values from 201 to 159. Let $Q = \overline{pq}$. We check if $p = q$ by checking $\sum_{i=1}^{d} (p_i - q_i)^2 > 0$ in all predicates of type $\mathcal{P}_2$ and $\mathcal{P}_7$. Since 1 check instead of 13 suffices, we can reduce the 159 sign values further to 147. Similarly, each check $\sum_{i=1}^{d} (x_i - y_i)^2 > 0$ where $x$ and $y$ are 2 consecutive vertices of $S'$ is contained in 2 predicates of type $\mathcal{P}_1$. This reduces the number of sign values further to 139. Note that these checks also tell us, if the vertex $S(t)$ is the same as $S(t_{i'})$ or $S(t_{i'+1})$. This information determines which subcurves of $S'$ that have at most 3 edges contain $t$ and are relevant for a structured covering. In total the range space $(\{F_\Delta(t) \mid t \in \mathbb{T}_n\}, \mathbb{X}_2^d)$ is a 139-combination of $\operatorname{sgn}(F)$, where $F$ is a class of functions mapping from $\mathbb{R}^{dm+1} \times \mathbb{R}^{dk}$ to $\mathbb{R}$. Applying Theorem 2.4.10 yields a bound of

$$2 \cdot 9d \cdot \log_2(12 \cdot 139 \cdot 4) \leq 229d$$

on the VC-dimension. □

## 5.4 The main algorithm

As our main algorithm, we apply the multiplicative weight update method for hitting sets described in Section 2.4.2 on $(Z_{\Delta, E(S)}, \{F_{9\Delta}(t) \mid t \in \mathbb{T}_n\})$. In this chapter, we discuss several specifications of our implementation that enable us to apply the framework in the form of Theorem 2.4.9 to our use case. In Section 5.4.1, we describe how to compute a $\Delta$-good simplification $S$ of $P$. In Sections 5.4.2 and 5.4.3, we specify how the data structures for the verifier and for sampling are implemented. In Section 5.4.4, we combine everything to derive our main result.

### 5.4.1 Simplification algorithm

In this section, we describe an algorithm to construct a $\Delta$-good simplification $S$ for a given polygonal curve $P$. Our simplification algorithm utilizes a data structure that is built on the input curve and which allows querying the Fréchet distance of a subcurve to an edge (up to some small approximation factor). For this, we use the following result by Driemel and Har-Peled [55].

**Theorem 5.4.1** (Theorem 5.9 in [55])**.** *Given a polygonal curve $Z$ with $n$ vertices in $\mathbb{R}^d$, one can build a data structure, in $O(\chi^2 n \log^2 n)$ time, that uses $O(n\chi^2)$ space, such that for a query edge $\overline{p\,q}$, and any two points $u$ and $v$ on the curve, one can $(1+\varepsilon)$-approximate the distance $d_F(Z[u,v], \overline{p\,q})$ in $O(\varepsilon^{-2} \log n \log\log n)$ time, and $\chi = \varepsilon^{-d} \log(\varepsilon^{-1})$.*

**Algorithm 4** Curve simplification

```
1:  procedure SimplifyCurve(Polygonal curve P in R^d, Δ > 0)
2:      Build data structure D of Theorem 5.4.1 on P with ε = 1/3.
3:      Let Ŝ be an empty stack.
4:      Ŝ.push(1)
5:      for 2 ≤ i ≤ n do
6:          j ← Ŝ.next_to_top()
7:          while j is defined and D.query(P[t_j, t_i], p_j p_i) ≤ (8/3)Δ do
8:              Ŝ.pop()
9:              j ← Ŝ.next_to_top()
10:         j ← Ŝ.top()
11:         if ‖P(t_j) − P(t_i)‖ ≥ Δ/3 then
12:             Ŝ.push(i)
13:     return the simplification S defined by the indices in Ŝ
```

**Theorem 5.4.2.** *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 $\Delta > 0$ be given. There exists an algorithm that outputs an index set defining a $\Delta$-good simplification of $P$. Furthermore, it does so in $O(n \log^2 n)$ time and $O(n)$ space.*

*Proof.* Consider Algorithm 4. We want to show, that the simplification $S$ of $P$ defined by the index set $\hat{S}$ is $\Delta$-good. For this we have to show, that $S$ fulfills properties $(i) - (iv)$ of Definition 5.2.1. Denote by $s$ the last item of $\hat{S}$, which is updated whenever $\hat{S}$ changes. Note that property $(i)$ follows immediately, as otherwise, the index $i_{j+1}$ would not have been added to $I$ in line 12.

For property $(ii)$ we will show the following invariance. Whenever we start some generic iteration of the loop in line 5, where we try to add $i$ to the index set, then $P[t_s, t_{i-1}] \subset \mathrm{b}_{3\Delta}(p_s)$. 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. After exiting the loop in line 7, we either updated $\hat{S}$, or we did not. In any case, we will assume, that $\|p_s - p_i\| < \Delta/3$, because otherwise, we add $i$ to $\hat{S}$ in this iteration. But then at the start of the next iteration, in which we consider adding $i+1$, we then have that $s = i$, for which the invariance trivially holds, as $P[t_s, t_{(i+1)-1}] = P[t_s, t_s] = p_s$.

Assume for now, that we did not update $\hat{S}$, that is we never entered line 8. Then $P[t_s, t_i] \subset \mathrm{b}_{3\Delta}(p_s)$, as $P[t_s, t_{i-1}]$, $p_{i-1}$ and $p_i$ lie inside $\mathrm{b}_{3\Delta}(p_s)$.

Otherwise, $d_F(P[t_s, t_i], \overline{p_s\, p_i}) \leq \textsc{query}(P[t_s, t_i], \overline{p_s\, p_i}) \leq 8\Delta/3$, implying together with our assumption $\|p_s - p_i\| < \Delta/3$, that $P[t_s, t_i] \subset \mathrm{b}_{3\Delta}(p_s)$, hence the invariance holds in every iteration.

This invariance implies, that whenever we start iteration $i$ of the loop, we have that

$$\begin{aligned} d_F(P[t_s, t_i], \overline{p_s\, p_i}) &\leq \max\left(d_F(P[t_s, t_{i-1}], \overline{p_s, p_s}), d_F(P[t_{i-1}, t_i], \overline{p_s, p_i})\right) \\ &\leq \max\left(3\Delta, d_F(P[t_{i-1}, t_i], \overline{p_s, p_i})\right) \\ &\leq \max\left(3\Delta, \Delta/3\right) = 3\Delta, \end{aligned}$$

where the second inequality follows form the invariance, as $\overline{p_s\, p_s} = p_s$, and the third from the fact, that $P[t_{i-1}, t_i]$ is an edge from $p_{i-1}$ to $p_i$, and we know, that $\|p_s - p_{i-1}\| \leq \Delta/3$. Thus at the start of each iteration $i$ we have that

$$d_F(P[t_s, t_i], \overline{p_s\, p_i}) \leq 3\Delta \qquad (*)$$

holds for $i$ and $s$. As we continuously remove the last item from $\hat{S}$, we only do so, if (*) holds for $\hat{S}$.next_to_top() and $i$. As we remove $s$, $\hat{S}$.next_to_top() takes the place of $s$, thus at every iteration of the loop in line 7, as well as when we exit the loop, (*) holds. Thus every time we possibly add any index in line 12, (*) holds, and thus property $(ii)$ is always maintained.

Property $(iii)$ follows directly for $i_1 = 1$. As $i_k$ may be less than $n$, the property follows from the invariance, as $\overline{p_{i_k}\, p_{i_k}} = p_{i_k}$. And thus $P[t_{i_k}, t_n] \subset \mathrm{b}_{3\Delta}(p_{i_k})$.

For property $(iv)$ observe that, if $d_F(P[t_{i_j}, t_{i_{j+2}}], \overline{p_{i_j}\, p_{i_{j+2}}}) < 2\Delta$ 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 7, as when we add $i_{j+2}$, $\hat{S}$.next_to_top() $= i_j$, and hence

$$\textsc{query}(P[t_{i_j}, t_{i_{j+2}}], \overline{p_{i_j}\, p_{i_{j+2}}}) \leq (4/3) d_F(P[t_{i_j}, t_{i_{j+2}}], \overline{p_{i_j}\, p_{i_{j+2}}}) < 8/3\Delta.$$

The space is dominated by the space for storing the data structure. For analysing 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 preprocessing time and the at most $O(n)$ queries to the data structure. Thus the iterations of the loop in line 7 are bounded by $O(n)$ overall. Thus the claim follows from Theorem 5.4.1. □

### 5.4.2 The verifier

We will now discuss how to implement the verifier. The verifier needs to decide for a query set $C \subseteq Z_{\Delta, E(S)}$ if $C$ is a hitting set and return a set $F_{9\Delta}(t)$ of $\mathcal{R} = \{F_{9\Delta}(z) \mid z \in \mathbb{T}_n\}$ such that $F_{9\Delta}(t) \cap C = \emptyset$ if it is not. Our implementation of the verifier will return the set $F_{9\Delta}(t)$ implicitly by returning the parameter $t$. To find a fitting $t$, the verifier will explicitly compute the structured $9\Delta$-coverage of $C$.

We will now discuss how to compute the structured $\Delta$-coverage of a fixed solution and how to test for given $i, j, t$ and $\Delta$, whether $t$ is in $\Psi_\Delta^{(i,j)}$. For technical reasons, we need to be able to deduce the index of the edge of the curve $P$ that contains a point $P(t)$ in constant time from the parameter $t \in [0, 1]$. To this end, we introduce the following more structured edge-parametrization.

**Definition 5.4.3.** *Let $P : [0,1] \to \mathbb{R}^d$. Let $(t_1, \ldots, t_n)$ be the vertex-parameters of $P$. Define the **edge-parametrization** $\eta_P : \mathbb{T}_n \to [0,1]$ via $\eta(i,t) = (1-t)t_i + tt_{i+1}$. This induces a function $P \circ \eta_P : \mathbb{T}_n \to [0,1]$.*

The cell $C_{i,j}$ in the $\Delta$-free space of $P$ and $Q$ corresponding to the $i$th edge of $P$ and $j$th edge of $Q$ is then defined as $C_{i,j} = [\eta_P(i,0), \eta_P(i,1)] \times [\eta_Q(j,0), \eta_Q(j,1)]$.

**Lemma 5.4.4.** *Let $P$ be a polygonal curve in $\mathbb{R}^d$, and let $(t_1, \ldots, t_n)$ be the vertex-parameters of $P$. Let $Q$ be a point in $\mathcal{Z}_E$ for some edge set $E$ and let $\Delta \geq 0$ be a real value. Assume that $P$ is given as a pointer to an array storing the sequence of vertices. Given any integer values $1 \leq i < j \leq n$, real value $t \in [0,1]$, there exists an algorithm that decides if $t \in \Psi_\Delta^{(i,j)}(P,Q)$, in $O(|j-i|)$ time.*

*Proof.* Note that the $\Delta$-free space diagram of $P$ and the edge $Q$ consists of a single row of free space cells and each free space cell can be computed locally from the two corresponding edges. We need to check if there exists a point $(b,1)$ on the top boundary with $b \in [\max(t, t_{j-1}), t_j]$ that is reachable by a bi-monotone path in the $\Delta$-free space which starts in a point $(a,0)$ with $a \in [t_i, \min(t, t_{i+1})]$ on the bottom boundary of the free space diagram. Using the technique by Alt and Godau [13] we can process the subset of the free space diagram that corresponds to $P[t_i, t_j]$ from left to right to check if there exists such a path. If there exists such a path, then we return "yes", otherwise we return "no". □

**Lemma 5.4.5.** *Given a polygonal curve $P \in \mathbb{X}_n^d$, a set $C \subset \mathbb{X}_2^d$ with $|C| = k$ and a real value $\Delta \geq 0$, there exists an algorithm that computes the structured $\Delta$-coverage $\Psi'_\Delta(P,C)$ in $O(nk\log(k))$ time and $O(nk)$ space.*

*Proof.* Let $J = \{1 \leq i \leq j \leq n \mid 0 \leq j - i \leq 3\}$. Fix an element $q \in C$ and an element $(i,j) \in J$. We have to compute

$$R(q,i,j) = \bigcup_{a,b \in [0,1]} \{s \in [t,t'] \mid t = \eta_P(a,i)), t' = \eta_P(b,j), d_F(P[t,t'], q) \leq \Delta, t \leq t'\}.$$

Consider the free space intervals $I_{i,0}^h = [a_{i,0}, b_{i,0}]$, $I_{j,1}^h = [a_{j,1}, b_{j,1}]$, $I_{i,1}^v = [c_{i,1}, d_{i,1}]$ and $I_{j-1,1}^v = [c_{j-1,1}, d_{j-1,1}]$ of the $\Delta$-free space $\mathcal{D}_\Delta(P,q)$. If any of the Intervals is empty or $c_{i,1} \leq d_{j-1,1}$, then we have $R(q,i,j) = \emptyset$, since there is no bi-monotone path in the free space from $I_{i,0}^h$ to $I_{j,1}^h$. Otherwise, we have $R(q,i,j) = [a_{i,0}, b_{j,1}]$. Each free space interval corresponds to the intersection of a line with a ball and can be computed in constant time. So in total, also $R(q,i,j)$ is an interval that can be computed in $O(1)$ time. All $R(q,i,j)$ can therefore be computed in $O(|C||J|)$ time and need $O(|C||J|)$ space.

Given $R(q,i,j)$ for all $q \in C$ and $(i,j) \in J$, we can then compute the structured $\Delta$-coverage $\Psi'_\Delta(P,C)$ with a standard scan algorithm over the computed intervals in $O(|C||J|\log(|C|))$ time. For the derivation of this bound on the running time, note that the number of overlapping intervals at any point of the scan is bounded by $O(|C|)$, since any point on the $i'$-th edge of $P$ can only be covered by an interval $R(q,i,j)$ with $i \leq i' \leq j$. The theorem statement follows by $|C| = k$ and $|J| = O(n)$. □

After computing the structured $9\Delta$-coverage $\Psi'_{9\Delta}(S,C)$, the first uncovered $t$ can be output in constant time.

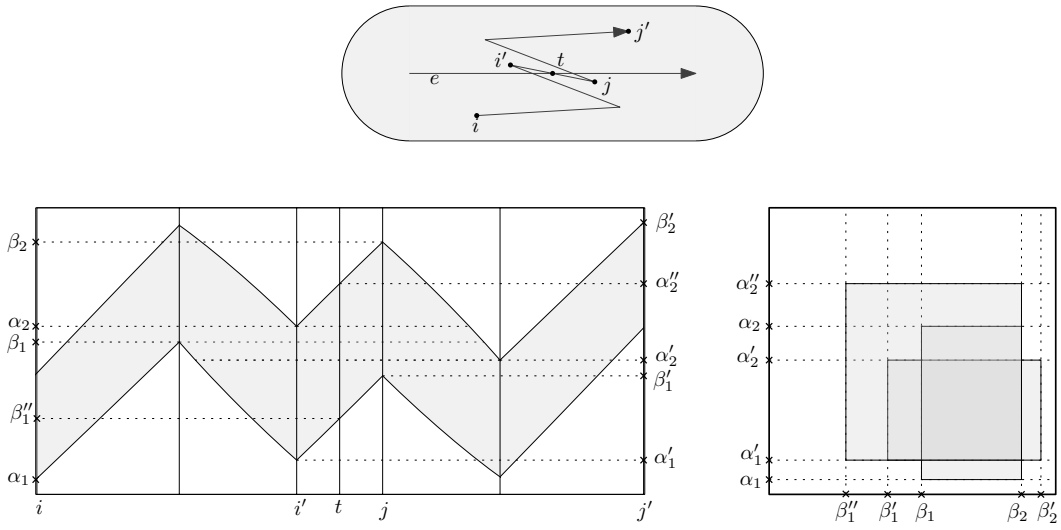


Figure 5.5: Structure of a feasible set. The image shows an example of the structure of the feasible set $F_{9\Delta}(t)$ restricted to the subedges of some edge $e$ of some curve $S$. The top image displays the edge $e$ and the subcurve $S[t_i, t_{j'}]$ that includes all subcurves of $S$ with complexity 4 that contain $t$. The left image shows the $9\Delta$-free space diagram of $e$ and $S[t_i, t_{j'}]$ with notations of the relevant parameters for computing the corresponding rectangles in the parameter space of $e$ based on Lemma 5.3.2. The right image shows the feasible set as a union of these rectangles. The rectangle $[\alpha_1, \alpha_2] \times [\beta_1, \beta_2]$ corresponds to the tuple $(i, j)$, the rectangle $[\alpha'_1, \alpha'_2] \times [\beta'_1, \beta'_2]$ to $(i', j')$ and $[\alpha'_1, \alpha''_2] \times [\beta''_1, \beta_2]$ to $(i', j)$. All other tuples correspond to rectangles that are contained in these three rectangles.

### 5.4.3 Data structure for sampling

We describe a simple static data structure to store the discrete probability distribution $\mathcal{D}_i$ over the finite set $Z_{\Delta,E(S)}$, which we use in our adaptation of the multiplicative weight update method. The data structure takes as input a polygonal curve $S \in \mathbb{X}_n^d$, and a sequence of values $t_1, \ldots, t_i \in [0, 1]$, which are used for the weight update. The data structure should support the following operations.

(1) Sample and return an (explicit) element of $Z_{\Delta,E(S)}$ according to $\mathcal{D}_i$.

(2) Given a query point $t$, determine if $\Pr_{\mathcal{D}_i}[F_{9\Delta}(t)]$ is at most $\varepsilon$.

In the following, we use the fact that the feasible set restricted to this candidate set has a nice structure that can be stored implicitly and can be computed fast. In particular, Lemma 5.3.2 states that the feasible set, when restricted to the subedges of an edge $e \in E(S)$, can be written as the union of constantly many rectangles. This structure is illustrated in Figure 5.5.

**The data structure** For each edge $e$ of $E(S)$, we will store an arrangement $\mathcal{A}_e$ of the horizontal and vertical lines that delineate the boundaries of the rectangles that form the sets $F_{9\Delta}(t_1), \ldots F_{9\Delta}(t_i)$ when restricted to the parameter space of subedges of $e$. For each cell $C$ of this arrangement, we store the following information (refer to Figure 5.6 for an illustration):

(i) $g_C$, the number of grid points in the cell $|C \cap Z_{\Delta,e}|$

(ii) $s_C$, the number of feasible sets $F_{9\Delta}(t_1), \dots F_{9\Delta}(t_i)$ that contain $C$

The weight function $w : Z_{\Delta,E(S)} \to \mathbb{R}_+$ that defines the distribution $\mathcal{D}_i$ can then be evaluated on the cell $C$ as

$$w(C) = 2^{s_C} g_C$$

In particular, this weight function defines the probability distribution $\mathcal{D}_i$ in the following sense. The probability of a grid point $Q \in Z_{\Delta,E(S)} \cap C$ is given by

$$\Pr[Q] = 2^{s_C} / w(Z_{\Delta,E(S)})$$

For computing the set of cells, for each edge $e \in E(S)$, we build the arrangement $\mathcal{A}_e$ by first collecting the coordinates of the horizontal and vertical lines that delineate the feasible sets and then sorting them by $x$ (resp. $y$)-coordinate. Since the arrangement is a (non-uniform) grid, the information for all cells can be stored in an array with appropriate indexing. Initially, for $i = 0$, we only have one cell $C$ for each edge $e_j \in E(S)$, which is the unit square, and we set $s_C = 0$, and $g_C = |X_j|$ (see Definition 5.2.9). For $i > 0$, we compute the number of feasible sets $s_C$ using dynamic programming, by scanning over the arrangement of cells in a column by column fashion. Computing the number of grid points $g_C$ can be done by scanning over the arrangement in a similar way. Here, we compute the number of gridpoints in the interval between two horizontal or vertical lines by using a binary search on the set of gridpoints $X_j$. Clearly, computing the arrangement $\mathcal{A}_e$ and the values $s_C$ and $g_C$ for each cell can be done in $O(i^2 \log(|X_j|))$ time and space per edge $e_j \in E(S)$.

Let $\mathcal{M} = \{C_1, \dots, C_m\}$ denote the union of the set of cells over all arrangements of the edges of $E(S)$, using an appropriate indexing (i.e., lexicographical ordering in the horizontal, and vertical direction, and in the index of the edge $e$ of $E(S)$). In addition to the values $s_C$ and $g_C$, we store the cumulative function $f : \mathcal{M} \to \mathbb{R}_+$, which is simply defined as $f(C_j) = \sum_{i=1}^{j} w(C_i)$ and can be computed by scanning over all cells in the order of their index. For consistency, we define $f(C_0) = 0$. Note that the total weight $w(Z_{\Delta,E(S)})$ is now stored in $f(C_m)$. The function $f$ can be computed in $O(m)$ time and space and this also bounds the total space used by the data structure.

**Sampling from the distribution** Using the cumulative function $f$ defined on the cells of the arrangements and the additional information for each cell, we can sample from $\mathcal{D}_i$ as follows. Draw a sample $x$ uniformly at random from the interval $[0, w(Z_{\Delta,E(S)})]$. Perform a binary search on the function values of the cumulative function for $x$, let cell $C_j$ be the result of the binary search. Let $j' = \lceil (x - f(C_{j-1}))/2^{s_{C_j}} \rceil$ and return the $j'$-th grid point (according to a lexicographical ordering) that lies in the cell $C_j$. Clearly, this can be done in time in $O(\log m + \log(\lambda/\Delta))$, where $\lambda$ denotes the length of the longest edge of $S$.

**Evaluating the weight of a feasible set** With the data structure as described above, we can evaluate $\Pr_{\mathcal{D}_i}[F_{9\Delta}(t)]$ as follows. For each edge $e \in E(S)$, we find the set of cells intersected fully or partially by the feasible set $F_{9\Delta}(t)$ by scanning over all cells associated with the edge $e$. If a cell is intersected only partially, we can determine the number of grid points that lie in the intersection by using a constant number of binary

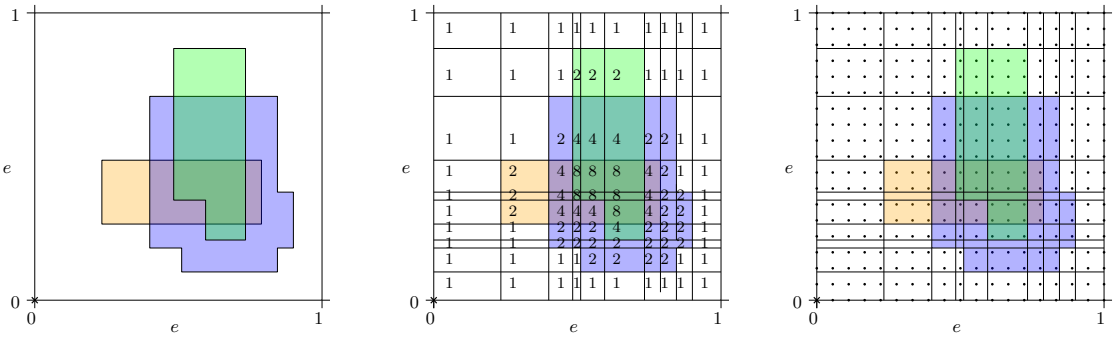


Figure 5.6: Left: Example of an arrangement of lines stored in the data structure for one edge $e \in S$. The left image shows the arrangement of the feasible sets that were used for the updates. Center: The actual arrangement of lines that is stored in the data structure. The number shown in each cell is the corresponding weight multiplicator resulting from doubling the weights in each update where the cell is included in the feasible set. Right: The grid of candidates in $Z_{\Delta,E(S)}$ that are subcurves of $e$. The probability to draw a candidate is proportional to the weight multiplicator of the cell that contains it.

searches, since $F_{9\Delta}(t)$ is the constant union of a set of rectangles when restricted to the parameter space of the edge $e$. From this, we can compute the weight of the feasible set by summing over all intersected cells. Dividing this weight by the total weight $w(Z_{\Delta,E(S)})$ yields the probability $\Pr_{\mathcal{D}_i}[F_{9\Delta}(t)]$. Clearly, the total time for evaluating the weight of one feasible set is in $O(m \log(\lambda/\Delta))$. (Better running times are possible by storing the cumulative function in a more structured way, but this does not affect our total running time.)

We conclude the section with a theorem summarizing what we have derived.

**Theorem 5.4.6.** *Given a polygonal curve $S \in \mathbb{X}_n^d$, and a sequence of values $t_1, \ldots, t_i \in [0,1]$, we can build a data structure that supports the following operations:*
*(1) Sample and return an explicit element of $Z_{\Delta,E(S)}$ according to $\mathcal{D}_i$.*
*(2) Given a query point $t$, determine if $\Pr_{\mathcal{D}_i}[F_{9\Delta}(t)]$ is at most $\varepsilon$.*
*Let $m = n \cdot i^2$, and let $\lambda$ denote the length of the longest edge of $S$. The query time for (1) is in $O(\log(m) + \log(\lambda/\Delta))$. The query time for (2) is in $O(m \log(\lambda/\Delta))$. The data structure can be built in $O(m \log(\lambda/\Delta))$ time and uses space in $O(m)$.*

Note that the weight update and weight reset step can be realized by building the data structure from the ground in $O(m \log(\lambda/\Delta))$ which is the same time as the query time for (2).

### 5.4.4 Result for implicit weight update

By using the verifier of Section 5.4.2 and using the data structure of Section 5.4.3 for maintaining the discrete probability distribution on the implicit candidate set $Z_{\Delta,E(S)}$, we get the following result for applying the multiplicative weight update method (Theorem 2.4.9) on $(X, \mathcal{R}) = (Z_{\Delta,E(S)}, \{F_{9\Delta}(t) \mid t \in \mathbb{T}_n\})$.

**Theorem 5.4.7.** *Let $S : \mathbb{T}_n \to \mathbb{R}^d$ be a polygonal curve and let $\Delta \in \mathbb{R}_+$. Assume for the range space $(Z_{\Delta,E(S)}, \{F_{9\Delta}(t) \mid t \in \mathbb{T}_n\})$ exists a hitting set of size $k$. Then, there exists*

*an algorithm that computes a hitting set of size $k' \in O(k \log(k))$ with expected running time in $O\left(nk^3 \log^4\left(\frac{n\lambda(P)}{\Delta k}\right)\right)$ and using space in $O\left(nk^2 \log^2\left(\frac{n\lambda(P)}{\Delta k}\right)\right)$.*

*Proof.* We apply Theorem 2.4.9 on the range space $(X, \mathcal{R}) = (Z_{\Delta,E(S)}, \{F_{9\Delta}(t) \mid t \in \mathbb{T}_n\})$. By Lemma 5.3.9, the VC-dimension of $(X, \mathcal{R})$ is in $O(d)$ which we assumed in the Problem definition (Section 2.5) to be constant. Furthermore, by Lemma 5.4.5, the verifier has a query time $T_V$ in $O(nk' \log(k'))$ and space requirement $S_V$ in $O(nk')$ and does not need any preprocessing.

Let $m = n \cdot i_{max}^2$ where $i_{max}$ is the maximum number of weight updates that can happen in a single run of k-MWU-Algorithm. Theorem 5.4.6 implies the following for the data structure that maintains the discrete probability distribution on the implicit candidate set $Z_{\Delta,E(S)}$: The query time $T_{\mathcal{D}}$ is in $O(\log(m) + \log(\lambda/\Delta) + m \log(\lambda/\Delta))$ and the space requirement $S_{\mathcal{D}}$, update time $U_{\mathcal{D}}$ and preprocessing time $P_{\mathcal{D}}$ are in $O(m \log(\lambda/\Delta))$. As shown in the proof of Theorem 2.4.9, the algorithm k-MWU-Algorithm always terminates if $|H| \geq k$. Therefore, we have

$$i_{max} \leq 8k \log_2\left(\frac{|Z_{\Delta,E(S)}|}{k}\right).$$

It remains to bound $|Z_{\Delta,E(S)}|$. For each edge, there are at most $\lambda(P)/\Delta + 2$ start points and $\lambda(P)/\Delta + 2$ end points that together define a candidate in $Z_{\Delta,E(S)}$. Therefore, the value $|Z_{\Delta,E(S)}|$ is in $O\left(n\frac{\lambda(P)^2}{\Delta^2}\right)$. Inserting all derived bounds in Theorem 2.4.9 yields the statement of the theorem. □

We can apply Theorem 5.4.7 on a $\Delta$-good simplification $S$ of a polygonal curve $P$. The simplification $S$ can be computed in $O(n \log^2(n))$ time by Theorem 5.4.2. Since a hitting set of $\{F_{9\Delta}(t) \mid t \in \mathbb{T}_n\}$ is a structured $9\Delta$-covering of $S$, we immediately get the following main result from Corollary 5.2.10.

**Theorem 5.4.8.** *Let $P \in \mathbb{X}_n^d$ and $\Delta \in \mathbb{R}_+$. Let $k$ be the minimum size of a solution to the $(\Delta, 2)$-covering problem on $P$. Let further $\lambda(P)$ be the arc length of the curve $P$. There exists an algorithm that outputs a $(12, O(\log(k))$-approximate solution. The algorithm needs in expectation $O\left(nk^3 \log^4\left(\frac{n\lambda(P)}{\Delta k}\right) + n \log^2(n)\right)$ time and $O\left(nk^2 \log^2\left(\frac{n\lambda(P)}{\Delta k}\right)\right)$ space.*

For general $\ell$, we get based on Lemma 5.1.1 the following result

**Theorem 5.4.9.** *Let $P \in \mathbb{X}_n^d$ and $\Delta \in \mathbb{R}_+$. Let $k$ be the minimum size of a solution to the $(\Delta, \ell)$-covering problem on $P$. Let further $\lambda(P)$ be the arc length of the curve $P$. There exists an algorithm that outputs a $(12, O(\ell \log(k\ell))$-approximate solution. The algorithm needs in expectation $O\left(n(k\ell)^3 \log^4\left(\frac{n\lambda(P)}{\Delta k\ell}\right) + n \log^2(n)\right)$ time and $O\left(n(k\ell)^2 \log^2\left(\frac{n\lambda(P)}{\Delta k\ell}\right)\right)$ space.*

# Chapter 6

# On the number of iterations of the DBA algorithm

The main content of this chapter previously appeared as the paper *On the number of iterations of the DBA algorithm* [30] by Frederik Brüning, Anne Driemel, Alperen Ergür and Heiko Röglin which was published in the *Proceedings of the 2024 SIAM International Conference on Data Mining (SDM)*. A full version of the paper is available on arXiv [31]. An initial version of the work has also been presented at the *39th European Workshop on Computational Geometry (EuroCG 2023)* [29] based on an extended abstract without formal publication.

## 6.1 Introduction

In this chapter, we analyze the number of iterations that DBA performs until convergence. The DTW Barycenter Averaging (DBA) algorithm is a widely used algorithm for estimating the mean of a given set of point sequences. We assume the algorithm is given $n$ sequences of $m$ points in $\mathbb{R}^d$ and a parameter $k$ that specifies the length of the mean sequence to be computed. In this context, the mean is defined as a point sequence that minimizes the sum of dynamic time warping distances (DTW). The algorithm is similar to the $k$-means algorithm in the sense that it alternately repeats two steps: (1) computing an optimal assignment to the points of the current mean, and (2) computing an optimal mean under the current assignment. We follow a line of thought that has proved successful for the closely related $k$-means algorithm by Lloyd [98]. For this algorithm it is known that the number of iterations in the worst-case is exponential [17, 75, 84, 130]. However, on most practical instances the $k$-means algorithm is reported to converge very fast. Moreover, using smoothed analysis, it has been shown that the expected running time under random perturbations of the input is merely polynomial [16]. This raises the question, to which extent these techniques may be applied in the analysis of DBA.

**Overview** In Section 6.2 we give two different upper bounds for the number of iterations of DBA in the worst-case. The first one is an exponential upper bound that is based on techniques from real algebraic geometry and uses specific properties of the space of dynamic time warping paths. The second bound is based on a potential function argument and it depends on certain geometric properties of the input data. More precisely, we obtain a worst-case upper bound that is linear in the length of the point sequences,

and linear in $\frac{1}{\varepsilon}$. Here $\varepsilon$ is the minimal distance between any two mean point sequences that may be visited in the iterations of DBA.

In Section 6.3 we present our upper bounds for the expected number of iterations in a semi-random data model. More precisely, we perform a smoothed analysis of the number of iterations of DBA under Gaussian perturbation (with any variance $\sigma^2$) of deterministic data. The techniques we use for the smoothed analysis are quite versatile and include anti-concentration estimates and standard tail bounds for the norm of a random vector. So the obtained results can be easily generalized to broader distributions, e.g. sub-Gaussian random variables. However, we prefer to present the ideas in a less technical manner on Gaussian perturbations. We show that the expected number of iterations until DBA converges is at most $\widetilde{O}\left(n^2 m^{8\frac{n}{d}+6} d^4 k^6 \sigma^{-2}\right)$, where the $\widetilde{O}(\cdot)$-notation omits logarithmic factors.

These upper bounds are complemented by an exponential worst-case lower bound in Section 6.4. In particular, we show that there is an instance of two point sequences with length $m = \Theta(k)$ in the plane such that DBA needs $2^{\Omega(k)}$ iterations to converge. The techniques in this section borrow from earlier work of Vattani [130]. Interestingly, our lower bound shows this behaviour already for only $n = 2$ sequences. In this setting, there also exists a dynamic programming algorithm that solves the same problem in $O(k^5)$ time [21], as mentioned above. Furthermore, we observe in Section 6.5 that, when applied to real-world data, the number of iterations that DBA performs on average is much lower than our theoretical analysis suggests. In particular, we observe only sublinear dependencies on any of the parameters $n, m$ and $k$. These empirical results support the implicit assumptions in our theoretical study of the algorithm, as the smoothed analysis under randomly perturbed instances avoids artificially constructed boundary cases and corresponding artificially high lower bounds.

### 6.1.1 Preliminaries

Let $X = \{\gamma_1, \ldots, \gamma_n\} \subset (\mathbb{R}^d)^m$ be a set of $n$ point sequences of length $m$ and $C \in (\mathbb{R}^d)^k$ be a point sequence of length $k$. We call a sequence $\pi$ of tuples $(p, c)$ where $p$ is an element of some point sequence in $X$ and $c$ is an element of the point sequence $C$ an ***assignment map*** between $X$ and $C$. We call an assignment map between $X$ and $C$ ***valid*** if for each $1 \leq i \leq n$ the sequence of all tuples $(p, c)$ of $\pi$ for which $p$ is a point of $\gamma_i$ forms a warping path $w(\pi)_i$ between $\gamma_i$ and $C$. We call a valid assignment map ***optimal*** if for each $1 \leq i \leq n$ the formed warping path is an optimal warping path. We define the ***cost*** of an ***assignment map*** $\pi$ as $\Phi(\pi) = \sum_{(p,c)\in\pi} \|p - c\|^2$. Similarly, we define the ***total warping distance*** of a valid assignment map $\pi$ with respect to a point sequence $x = (x_1, \ldots, x_k)$ as $\Psi_\pi(x) = \sum_{i=1}^{n} \sum_{(j_1,j_2)\in w(\pi)_i} \|\gamma_{i,j_1} - x_{j_2}\|^2$.

### 6.1.2 The DBA Algorithm

Let $X$ be a set of $n$ point sequences $\gamma_1, \ldots, \gamma_n \in (\mathbb{R}^d)^m$. Let $C \in (\mathbb{R}^d)^k$ be another point sequence and $w^{(1)}, \ldots, w^{(n)} \in \mathcal{W}_{m,k}$ be chosen such that $w^{(i)}$ is an optimal warping path between $\gamma_i$ and $C$. Then $w^{(1)}, \ldots, w^{(n)}$ define an assignment map $\pi$ between $X$ and $C$. The assignment map $\pi$ can be represented by sets $S_1(\pi), \ldots, S_k(\pi)$, where

$$S_i(\pi) = \cup_{j=1}^{n} \{\gamma_{j,t} \mid (i, t) \in w^{(j)}\}.$$

By construction, $\pi$ minimizes the DTW distances between $\gamma_1, \ldots, \gamma_n$ and $C$. In the opposite direction, the following sequence $C_\pi$ minimizes the sum of squared distances for fixed $\pi$.

$$C_\pi = (c_1(\pi), c_2(\pi), c_3(\pi), \ldots, c_k(\pi))$$

where $c_i(\pi) := \frac{1}{|S_i(\pi)|} \sum_{p \in S_i(\pi)} p$. DBA alternately computes such assignment maps and average point sequences as follows.

1. Let $\pi_0$ be an initial assignment map (e.g. randomly drawn $w_0^{(1)}, \ldots, w_0^{(n)}$). Let $j \leftarrow 0$.
2. Let $j \leftarrow j + 1$. Compute the average point sequence $C_{\pi_{j-1}}$ based on $\pi_{j-1}$.
3. Compute optimal warping paths $w_j^{(i)}$ between $\gamma_i$ and $C_{\pi_{j-1}}$ for all $1 \leq i \leq n$. The warping paths define an optimal assignment map $\pi_j$ between $X$ and $C_{\pi_{j-1}}$.
4. If $\Phi(\pi_j) \neq \Phi(\pi_{j-1})$, then go to Step 2. Otherwise, terminate.

## 6.2 Upper bounds

We present two different upper bounds on the number of steps performed by the DBA algorithm. The first approach is based on a theorem from real algebraic geometry and the resulting bound holds for any input data. The second approach is based on a potential function argument and it uses a geometric assumption on the input data.

### 6.2.1 An unconditional upper bound

In this section, we will employ some classical tools from real algebraic geometry to derive an upper bound on the number of steps performed by DBA. In particular, we will use a bound on sign patterns of polynomials (Theorem 2.4.15) that is a variation of the classical bounds by Oleĭnik and Petrovski [108], Milnor [102], Thom [125] and Warren [132].

Suppose the input point sequences to DBA are $\gamma_1, \gamma_2, \ldots, \gamma_n$, and consider the set $\mathcal{W}^*_{m,k}$ of all possible optimal warping paths between a point sequence of length $m$ and a point sequence of length $k$. We define quadratic polynomials $F_{\gamma_i,w}$ that encode the cost of a path $w \in \mathcal{W}^*_{m,k}$ on the point sequence $\gamma_i$ as follows:

$$F_{\gamma_i,w}(x_1, x_2, \ldots, x_k) := \sum_{(j_1,j_2) \in w} \|\gamma_{i,j_1} - x_{j_2}\|^2$$

To be able to compare different warping paths between input point sequences and an average point sequence $x$, we consider the following family of quadratic polynomials.

$$\mathcal{F} := \{F_{\gamma_i,w}(x) - F_{\gamma_i,v}(x) : 1 \leq i \leq n \ , \ w, v \in \mathcal{W}^*_{k,m}\}$$

We make a simple observation: If $w$ is an optimal warping path between $\gamma_i$ and $C_\pi$ then for all $h \in \mathcal{W}^*_{m,k}$ we have $F_{\gamma_i,w}(C_\pi) - F_{\gamma_i,h}(C_\pi) \leq 0$. Now suppose the warping path $w$ between $\gamma_i$ and $C_\alpha \in \mathbb{R}^{dk}$ is part of the optimal assignment map between input point sequences and $C_\alpha$. Also suppose that DBA updates the warping path $w$ to another

path $v$ after the average point sequence is updated to $C_\beta$. This means the following inequalities are true

$$F_{\gamma_i,v}(C_\beta) - F_{\gamma_i,w}(C_\beta) \leq 0 \leq F_{\gamma_i,v}(C_\alpha) - F_{\gamma_i,w}(C_\alpha)$$

Since the cost decreases at every step, both of the inequalities cannot be an equality for all $i$ at the same time. In particular, there exists an $i$ such that the sign of quadratic polynomial $F_{\gamma_i,v}(x) - F_{\gamma_i,w}(x)$ or the sign of $F_{\gamma_i,w}(x) - F_{\gamma_i,v}(x)$ is different for the values $x = C_\alpha$ and $x = C_\beta$. This implies that the sign patterns of quadratic polynomials in $\mathcal{F}$ partition the space $\mathbb{R}^{dk}$ in such a way that two consecutive average point sequences computed by DBA are separated.

We now argue that the algorithm visits any connected component at most once.

**Lemma 6.2.1.** *DBA does not visit a connected component of the realizable sign patterns of $\mathcal{F}$ more than once (except for the very last step).*

*Proof.* Assume a connected component contains $C_{\pi_i}$ and $C_{\pi_j}$ where $i < j$ and the algorithm does not converge in $C_{\pi_j}$. Because two consecutive average point sequences computed by the algorithm are separated by the realizable sign patterns of $\mathcal{F}$, we have $i + 1 < j$. Since the cost decreases monotonically at every step of the algorithm, we have

$$\Psi_{\pi_i}(C_{\pi_i}) > \Psi_{\pi_{i+1}}(C_{\pi_{i+1}}) > \Psi_{\pi_j}(C_{\pi_j})$$

Observe that $\pi_{i+1}$ was computed based on $C_{\pi_i}$ and is thus an optimal assignment map for $C_{\pi_i}$. Within every connected component, the set of optimal warping paths remains unchanged. Therefore, $\pi_{i+1}$ is also optimal for $C_{\pi_j}$. Since $C_{\pi_{i+1}}$ is optimal for $\pi_{i+1}$, we have

$$\Psi_{\pi_j}(C_{\pi_j}) \geq \Psi_{\pi_{i+1}}(C_{\pi_j}) \geq \Psi_{\pi_{i+1}}(C_{\pi_{i+1}}).$$

This contradicts the statement that the cost decreases at every step. □

Lemma 6.2.1 implies that the number of steps of DBA algorithm (except the very last step) is upper bounded by the number of visited regions in the sign realization of the family $\mathcal{F}$. Using Theorem 2.4.15, we can give an upper bound on the number of sign patterns of the quadratic forms $\mathcal{F}$.

**Lemma 6.2.2.** *The number of sign patterns of $\mathcal{F}$ is at most $6(4n|\mathcal{W}^*_{m,k}|^2)^{dk}$.*

**Observation 6.2.3.**

$$|\mathcal{W}^*_{m,k}| \leq \binom{m+k-2}{m-1} \leq \min\{m^{k-1}, k^{m-1}\}.$$

*Proof.* Indeed, the number of paths on an $m \times k$ grid that can take steps $(i, j) \to (i + 1, j)$ or $(i, j) \to (i, j + 1)$ is given by $\binom{m+k-2}{m-1}$, since any such path takes $m + k - 2$ steps. However, our warping paths may have diagonal steps of the form $(i, j) \to (i + 1, j + 1)$. Note that such a diagonal step corresponds to 'omitting' an assignment term between two points from the total sum and therefore always leads to a decrease in the cost. Thus, allowing diagonal steps does not increase the total number of realizable warping paths. □

Now, the bound in Lemma 6.2.2 together with Observation 6.2.3 immediately gives an upper bound on the number of steps of DBA leading to the following theorem.

**Theorem 6.2.4.** *DBA converges in at most $6(4n)^{dk}\binom{m+k-2}{m-1}^{2dk}$ iterations.*

### 6.2.2 Upper bound based on geometric properties of the input data

We denote the set of valid assignment maps from $n$ input point sequences of length $m$ to a point sequence of length $k$ with $\mathcal{A}_{n,m,k}$. It is $|\mathcal{A}_{n,m,k}| \leq m^{kn}$. For an assignment map $\pi \in \mathcal{A}_{n,m,k}$ and a point sequence $x = (x_1, x_2, \ldots, x_k)$ with $x_i \in \mathbb{R}^d$, we can rewrite the total warping distance as

$$\Psi_\pi(x) := \sum_{i=1}^{k} \sum_{y \in S_i(\pi)} \|y - x_i\|^2$$

Our first observation is that for all $x$ we have $\Psi_\pi(x) \geq \Psi_\pi(C_\pi)$. We would like to express $\Psi_\pi(x)$ in a way that resembles an inertia. We set $I_\pi := \sum_{i=1}^{k} \sum_{y \in S_i(\pi)} (\|y\|^2 - \|c_i(\pi)\|^2)$. Observe that $I_\pi = \Psi_\pi(C_\pi)$. We see $I_\pi$ as the inertia of $\pi$, and we have

$$\Psi_\pi(x) = I_\pi + \sum_{i=1}^{k} |S_i(\pi)| \, \|x_i - c_i(\pi)\|^2$$

We use the following geometric properties of the input:
**Normalization Property:** Let $B > 0$. For any vector $y$ in any input point sequence, we have $\|y\|^2 \leq B$.
**Separation Property:** Let $\varepsilon > 0$. For any two different assignment maps $\alpha, \beta \in \mathcal{A}_{n,m,k}$ we have

$$\|C_\alpha - C_\beta\|^2 := \sum_{i=1}^{k} (c_i(\alpha) - c_i(\beta))^2 \geq \varepsilon$$

With the help of the following lemma, we can derive an upper bound in Theorem 6.2.6.

**Lemma 6.2.5.** *For all $\alpha \in \mathcal{A}_{n,m,k}$, it is $I_\alpha \leq Bn(m+k)$.*

*Proof.*

$$I_\alpha = \sum_{i=1}^{k} \sum_{y \in S_i(\alpha)} (\|y\|^2 - \|c_i(\alpha)\|^2) \leq B \sum_{i=1}^{k} |S_i(\alpha)|$$

Note that every warping path between the average point sequence of length $k$ and an input point sequence of length $m$ consists of at most $m + k$ many steps. This means every point sequence contributes to the sum $\sum_{i=1}^{k} |S_i(\alpha)|$ with at most $m + k$ elements, and hence we have $\sum_{i=1}^{k} |S_i(\alpha)| \leq n(m + k)$. □

**Theorem 6.2.6.** *If the input data satisfies the normalization property with parameter $B$ and separation property with parameter $\varepsilon$, then the number of steps performed by DBA is at most*

$$\frac{B(m+k)}{\varepsilon}$$

*Proof.* Suppose DBA has started from assignment map $\pi_0$. If DBA takes a step from some assignment map $\alpha$ to some assignment map $\beta$ this means

$$\begin{aligned} I_\alpha &= \Psi_\alpha(C_\alpha) > \Psi_\beta(C_\alpha) \\ &= I_\beta + \sum_{i=1}^{k} |S_i(\beta)| \, \|c_i(\alpha) - c_i(\beta)\|^2 \end{aligned}$$

Since $|S_i(\beta)| \geq n$ for all $i$, this implies $I_\alpha > I_\beta + n\|C_\alpha - C_\beta\|^2$. Using the separation property of the data and Lemma 6.2.5 , we get

$$I_\alpha > I_\beta + n\varepsilon \geq I_\beta + \frac{\varepsilon}{B(m+k)} I_{\pi_0} \tag{6.1}$$

Let $\pi_T$ be the assignment map after $T$ steps of DBA, then we have by (6.1) that

$$I_{\pi_T} < I_{\pi_0} - T\frac{\varepsilon}{B(m+k)} I_{\pi_0}$$

□

## 6.3 Smoothed Analysis

Our randomness model is as follows: An adversary specifies an instance $X' \in ([0,1]^d)^{nm}$ of $n$ point sequences $\gamma_1, \ldots, \gamma_n$ of length $m$ in $[0,1]^d$, where each sequence $\gamma_i$ is given by its $m$ points $\gamma_i = (\gamma_{i,1}, \ldots, \gamma_{i,m})$. Then we add to each point of $X'$ an $d$-dimensional random vector with independent Gaussian coordinates of mean 0 and standard deviation $\sigma$. The resulting vectors form the input point sequences. We assume without loss of generality that $\sigma \leq 1$, since the case $\sigma > 1$ corresponds to a scaled down instance $X' \in ([0, \frac{1}{\sigma}]^d)^{nm}$ with additive $d$-dimensional Gaussian random vectors with mean 0 and standard deviation 1. We call this randomness model $m$-length sequences with $\mathcal{N}(0,\sigma)$ perturbation.

We note that the results in this section hold for a more general family of random input models (See Section 1.5 of [60] or Section 3.1 of [49]). We conduct the analysis only for Gaussian perturbation for the sake of simplicity and obtain the following theorem.

**Theorem 6.3.1.** *Suppose $d \geq 2$, then the expected number of iterations until DBA converges is at most*

$$O\left(\frac{n^2 m^{8\frac{n}{d}+6} d^4 k^6 \ln(nm)^4}{\sigma^2}\right).$$

To prove Theorem 6.3.1, we first bound the probability that the normalization property and the separation property hold for suitable parameters.

**Lemma 6.3.2.** *We have*

$$\mathbb{P}\{\max_{1\leq i\leq n} \max_{y\in\gamma_i} \|y\| \geq \sqrt{d} + t\sigma\sqrt{2d\ln(nm)}\} \leq e^{1-t^2}$$

*Proof.* By our assumptions, every vector in input sequences is given by $D + Y$ where $D$ is a deterministic vector with norm at most $\sqrt{d}$ and $Y$ is a random vector with Gaussian i.i.d coordinates $\mathcal{N}(0,\sigma)$. By triangle inequality $\|D+Y\| \leq \sqrt{d} + \|Y\|$. Since $Y$ has Gaussian i.i.d coordinates $\mathcal{N}(0,\sigma)$, we can apply the standard tail bound $\mathbb{P}\{\|Y\| \geq t\sigma\sqrt{d}\} \leq e^{1-\frac{t^2}{2}}$. □

**Lemma 6.3.3.** *Let $C_\alpha$ and $C_\beta$ be average point sequences corresponding to two different assingment maps $\alpha$ and $\beta$. Then, we have $\mathbb{P}\{\|C_\alpha - C_\beta\|^2 \leq \varepsilon\} \leq \left(\frac{nm\sqrt{\varepsilon}}{\sigma}\right)^d$. Furthermore, the separation property with parameter $\varepsilon$ holds with probability at least*

$$1 - m^{4n}\left(\frac{nm\sqrt{\varepsilon}}{\sigma}\right)^d.$$

*Proof.* Since the instances are perturbed and the assignment maps are different, we have with probability 1 that there exists an $i \in [k]$ such that $c_i(\alpha) \neq c_i(\beta)$. The event $\|C_\alpha - C_\beta\|^2 \leq \varepsilon$ further implies $\|c_i(\alpha) - c_i(\beta)\| \leq \sqrt{\varepsilon}$. We bound the probability that this event $\|c_i(\alpha) - c_i(\beta)\| \leq \sqrt{\varepsilon}$ occurs for the fixed $c_i(\alpha)$ and $c_i(\beta)$.

Let $S_i(\alpha)$ and $S_i(\beta)$ denote the sets of points in $X$ that were assigned to $c_i(\alpha)$ and $c_i(\beta)$ respectively. Since $c_i(\alpha) \neq c_i(\beta)$, it immediately follows that $S_i(\alpha) \neq S_i(\beta)$. So we can fix a point $s \in S_i(\alpha) \triangle S_i(\beta)$. We let an adversary fix all points in $S_i(\alpha) \cup S_i(\beta) \setminus \{s\}$. In order for $c_i(\alpha)$ and $c_i(\beta)$ to be $\sqrt{\varepsilon}$-close, we need $s$ to fall into a hyperball of radius $nm\sqrt{\varepsilon}$. Because $s$ is drawn from a Gaussian distribution with standard deviation $\sigma$, this happens with probability at most $\left(\frac{nm\sqrt{\varepsilon}}{\sigma}\right)^d$. So in total, we have $\mathbb{P}\{\|c_i(\alpha) - c_i(\beta)\| \leq \sqrt{\varepsilon}\} \leq \left(\frac{nm\sqrt{\varepsilon}}{\sigma}\right)^d$ and therefore $\mathbb{P}\{\|C_\alpha - C_\beta\|^2 \leq \sqrt{\varepsilon}\} \leq \left(\frac{nm\sqrt{\varepsilon}}{\sigma}\right)^d$.

To prove the second claim, we apply a union bound over all possible choices for $c_i(\alpha)$ and $c_i(\beta)$. Since each $c_i(\alpha)$ and $c_i(\beta)$ is uniquely determined by its assigned points $S_i(\alpha)$ and $S_i(\beta)$ it suffices to bound these. For each input point sequence, there are at most $\binom{m}{2}$ possible choices for the set of points that get assigned to a fixed center point: This is the case since all points that get assigned to the same center point have to be consecutive. So the points that get assigned to the center point are uniquely determined by the first and the last point that gets assigned to the center point. For $n$ input point sequences all possible assignments to an arbitrary center point are therefore bounded by $\binom{m}{2}^n$. Since we choose two center points $c_i(\alpha)$ and $c_i(\beta)$, there are at most $\binom{m}{2}^{2n} \leq m^{4n}$ possible choices for the assigned points $S_i(\alpha)$ and $S_i(\beta)$ that determine $c_i(\alpha)$ and $c_i(\beta)$. The statement follows by applying the union bound over possible choices for $c_i(\alpha)$ and $c_i(\beta)$. □

As a combination of Lemma 6.3.2 and Lemma 6.3.3, we get the following lemma.

**Lemma 6.3.4.** *Let $\gamma_1, \gamma_2, \ldots, \gamma_n$ be independent $m$-length sequences with $\mathcal{N}(0, \sigma)$ perturbation, and suppose $d \geq 2$. Then, the DBA algorithm implemented on the input data $\gamma_1, \gamma_2, \ldots, \gamma_n$ converges in at most*

$$s\left(\frac{a_1 n^2 m^{8\frac{n}{d}+5} d^3 k^5 \ln(nm)^3}{\sigma^2}\right)$$

*steps with probability at least $1 - s^{-\frac{d}{2}} - (4n)^{-dk} - 2^{-8mdk^2}$ for all $s \geq 1$ where $a_1$ is constant.*

*Proof.* In Lemma 6.3.2, we set $t = 2\ln((4n)^{dk} 2^{8mdk^2}) \leq 16mdk^2 \ln(8n)$ to get that the normalization property is fulfilled for some $B \leq a_2 \sigma^2 d^3 k^4 m^2 \ln(nm)^3$ with probability at least $1 - (4n)^{-dk} - 2^{-8mdk^2}$ where $a_2$ is constant. Then we use Lemma 6.3.3 with $\varepsilon = \frac{\sigma^2}{sn^2 m^{8\frac{n}{d}+2}}$ and apply Theorem 6.2.6. □

With the help of the Lemma 6.3.4, we are now ready to prove Theorem 6.3.1.

*Proof of Theorem 6.3.1.* Let $X$ be the number of steps that DBA performs. By Theorem 6.2.4, we have that $X \leq a_3(4n)^{dk} 2^{8mdk^2}$ for some constant $a_3$. We set $M := a_3(4n)^{dk} 2^{8mdk^2}$ for simplicity and get

$$\mathbb{E}[X] = \sum_{i=1}^{M} \mathbb{P}\{X \ \geq i\} \leq K + \sum_{t=K}^{M} \mathbb{P}\{X \ \geq t\}$$

for any $K$. We set $K \coloneqq a_1 n^2 m^{8\frac{n}{d}+5} d^3 k^5 \ln(nm)^3/\sigma^2$. By Lemma 6.3.4, it is

$$\mathbb{P}\{X \ \geq sK\} \leq s^{-\frac{d}{2}} + (4n)^{-dk} + m^{-8mdk^2}$$

for all $s \geq 1$. Therefore, we have

$$\sum_{t=K}^{M} \mathbb{P}\{X \geq t\} \leq K \cdot \sum_{s=1}^{\frac{M}{K}} s^{-\frac{d}{2}} + (4n)^{-dk} + m^{-8mdk^2}$$

Since $d \geq 2$, we have $s^{\frac{-d}{2}} \leq \frac{1}{s}$. Moreover, $\frac{M}{K}\left((4n)^{-dk} + m^{-8mdk^2}\right) \leq 1$. So, we have

$$\sum_{t=K}^{M} \mathbb{P}\{X \geq t\} \leq K\left(1 + \sum_{s=1}^{\frac{M}{K}} \frac{1}{s}\right) \leq K + K \ln \frac{M}{K}$$

Hence,

$$\mathbb{E}X \leq 2K + K \ln \frac{M}{K} \leq 2K + Kmdk^2 \ln\left(a_4 n\right)$$

for some constant $a_4$.

□

**Remark 6.3.5.** *Note that for the discrete case, where the positions of the points in the center point sequence are restricted to the input points, we would get an upper bound on the number of iterations which would be polynomial in $n$, instead of exponential in $n$, since for the positions of $c_i(\alpha)$ and $c_i(\beta)$ in the proof of Lemma 6.3.3, there are only $\binom{nm}{2}$ instead of $\binom{m}{2}^{2n}$ possible choices.*

## 6.4 Lower bound

In this section, we give a lower bound on the worst-case number of iterations until DBA converges for two input point sequences of length $m \in \mathbb{N}$ and an average point sequence of length $k = \Theta(m)$ in $\mathbb{R}^2$. By duplicating the input point sequences this lower bound immediately yields a lower bound for $n \in \mathbb{N}$ input point sequences of length $m$ and an average point sequence of length $k = \Theta(m)$ in $\mathbb{R}^d$. To construct the instance for our lower bound, we directly draw from the work of Andrea Vattani on the lower bound for the worst-case number of iterations of the $k$-means method [130]. We take his exact construction and modify it to suit our needs by connecting input points to point sequences and scaling up the integer weights of the points by a constant factor. Here a point with integer weight resembles multiple points in the same position that are consecutive points of the corresponding point sequence. The resulting instance will lead to a lower bound that is exponential in $m$.

### 6.4.1 Construction

Before we give a detailed construction, we give a motivation for modifying a $k$-means construction to show a lower bound for the number of iterations of the DBA algorithm. The reason is that the algorithms are very similar. In fact, we can observe that DBA can behave exactly like the $k$-means algorithm and converge in the same number of iterations

if in all iterations the optimal $k$-means assignment maps are also valid DBA assignment maps.

To construct a suitable DBA instance, the idea is to connect the points of the instance in [130] to point sequences such that this condition holds. This is not directly possible, so we modify the instance such that we still get the same number of iterations for both algorithms. A challenge here is that at some steps of the algorithm some points of the average point sequence would not get mapped to any point of one of the input point sequences by the assignment map corresponding to $k$-means. This happens independent of the choice of the point sequences and implies that the respective assignment maps are not valid. Intuitively, the challenge is solved by letting the problematic points on the average sequence "steal" points from neighboring clusters. To ensure that the general structure remains unchanged, we replace all points of the input point sequences by multiple points in the same position. In the following, we give a detailed description of the construction of the DBA instance based on the instance in [130].

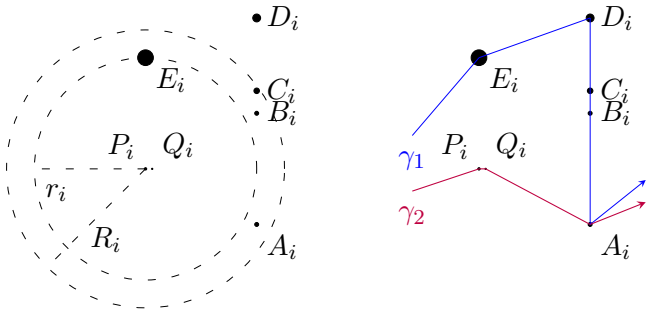


Figure 6.1: Schematic drawing of gadget $G_i$ ($i \geq 1$) in $k$-means instance (Left) and in DBA instance (Right).

The instance $X'$ of Andrea Vattani consists of a sequence of gadgets $G_0, G_1, \ldots, G_{\lceil \frac{k-1}{2} \rceil}$. For each $i \neq 0$, the gadget $G_i$ is given by a tuple $(\mathcal{P}_i, \mathcal{T}_i, r_i, R_i)$ where the set $\mathcal{P}_i = \{P_i, Q_i, A_i, B_i, C_i, D_i, E_i\} \subset \mathbb{R}^2$ determines the positions of the input points, $\mathcal{T}_i \subset \mathbb{R}^2$ determines the initial position of two centers corresponding to the gadget and $r_i > 0$ and $R_i > r_i$ are the inner and outer radius of the gadget. In each position of $\mathcal{P}_i$ the gadget contains a constant number of points determined by weights $w_P, w_Q, w_A, w_B, w_C, w_D, w_E$ that are independent of the index $i$ (see Table 6.1 for the exact values). Note that for our use of the instance, we scale the weights by a constant factor of $100M$, where $M \in \mathbb{N}$ will be determined later. The positions of each gadget are identical up to translation of the position $P_i$ and up to a scaling of the relative distance of all other positions to $P_i$. The position can be seen in Table 6.1. A $k$-means instance needs $k$ initial centers $C = \{c_1, \ldots, c_k\}$. We say that a center point $c_j$ corresponds to a gadget $G_i$, if $\lceil \frac{j-1}{2} \rceil = i$. The initial position $\mathcal{T}_i$ of the two centers corresponding to $G_i$ are determined as the mean of the points in an initial cluster. The two initial clusters for each gadget $G_i$ with $i \neq 0$ are defined as all points in the position $A_i$ and all points in the positions $P_i, Q_i, B_i, C_i, D_i, E_i$. For an illustration of one Gadget $G_i$ of $X'$ with $i \neq 0$ see Figure 6.1. The gadget $G_0$ is given by $(\mathcal{P}_0, C_0)$ where $\mathcal{P}_0 = C_0 = \{F\}$. So all the $w_F$ points in position $F = (0, 0)$ determine the position $(0, 0)$ of the initial center $c_1$.

To construct an instance $X$ for DBA out of the instance $X'$ for the $k$-means algorithm,

| Weights | Positions $\mathcal{P}_i$ | Other values |
|---|---|---|
| $w_P = 100M$ | $P_i = S_{i-1} + (1-\varepsilon)R_i(1,0)$ | $\varepsilon = 10^{-6}$ |
| $w_Q = 1M$ | $Q_i = P_i + r_i(10^{-5}, 0)$ | $r_1 = 1$ |
| $w_A = 400M$ | $A_i = P_i + r_i(1, -0.5)$ | $r_i \approx 40.41608 r_{i-1}$ |
| $w_B = 400M$ | $B_i = P_i + r_i(1, 0.5)$ | $R_i = 1.25 r_i$ |
| $w_C = 1100M$ | $C_i \approx P_i + r_i(1, 0.70223)$ | $w_F = 5000M$ |
| $w_D = 3100M$ | $D_i \approx P_i + r_i(1, 1.35739)$ | $F = S_0 = (0,0)$ |
| $w_E = 27400M$ | $E_i = P_i + r_i(0,1)$ | $S_i \approx P_i + r_i(1, 0, 99607)$ |

Table 6.1: Approximate weights and positions of the points in the $k$-means instance $X'$ of [130].

we just connect the points of the gadgets together to form two input point sequences. The first point sequence $\gamma_1$ starts with half of the points in position $F$ followed by all points in the positions $E_i, D_i, C_i, B_i$ and half of the points in position $A_i$ in the given order and in the natural order $1, \ldots, \lceil \frac{k-1}{2} \rceil$ of the gadgets $G_i$. The second point sequence $\gamma_2$ starts with the other half of the points in position $F$ followed by all points in the positions $P_i, Q_i$ and the other half of the points in position $A_i$ (see also Figure 6.1). The initial assignment map is chosen as in the $k$-means instance so that the initial average point sequence has its points $c_1, \ldots, c_k$ in the same position as the centers of the $k$-means instance.

### 6.4.2 Analysis

For a run of the $k$-means algorithm on the instance $X'$, all the occurring assignment maps between points and centers in any step of the algorithm are described in [130]. Directly from the positions of the center points given by these assignment maps, we can observe the following properties for any set of center points that is present in any iteration of the $k$-means algorithm.

**Observation 6.4.1.** *Let $C = \{c_1, \ldots, c_k\}$ be a set of center points present in any iteration of the $k$-means algorithm on $X'$. The following properties hold for $C$ and any $0 \leq i \leq \lceil \frac{k-1}{2} \rceil$:*

1. *Let $p$ be a point that lies in the gadget $G_i$. Let $c_j \in C$ be the closest center point to $p$ (i.e. $c_j = \arg\min_{c \in C} \|p - c\|$). We have*

$$\max_{c \in \{c_{j-1}, c_j, c_{j+1}\}} \|p - c\| \leq \alpha r_i$$

   *where $\alpha > 0$ is a constant that is independent of $i, p$ and the number of gadgets.*

2. *Let $p$ be a point that lies in either one of the gadgets $G_{i-1}, G_i, G_{i+1}$. Let further $c' = \arg\min_{c \in C} \|p - c\|$ be the closest center point and $c'' = \arg\min_{c \in C \setminus \{c'\}} \|p - c\|$ bet the second closest center point. We have*

$$\|p - c''\| - \|p - c'\| \geq \beta r_i$$

   *where $\beta > 0$ is a constant that is independent of $i, p$ and the number of gadgets.*

Furthermore, it is easy to check that the following method always creates a valid assignment map to the center point sequence in any iteration of DBA on the instance $X$, based on the assignment map in the same iteration of the $k$-means algorithm.

*Method for creating valid assignment map:* Take a fixed iteration of the $k$-means algorithm on $X'$. Let $C' = (c'_1, \ldots, c'_k)$ be the sequence of centers in this iteration of the $k$-means algorithm and let $C = (c_1, \ldots, c_k)$ be the points of the average point sequence in this iteration of DBA. Let $\pi'$ be the assignment map that assigns each point in $X'$ to its nearest neighbor in $C'$. We can interpret $\pi'$ also as an assignment map of $X$ that assigns points of $X$ that lie in the same position as points of $X'$ to points of the center point sequence $C$ that have the same index as the assigned centers in $C'$. Let $I$ be the set of the indices of all points in $C$ that do not get assigned to any point of the point sequence $\gamma_2$ by this assignment map $\pi'$. Note that there are no $i, j \in I$ with $i = j + 1$. For each point $j \in I$ take the last point $p_-(c_j)$ of $\gamma_2$ that got assigned to $c_{j-1}$ and the first point $p_+(c_j)$ of $\gamma_2$ that got assigned to $c_{j+1}$. Choose $p(c_j) = \arg\min_{p \in \{p_-(c_j), p_+(c_j)\}} \|c_j - p\|$. Replace the previous assignment of $p(c_j)$ with $c_j$. Since we scale up the instance $\pi$ by a constant factor $M$, there are at least $M$ points of $Q$ at position $p(c_j)$. Only reassign the first respectively last of these points to $c_j$. Denote the newly created assignment map with $\pi$.

In the following, we prove that the created valid assignment map $\pi$ is also always an optimal assignment map for the instance $X$ (scaled by a suitable constant $M$) and its corresponding center point sequence. This result then directly implies that DBA on $X$ needs as many iterations as the $k$-means algorithm on $X'$ to converge, since the assignment map changes in each two consecutive iterations of DBA, in which the corresponding assignment map of the $k$-means algorithm changes. So DBA only converges in any iteration, in which the $k$-means algorithm converges.

**Theorem 6.4.2.** *Let $M > 4\frac{\alpha}{\beta} + 4\frac{\alpha^2}{\beta^2}$. For any iteration of DBA on the instance $X$ scaled up by the factor $M$, the valid assignment map $\pi$ is an optimal assignment map.*

*Proof.* Fix an iteration of the $k$-means algorithm on $X'$. If we take the first iteration, then the center points (that the new assignment map is based on) are the same as the points of the average point sequence in the first iteration of DBA. Otherwise, let $\pi'_1$ be the assignment map in the previous iteration of the $k$-means algorithm and $\pi_1$ be the corresponding assignment map in the previous iteration of DBA.

We first show that the center points of $C_{\pi'_1}$ that correspond to gadget $G_i$ differ from their corresponding center points of $C_{\pi_1}$ by at most $\frac{\alpha}{M} r_i$. From the construction of $\pi'_1$, we know that the assigned points to each center point $c_j(\pi'_1)$ in $\pi'_1$ and $c_j(\pi_1)$ in $\pi_1$ differ by at most one point. Assume that $c_j(\pi_1)$ has one more point $p$. We have

$$\begin{aligned}
\|c_j(\pi'_1) - c_j(\pi_1)\| &= \|c_j(\pi'_1) - \frac{c_j(\pi'_1) \cdot |S_j(\pi'_1)| + p}{|S_j(\pi'_1)| + 1})\| \\
&= \|\frac{c_j(\pi'_1) - p}{|S_j(\pi')| + 1}\| \\
&\leq \frac{1}{M}\|c_j(\pi'_1) - p\| \\
&\leq \frac{\alpha}{M} r_i
\end{aligned}$$

Here the second to last inequality follows by $|S_j(\pi')| \geq M$ (at least one scaled-up point is assigned) and the last inequality follows by the first part of Observation 6.4.1. The case

that $c_j(\pi'_1)$ has one more point is analogous. By the second property of Observation 6.4.1, we know that the difference between the distances of a point from gadget $G_{i-1}, G_i$ or $G_{i+1}$ to its closest center and its second closest center on $C_{\pi'_1}$ is at least $\beta r_i$. So, for $M > 2\frac{\alpha}{\beta}$, it holds that

$$2\frac{\alpha}{M}r_i < \beta r_i,$$

and it is ensured that the closest center of each point in both assignment maps $\pi'_1$ and $\pi_1$ has the same index. It further holds that the difference between the distances of a point from gadget $G_{i-1}, G_i$ or $G_{i+1}$ to its closest center $c'$ and its second closest center $c''$ on $C_{\pi_1}$ is at least

$$\|c'' - p\| - \|c' - p\| \geq \beta r_i - 2\frac{\alpha}{M}r_i. \tag{6.2}$$

This is an important property for showing the optimality of the new assignment map.

Let $\pi'$ be the assignment map in the fixed iteration of the $k$-means algorithm corresponding to the assignment map $\pi$ in the same iteration of DBA. We have to show that $\pi$ is an optimal assignment map between the input point sequences the average point sequence $C_{\pi_1}$ generated from the assignment map of the previous iteration (or the starting average point sequence in the first iteration). Let $OPT = \sum_{p \in X} \min_j \|c_j(\pi_1) - p\|^2$ be the cost of the assignment map $\pi'$ between the input point sequences and $C_{\pi_1}$. We know that this assignment map is optimal, if it is valid since each point of the input point sequence has to be assigned to at least one point of the center point sequence and $\pi'_j$ assigns each point only to its nearest neighbor. But this assignment map is not valid, since there are multiple points of $C_{\pi_1}$ that do not get assigned to any point of the point sequence $\gamma_2$ in instance $X$. We denote the index set of these center points with $I$ and denote the points of $C_{\pi_1}$ simply with $c_1, \ldots, c_k$ instead of $c_1(\pi_1), \ldots, c_k(\pi_1)$. We show that for suitable $M$ each valid assignment map results in a greater cost than $\pi$, by comparing their resulting costs to $OPT$. Fix an arbitrary subset $I'$ of $I$. The cost of any assignment map that does not assign any point $c_j$ with $j \in I'$ to either $p_+(c_j)$ or $p_-(c_j)$ is greater than $OPT$ by at least

$$M \sum_{c_j \in I'} \min(\Delta_+(j), \Delta_-(j)) + \sum_{c_j \in I \setminus I'} \|c_j - p(c_j)\|^2 - \min_t(\|c_t - p(c_j)\|^2)$$

where

$$\Delta_+(j) = \min_{t \neq j+1}(\|c_t - p_+(c_j)\|^2) - \|c_{j+1} - p_+(c_j)\|^2,$$
$$\Delta_-(j) = \min_{t \neq j-1}(\|c_t - p_-(c_j)\|^2) - \|c_{j-1} - p_-(c_j)\|^2.$$

Here we use that for each $c_j \in I$, either all the $M$ points in position $p_+(c_j)$ or in position $p_-(c_j)$ have to get reassigned to another center point. The assignment map $\pi$ is the best assignment map that assigns all $c_j \in I$ to either $p_+(c_j)$ or $p_-(c_j)$. The cost of $\pi$ is exactly

$$\sum_{c_j \in I} \|c_j - p(c_j)\|^2 - \min_t(\|c_t - p(c_j)\|^2)$$

greater than $OPT$. So, if $M$ is chosen such that

$$M \min(\Delta_+(j), \Delta_-(j)) > \|c_j - p(c_j)\|^2 \tag{6.3}$$

then $\pi$ is an optimal assignment map. By the first part of Observation 6.4.1, we have $\|c_j(\pi'_1) - p(c_j)\| \leq \alpha r_i$. Since the center points of $C_{\pi'_1}$ that correspond to gadget $G_i$ differ from their corresponding center points of $C_{\pi_1}$ by at most $\frac{\alpha}{M} r_i \leq \alpha r_i$, we further get by triangle inequality

$$\|c_j - p(c_j)\| \leq \|c_j(\pi'_1) - p(c_j)\| + \|c_j(\pi'_1) - c_j\| \leq 2\alpha r_i$$

Since $c_{j+1}$ is the closest center to $p_+(c_j)$ and $c_{j-1}$ is the closest center to $p_-(c_j)$, we get by Equation (6.2) that

$$\min(\Delta_+(j), \Delta_-(j) > (\beta - 2\frac{\alpha}{M})^2 r_i^2.$$

Here we used that for any $a > b > 0$ it is $(a-b)^2 > a^2 - b^2$. It remains to show that $M(\beta - 2\frac{\alpha}{M})^2 > 4\alpha^2$. We have

$$\begin{aligned} M(\beta - 2\frac{\alpha}{M})^2 &= M\beta^2 - 4\beta\alpha + 4\frac{\alpha^2}{M} \\ &> M\beta^2 - 4\beta\alpha \\ &> 4\alpha^2 \end{aligned}$$

The last inequality follows by the assumption of the theorem that $M > 4\frac{\alpha}{\beta} + 4\frac{\alpha^2}{\beta^2}$. So Equation (6.3) is fulfilled and $\pi$ is an optimal assignment map. □

Since $M$ can be chosen as a constant independent of $k$ the point sequences $\gamma_1$ and $\gamma_2$ of $X$ have a length $\Theta(k)$ each. We technically require that both point sequences have the same length. The difference in length can easily be balanced by adding an extra gadget in front of gadget $G_0$ far away from the other gadgets that contains one point of the longer point sequence and the difference in length plus one points of the smaller point sequence. Also, add another center point corresponding to the gadget and initialize it at the mean of all points in the gadget. If all the points are far enough away from any other points of the instance, the corresponding center point does not change its position and the gadget does not interfere with any other gadget. By the results of [130], we know that the $k$-means algorithm needs $2^{\Omega(k)}$ iterations on the instance $X'$. As stated earlier, Theorem 6.4.2 implies that DBA on $X$ needs the same amount of iterations. We therefore achieve the following lower bound.

**Theorem 6.4.3.** *Let $k \in \mathbb{N}$. There is an instance of two point sequences with length $m = \Theta(k)$ in the plane such that DBA needs $2^{\Omega(k)}$ iterations to converge.*

## 6.5 Experiments on the M5 data set

In this section, we present our empirical observations on the number of iterations of DBA on real-world data. We run our implementation of the algorithm on real-world data sets of time series and observe that in practice it runs much faster than the theoretical guarantees ensure.

### 6.5.1 Research questions

We are interested in how the number of iterations of the DBA algorithm depends on the complexity of the input and output for practical data sets. More specifically, we ask the following research questions.

- What is the dependency of the number of iterations on the number $n$ of input point sequences?
- What is the dependency of the number of iterations on the length $m$ of the input point sequences?
- What is the dependency of the number of iterations on the length $k$ of the output center?

### 6.5.2 Data set(s)

To answer our research questions, we apply the DBA algorithm on the data set from the M5 Competition [100]. Preliminary experiments on data sets from the UCR Time Series Classification Archive [42] have not shown a clear dependency on any of the quantities in question. We conjecture that this result can be attributed to the heterogeneity of the data sets and the relatively short length of the studied input point sequences. For a more detailed analysis of these preliminary experiments see Appendix 6.7.

The data set of the M5 Competition consists of the unit sales of products from 10 different Walmart stores in the USA. The sold number of units was tracked daily over 1942 days from 2011-01-29 to 2016-06-19 for each product aggregated for each store separately. The products can be divided into the 7 product departments Hobbies 1-2, Foods 1-3 and Household 1-2. To create input sequences for DBA, we take for each product the time series that consists of the summed-up daily sales of this product over all 10 stores. The number of input sequences in each department is given in Table 6.2. In our experiments, we run DBA on sets of subsequences of such input sequences.

| Hobbies 1 | Hobbies 2 | Foods 1 | Foods 2 | Foods 3 | Household 1 | Household 2 |
|---|---|---|---|---|---|---|
| 416 | 149 | 216 | 398 | 823 | 532 | 515 |

Table 6.2: Number of input sequences (different products) per department.

### 6.5.3 Setup of the experiments

To perform experiments, DBA was implemented in C++. The implementation is based on the python implementation of Petitjean [106, 107] and can be found at [24]. In the following, we describe the setup of the experiments. All of our experiments use the same random initialization method to select the first center point sequence.

#### Random initialization

We initialize the first center point sequence from a random valid assignment of the input sequences. The valid assignment is created by a combination of random walks consisting of one walk per input sequence. Such a random walk starts with assigning the first point of the input series to the first point of the center point sequence and then chooses the next assignment by either moving forward on both input and center sequence or only moving forward on one of them. Each option has probability $1/3$. If it reaches the end of one sequence it continues to only move forward on the other one. After the random assignment is computed, the center points are chosen as the arithmetic means of the assigned points from the input series.

**Experiment 1: Dependency on the number $n$ of input point sequences**

We run DBA on sets of subsequences of consecutive days with varying sizes. The specific sizes of the sets are taken exponentially growing as 25, 50, 100, 200, 400, 800, 1600 and 3049. To choose the time series for a specific set of size $s$, we draw a number $r$ between 1 and $3049 - s$ uniformly at random and take the time series at the positions $r, r+1, \ldots, r+s$. We perform the experiment three times for three fixed values 100, 300 and 500 for the length of the subsequences. In each run of the experiment, we run DBA ten times for each fixed size of input sets and for input subseries of the fixed length, where we draw the starting day of the input subseries uniformly at random (same starting day for all series in the same run). For each run of DBA, we track the number of iterations.

**Experiment 2: Dependency on the length $m$ of the input point sequences**

In this experiment, we run DBA on each product department separately. This approach leads to a very natural division of the data set that creates data sets of different sizes. To be able to analyze the dependency of the number of iterations on the length of the time series, we run DBA on subsequences of input sequences with different lengths. We explicitly choose for each input sequence, one subsequence of consecutive days. The values for the length of the subsequences are taken exponentially growing as $15 \cdot 2^i$ for $0 \leq i \leq 7$ (15, 30, 60, 120, 240, 480, 960, 1920). We run DBA ten times for each fixed value of the length, where we draw the starting day of the input subseries uniformly at random (same starting day for all series in the same run). For each run of DBA, we track the number of iterations.

**Experiment 3: Dependency on the length $k$ of the output center**

For the same reasons as in Experiment 2, we run the experiment on different Product departments separately. For the sake of a clearer presentation, we restrict ourselves to the product departments Foods 1, Foods 2 and Foods 3. As input sequences, we take all time series that correspond to the respective product department. For each department, we run DBA for several lengths of the output center, where we test each value of the length ten times. The values for the length of the center are taken again as $15 \cdot 2^i$ for $0 \leq i \leq 7$ (15, 30, 60, 120, 240, 480, 960, 1920). For each run of DBA, we track the number of iterations.

## 6.6 Results of the experiments

In this section, we state the results of the described experiments. More detailed tables and box plots of the exact results can be found in Appendix 6.8. The results of Experiment 1, 2 and 3 are depicted in Figure 6.2, 6.3 and 6.4. For Experiment 1, the relation between the average number of iterations and the number of the input point sequences is shown in Figure 6.2. For Experiment 2, the relation between the average number of iterations and the length of the input point sequences is shown in Figure 6.3 and for Experiment 3, the relation between the average number of iterations and the length of the output center is shown in Figure 6.4. As we can see in all three figures, the dependency on each parameter seems to be sublinear. Since the function graphs appear linear in the log-log plots, the dependencies seem to be polynomial for exponents smaller 1. Estimations for the

exponents which were computed with linear regression on the points in the log-log-plots are given in Table 6.3, 6.4 and 6.5.

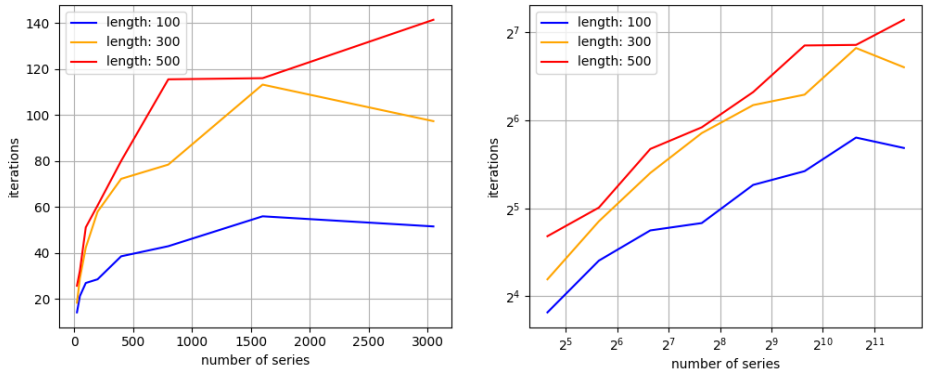


Figure 6.2: Depiction of the average number of iterations of the DBA algorithm with respect to the number of time series in the chosen sets. Each function graph corresponds to a fixed length of all time series in the sets. The left graphic uses normal scales and the right graphic uses logarithmic scales on both axes.

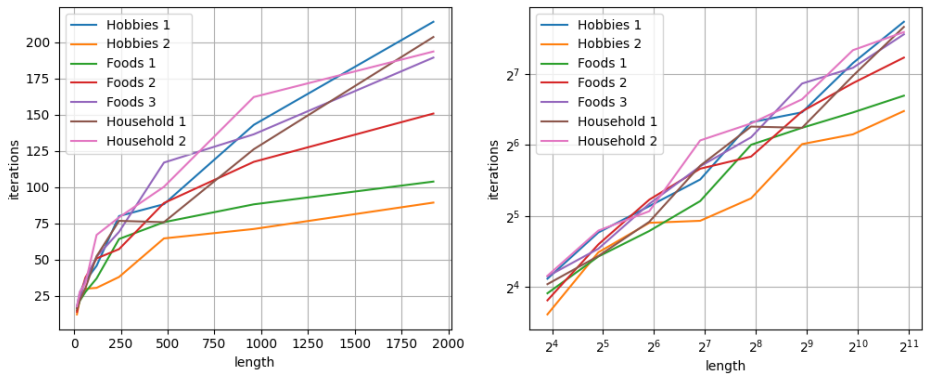


Figure 6.3: Depiction of the average number of iterations of the DBA algorithm with respect to the length of the series in the chosen sets. Each function graph corresponds to one product department. The left graphic uses normal scales and the right graphic uses logarithmic scales on both axes.

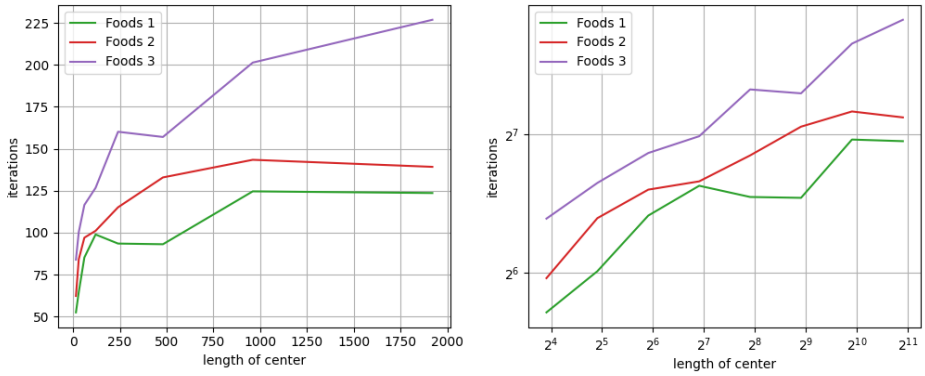


Figure 6.4: Depiction of the average number of iterations of the DBA algorithm with respect to the length of the center point sequence. Each function graph corresponds to one product department. The left graphic uses normal scales and the right graphic uses logarithmic scales on both axes.

| Length | 100 | 200 | 300 |
|---|---|---|---|
| Exponent | 0.27 | 0.35 | 0.36 |

Table 6.3: Experiment 1: Estimations for the exponents of the dependency on the size of the input point sequences for each fixed length of the input point sequences.

| Dep. | Hobbies 1 | Hobbies 2 | Foods 1 | Foods 2 | Foods 3 | Househ. 1 | Househ. 2 |
|---|---|---|---|---|---|---|---|
| Exp. | 0.50 | 0.38 | 0.42 | 0.47 | 0.50 | 0.51 | 0.50 |

Table 6.4: Experiment 2: Estimations for the exponents of the dependency on the length of the output center for each Department.

| Department | Foods 1 | Foods 2 | Foods 3 |
|---|---|---|---|
| Exponent | 0.16 | 0.16 | 0.20 |

Table 6.5: Experiment 3: Estimations for the exponents of the dependency on the length of the input point sequences for each Department.

## 6.7 Experiments on the UCR Time Series Classification Archive

The UCR Time Series Classification Archive contains many data sets from different areas. For our experiments, we selected the same subset of data sets as Schultz and Jain [115]. For each data set, we merge the training and test sets into a single data set. Each data set is divided into several classes and only contains time series of identical length. In Table 6.6, we display the name, the number $c$ of classes, the total number $n$ of time series, the average number $n/c$ of time series per cluster and the length $m$ of each data set.

| Dataset | $c$ | $n$ | $n/c$ | $m$ | $\mu_1$ | $\sigma_1^2$ | $\mu_2$ | $\sigma_2^2$ |
|---|---|---|---|---|---|---|---|---|
| 50words | 50 | 905 | 18.10 | 270 | 50.66 | 49.56 | 45.41 | 45.86 |
| Adiac | 37 | 781 | 21.11 | 176 | 3.24 | 3.93 | 38.05 | 9.45 |
| Beef | 5 | 60 | 12.00 | 470 | 88.60 | 61.83 | 85.6 | 53.02 |
| CBF | 3 | 930 | 310.00 | 128 | 98.00 | 39.30 | 67.77 | 18.93 |
| ChlorineConc. | 3 | 4307 | 1435.67 | 166 | 78.67 | 10.34 | 96.67 | 38.42 |
| Coffee | 2 | 56 | 28.00 | 286 | 37.50 | 8.50 | 40.15 | 6.37 |
| ECG200 | 2 | 200 | 100.00 | 96 | 65.50 | 12.50 | 53.40 | 9.53 |
| ECG5000 | 5 | 5000 | 1000.00 | 140 | 110.40 | 80.50 | 136.50 | 144.87 |
| ElectricDevices | 7 | 16637 | 2376.71 | 96 | 50.14 | 26.44 | 57.96 | 47.15 |
| FaceAll | 14 | 2250 | 160.71 | 131 | 40.43 | 17.88 | 67.29 | 28.81 |
| FaceFour | 4 | 112 | 28.00 | 350 | 29.50 | 9.96 | 33.45 | 7.89 |
| Fish | 7 | 350 | 50.00 | 463 | 93.57 | 33.67 | 77.84 | 23.20 |
| Gun_Point | 2 | 200 | 100.00 | 150 | 46.00 | 4.00 | 62.75 | 22.29 |
| Lighting2 | 2 | 121 | 60.50 | 637 | 80.00 | 8.00 | 47.85 | 10.36 |
| Lighting7 | 7 | 143 | 20.43 | 319 | 30.29 | 24.52 | 23.64 | 8.12 |
| OliveOil | 4 | 60 | 15.00 | 570 | 34.75 | 21.09 | 49.90 | 20.10 |
| OSULeaf | 6 | 441 | 73.50 | 427 | 164.00 | 54.99 | 126.63 | 120.17 |
| PhalangesOutl. | 2 | 2658 | 1329.00 | 80 | 57.50 | 3.50 | 192.00 | 74.70 |
| SwedishLeaf | 15 | 1125 | 75.00 | 128 | 28.34 | 11.01 | 55.82 | 15.36 |
| synthetic_control | 6 | 600 | 100.00 | 60 | 32.00 | 8.52 | 31.05 | 10.45 |
| Trace | 4 | 200 | 50.00 | 275 | 29.00 | 4.06 | 54.55 | 25.66 |
| Two_Patterns | 4 | 5000 | 1250.00 | 128 | 104.25 | 23.59 | 81.53 | 28.97 |
| wafer | 2 | 7164 | 3582.00 | 152 | 67.5 | 4.50 | 52.05 | 7.63 |
| yoga | 2 | 3300 | 1650.00 | 426 | 672.5 | 197.5 | 563.95 | 292.07 |

Table 6.6: Characteristics of the UCR TS datasets and the results for applying DBA. Here, $c$ stands for the number of classes, $n$ for the number of series, $n/c$ is therefore the average number of series in a class and $m$ is the length of each time series. The value $\mu_i$ stands for the average number of iterations and $\sigma_i^2$ stands for the variance in the number of iterations over all classes. Here the index $i = 1$ stands for medoid initialization and $i = 2$ stands for random initialization.

We run the DBA algorithm on the chosen data sets for two different initialization methods (medoid and random) and analyze the dependency of the number of iterations on the number and length of the series. As input, we take each class of the data set separately and track the number of iterations needed by the DBA algorithm to converge. In the end, we compute the average number of iterations over all classes and the corresponding variances.

For the medoid initialization, we take the input point sequence as the starting center that minimizes the DTW distance to the other input point sequences. For the random initialization, we construct the starting center from a valid assignment of the input sequences. The valid assignment is created by a combination of random walks consisting of one walk per input sequence. Such a random walk starts with assigning the first point of the input series to the first point of the center point sequence and then chooses the next assignment by either going to the next point on the input series while staying at the same point of the center point sequence, going to the next point on the center point sequence while staying at the same point of the input series or moving to the next point

on both series. Each option is taken with probability 1/3. It can also happen, that there is only one valid option remaining, which is then always chosen. After the random assignment is computed, the center points are chosen as the arithmetic means of the assigned points from the input series. In the case of the random initialization, we run the DBA algorithm 10 times per cluster and calculate the average number of iterations over all runs on all clusters.

For each data set, we depict the tracked average number of iterations of the DBA algorithm and the corresponding variances in Table 6.6 and Figure 6.5. The figures do not suggest any clear dependency of the number of iterations on the number or length of the input point sequences for any of the two initialization methods. We conjecture that this result can be attributed to the heterogeneity of the data sets.

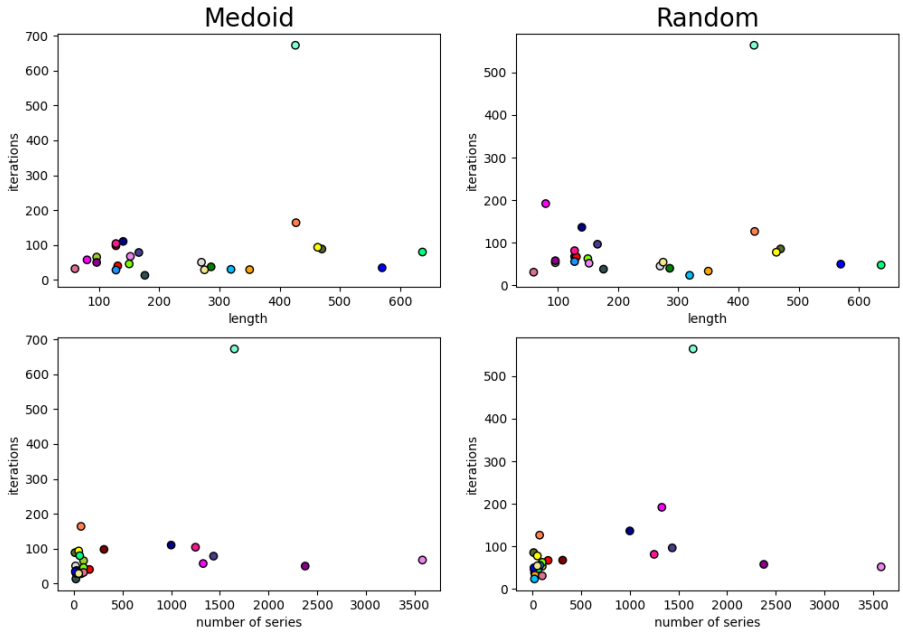


Figure 6.5: Depiction of the average number of iterations of DBA with respect to length and number of series in the corresponding data sets. Each color corresponds to one data set of the UCR Time Series Clasification Archive. The left side shows the results for the medoid initialization and the right side shows the results for the random initialization.

## 6.8 Data of the experiments on the M5 data set

This chapter includes the detailed data of the experiments on the M5 data set. The data of Experiment 1 about the Dependency on the number of input point sequences is depicted in Table 6.7, Table 6.8 and Figure 6.7, the data of Experiment 2 about the Dependency on the length of the input point sequences is depicted in Table 6.9, Table 6.10 and Figure 6.7 and the data of Experiment 3 about the Dependency on the length of the output center is depicted in Table 6.11, Table 6.12 and Figure 6.8.

| Mean of the number of iterations (Experiment 1) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Sequences Length | 25 | 50 | 100 | 200 | 400 | 800 | 1600 | 3149 |
| 100 | 14.1 | 21.2 | 26.9 | 28.5 | 38.5 | 42.9 | 55.9 | 51.5 |
| 300 | 18.3 | 28.9 | 42.3 | 57.9 | 72.2 | 78.4 | 113.2 | 97.3 |
| 500 | 25.7 | 32.2 | 51.1 | 60.6 | 80 | 115.5 | 116 | 141.4 |

Table 6.7: Mean of the number of iterations with respect to the number of input point sequences. Each row corresponds to one fixed length of the input sequences

| Variance of the number of iterations (Experiment 1) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Sequences Length | 25 | 50 | 100 | 200 | 400 | 800 | 1600 | 3149 |
| 100 | 2.88 | 4.73 | 7.53 | 5.41 | 8.67 | 11.94 | 22.87 | 17.25 |
| 300 | 5.64 | 6.99 | 8.82 | 14.02 | 25.72 | 26.75 | 57.45 | 29.33 |
| 500 | 8.94 | 6.90 | 7.62 | 17.49 | 21.67 | 34.80 | 34.35 | 44.50 |

Table 6.8: Variance of the number of iterations with respect to the number of input point sequences. Each row corresponds to one fixed length of the input sequences

| Mean of the number of iterations (Experiment 2) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Length Department | 15 | 30 | 60 | 120 | 240 | 480 | 960 | 1920 |
| Hobbies 1 | 17.3 | 27.2 | 35.2 | 45.7 | 80 | 88.2 | 142.9 | 214 |
| Hobbies 2 | 12.2 | 22.4 | 29.9 | 30.5 | 38 | 64.6 | 71.1 | 89.3 |
| Foods 1 | 15 | 21.5 | 27.6 | 37 | 64.2 | 75.8 | 88.1 | 103.8 |
| Foods 2 | 14 | 24.2 | 37.5 | 50.8 | 57.2 | 89 | 117.4 | 150.7 |
| Foods 3 | 17.7 | 23.2 | 36 | 52 | 69.1 | 116.9 | 136.4 | 189.3 |
| Household 1 | 16.4 | 21.6 | 30.2 | 52.3 | 76.7 | 75.8 | 126.1 | 203.5 |
| Household 2 | 17.8 | 27.7 | 33.5 | 67 | 79.1 | 100.2 | 162.1 | 193.5 |

Table 6.9: Mean of the number of iterations with respect to the length of the input point sequences. Each row corresponds to one product department.

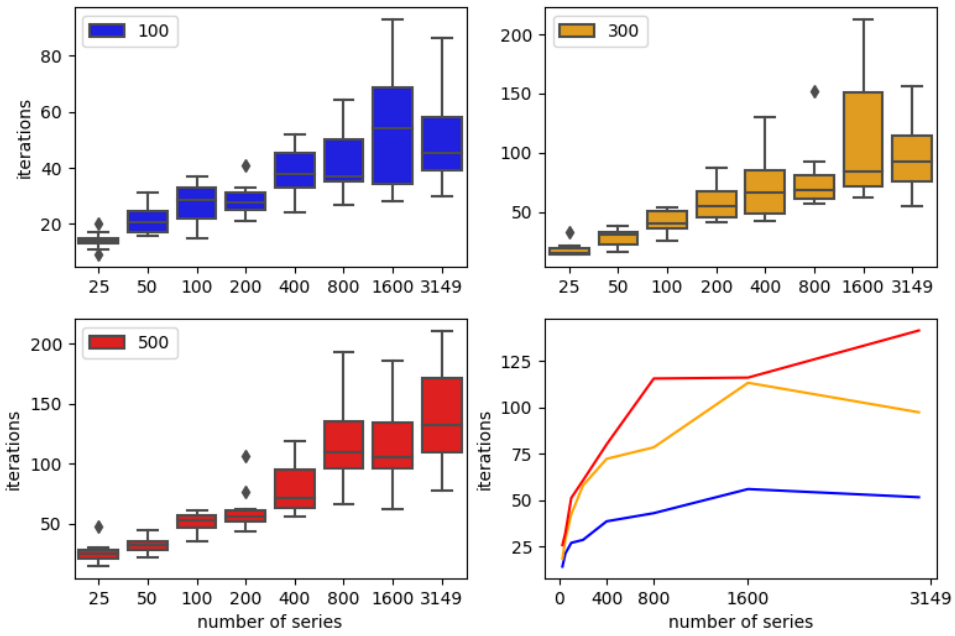


Figure 6.6: Experiment 1: Boxplots of the number of iterations for each department and each number of input sequences. Each graphic corresponds to one fixed length of the input sequences and has a logarithmic scale on the horizontal axis. Only the last graphic has regular scale on the horizontal axis. It depicts the mean of the number of iterations per number of input sequences for each of the departments.

| Variance of the number of iterations (Experiment 2) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Length / Department | 15 | 30 | 60 | 120 | 240 | 480 | 960 | 1920 |
| Hobbies 1 | 3.38 | 11.12 | 12.66 | 15.47 | 30.43 | 20.36 | 44.97 | 70.87 |
| Hobbies 2 | 2.56 | 9.31 | 18.56 | 8.09 | 11.06 | 12.86 | 7.99 | 28.43 |
| Foods 1 | 4.47 | 5.57 | 10.34 | 12.17 | 15.07 | 21.89 | 10.19 | 27.32 |
| Foods 2 | 3.16 | 7.00 | 11.81 | 19.72 | 14.61 | 27.66 | 20.19 | 54.80 |
| Foods 3 | 6.42 | 6.08 | 11.33 | 19.44 | 37.63 | 45.65 | 41.44 | 33.05 |
| Household 1 | 3.41 | 3.53 | 8.22 | 28.78 | 32.95 | 19.1405 | 47.66 | 68.82 |
| Household 2 | 6.87 | 12.1 | 9.57 | 30.94 | 27.06 | 21.50 | 40.80 | 51.16 |

Table 6.10: Variance of the number of iterations with respect to the length of the input point sequences. Each row corresponds to one product department

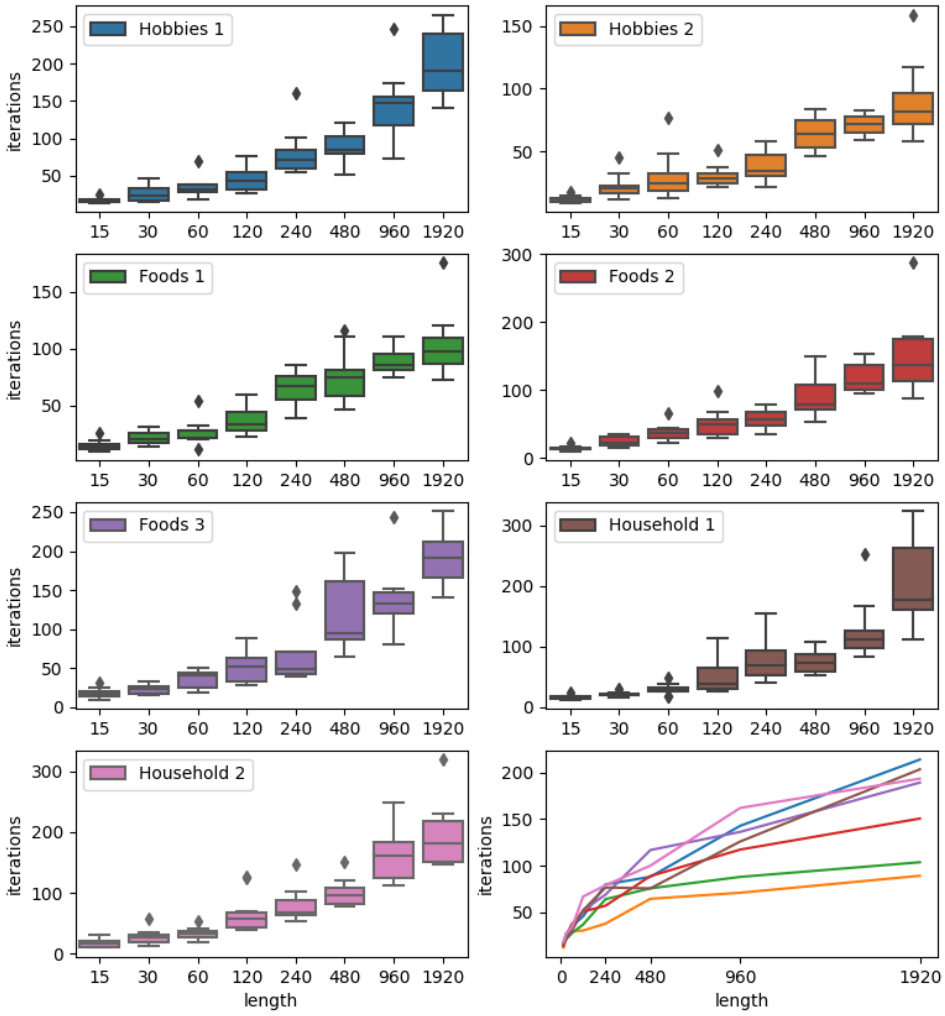


Figure 6.7: Experiment 2: Boxplots of the number of iterations for each department and each length of input sequences. Each graphic corresponds to one product department and has a logarithmic scale on the horizontal axis. Only the last graphic has regular scale on the horizontal axis. It depicts the mean of the number of iterations per length for each of the departments.

| Mean of the number of iterations (Experiment 3) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Length of Center<br>Department | 15 | 30 | 60 | 120 | 240 | 480 | 960 | 1920 |
| Foods 1 | 52.5 | 64.5 | 85.2 | 98.9 | 93.5 | 93.1 | 124.6 | 123.6 |
| Foods 2 | 62.3 | 84.1 | 97 | 101.1 | 115.1 | 132.9 | 143.4 | 139.2 |
| Foods 3 | 83.9 | 100.3 | 116.5 | 126.7 | 160.1 | 157 | 201.3 | 226.8 |

Table 6.11: Mean of the number of iterations with respect to the length of the output center. Each row corresponds to one product department.

| Variance of the number of iterations (Experiment 3) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Length of Center<br>Department | 15 | 30 | 60 | 120 | 240 | 480 | 960 | 1920 |
| Foods 1 | 6.07 | 11.60 | 15.90 | 17.47 | 13.07 | 14.01 | 20.11 | 36.69 |
| Foods 2 | 11.1 | 19.24 | 17.78 | 13.86 | 28.68 | 51.42 | 40.39 | 32.44 |
| Foods 3 | 25.78 | 13.33 | 27.15 | 25.08 | 50.25 | 36.90 | 53.62 | 66.71 |

Table 6.12: Variance of the number of iterations with respect to the length of the output center. Each row corresponds to one product department

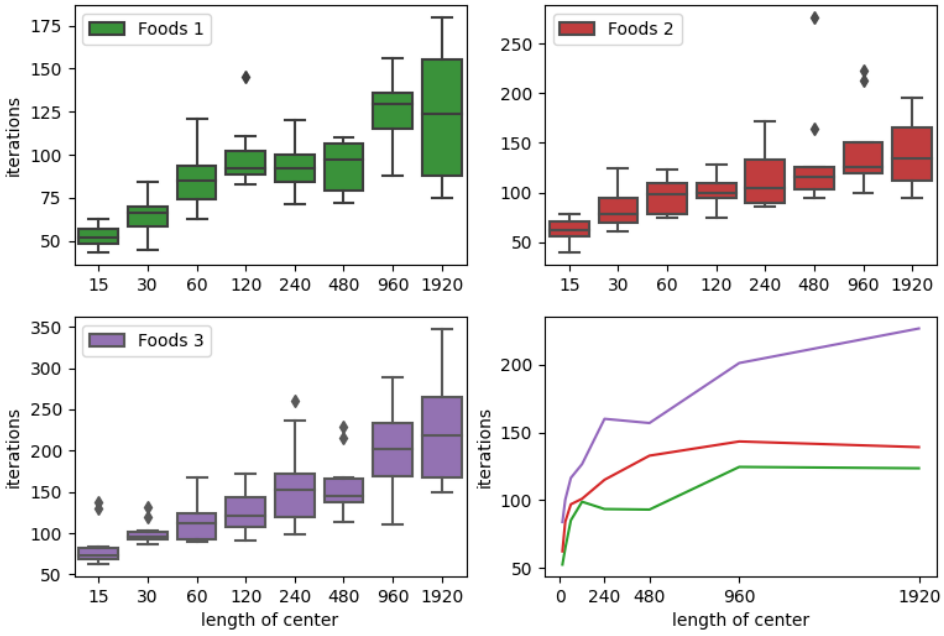


Figure 6.8: Experiment 3: Boxplots of the number of iterations for each department and each length of the output center. Each graphic corresponds to one product department and has a logarithmic scale on the horizontal axis. Only the last graphic has regular scale on the horizontal axis. It depicts the mean of the number of iterations per fixed length of the output center for each of the departments.

# Chapter 7

# Conclusions

In this thesis, we have studied the clustering of polygonal curves and polygonal regions in various ways. We have analyzed the VC-dimension and shattering dimension of elastic distance measures, discussed subtrajectory clustering under the Fréchet distance and investigated the number of iterations of the DBA algorithm. In the case of our VC-dimension bounds, we considered range spaces of balls around polygonal curves and regions with respect to variations of the Hausdorff distance, the Fréchet distance and the dynamic time warping distance (DTW). We showed bounds that are tight for each parameter separately in the case of polygonal curves for the Hausdorff distance and the Fréchet distance. We showed similar bounds for the Hausdorff distance of polygonal regions. Our techniques also directly implied VC-dimension bounds for DTW and the average Hausdorff distance. In the context of subtrajectory clustering, we developed bicriteria approximation algorithms with approximation factors in the size and radius of the solution. For the number of iterations of the DBA algorithm, we showed lower and upper bounds in the worst-case that are exponential in the length of the center sequence and a smoothed upper bound that is exponential in $\frac{n}{d}$, where $n$ is the number of input sequences and $d$ is the ambient dimension. We complemented these findings with experiments that show fast convergence with a growth rate that is sublinear in all considered parameters. In the following, we discuss open problems and further research directions.

## 7.1 VC-dimension and shattering dimension of elastic distance measures

We considered bounds on the VC-dimension of balls under various distance measures. In the cases where we have no tight bounds, it is an interesting direction to study lower bounds and investigate if our upper bounds are already tight or if they can still be improved. Another natural question is whether our techniques can be applied to other distance measures or other geometric objects like smoothed curves or unions of higher-dimensional polytopes as well. Since the techniques only require a distance query to be answered by simple predicates, it seems reasonable that such predicates can also be found for other distance measures or other geometric objects. A different direction would be to consider applications that are based on the studied range spaces. The VC-dimension results have already been used to improve algorithms for subtrajectory clustering in this thesis. It would be an interesting direction to consider problems in the areas of prediction,

density estimation, classification or hot spot detection as well. Even without using the VC-dimension directly, our techniques that split distance queries into predicates, which just depend on sign values of polynomials, might find applications in these areas. The arrangement of zero sets of the polynomials captures the structure of the distance query well and has already been utilized successfully in the case of the Fréchet distance for range searching, nearest neighbor classification, distance oracles and curve simplification by Cheng and Huang [43]. The extension of these results to the other distance measures discussed in this thesis and to polygonal regions in the case of the Hausdorff distance seems to be a straightforward implication of our results from Chapter 3. It would be interesting to investigate in which other of the mentioned areas the arrangement proves to be useful.

## 7.2 Subtrajectory Clustering

There are many variants of subtrajectory clustering that arise from application-specific considerations, see also the discussion of related work in Section 1.2. We expect that our general approach and our problem definition can be applied to a lot of these variants. For example, all of our algorithms can be easily extended to the setting of multiple input curves. We mention some other variants that we find interesting.

(1) Outputting a graph: The output of our algorithm is a set of center curves. In some applications, such as map construction, we may prefer the output to be a geometric graph. This can be easily obtained by connecting the center curves to form a geometric graph using additional edges where the input trajectory moves from one cluster to the next. How to do this optimally would be a subject for future research.

(2) Covering with gaps: One might be interested in a problem variant where not the entire curve but only a certain fraction of it needs to be covered. It would be interesting to analyze our techniques in this setting or related settings, like the facility location version of the problem in [3].

(3) Input curves: In our work, we assume that our input curves are given in the form of polygonal curves. However, it is conceivable that our general approach to the discrete problem still works if the input is given in the form of piecewise polynomial curves with breakpoints; again, we leave this to future work.

(4) Other distance measures: Similarly, we think that our general approach is still applicable if the Fréchet distance is replaced by some other distance measure that satisfies the triangle inequality. It would be interesting to study such cases in more detail or consider distance measures like dynamic time warping that do not satisfy the triangle inequality.

(5) Maximum coverage problem: The maximum coverage problem is a closely related problem that asks to maximize the coverage for a given fixed number of center curves. In [26], we show that our techniques can also be applied in this case to directly yield an approximate solution. It would be interesting to study this problem in more detail and try to improve the results.

As mentioned earlier, it may be tempting to allow for center curves of arbitrary complexity. However, this would lead to the trivial solution of the curve $P$ being an

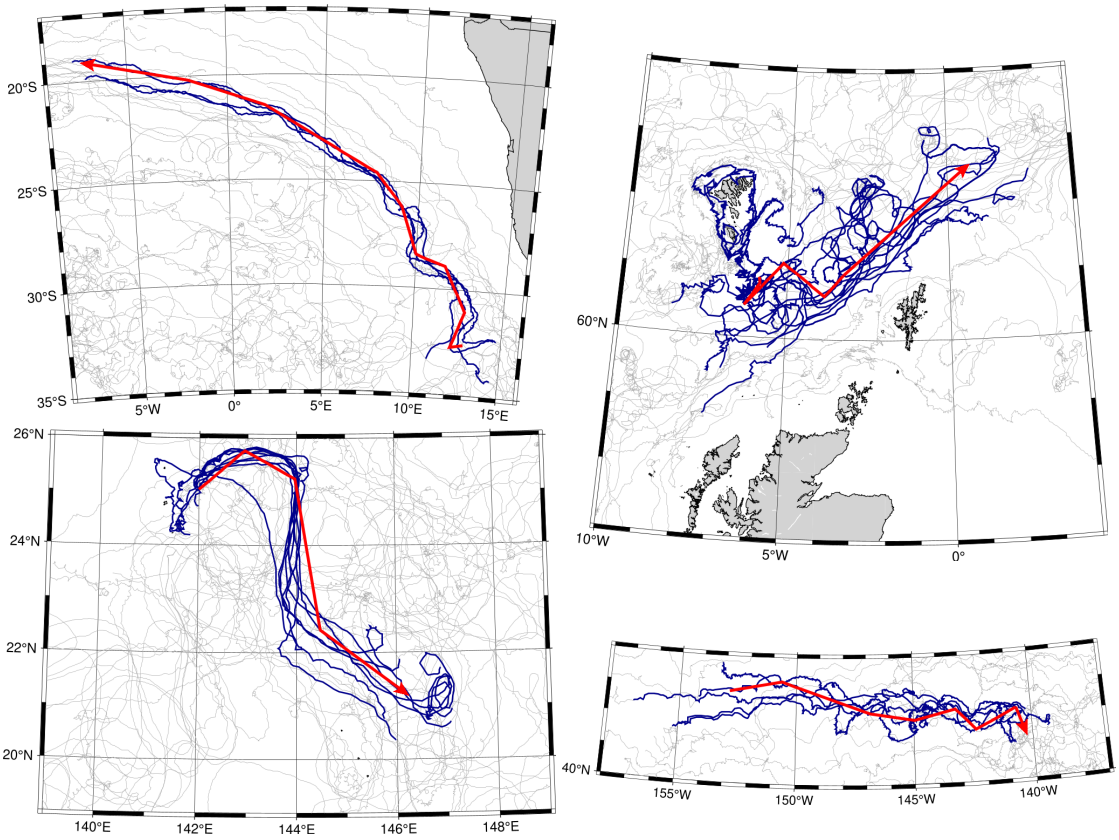


Figure 7.1: Ocean surface drifters from the NOAA Global Drifter Program [99]: 4 Clusters of Subtrajectories with their corresponding center curves that were selected in the first 10 selections of the greedy algorithm in [47]. Source of picture: [47]

optimal center curve. In any case, we think that some form of controlled regularization is necessary in the problem definition. Generally, it would also be interesting to further study the original problem and try to improve approximation factors, running time and space or find some hardness of approximation results. Another promising research direction is to apply our developed algorithms to real-world data. Conradi and Driemel [47] already applied some of the techniques from our work [25] to perform subtrajectory clustering on human motion data and ocean drifter data (see also Figure 7.1). It would be interesting to see if our other techniques, like the implicit weight update approach for the multiplicative weight update method, would also work well on real-world data.

## 7.3 The number of iterations of the DBA algorithm

Our experimental results for the number of iterations of the DBA algorithm show a gap between the theoretical bounds we were able to prove in the model of smoothed analysis and the number of steps DBA takes on real-world data. This suggests two directions for further research: (1) developing refined worst-case bounds under realistic input models, and (2) sharpening the obtained smoothed analysis estimates to better reflect practical performance.

The structure of DBA is very similar to the classical $k$-means algorithm. One would

naturally expect the techniques from the analysis of the $k$-means algorithm to be useful for the analysis of DBA. This was true for the geometric ideas of Vattani in the case of proving lower bounds. However, surprisingly, the techniques for the smoothed analysis of $k$-means were not effective in the smoothed analysis of DBA. The main difference seems to be the following: In every step of the $k$-means algorithm, i.e. every change of assignment for a point from one center to another, the distance decreases. This gives $k$-means a very monotonic behaviour. In DBA we use dynamic time warping distance (DTW) and the behaviour is not necessarily monotonic: A change in the assignment of a point $x$ in point sequence $\gamma_1$ can *increase* the distance of the point $x$ while the total DTW distance between point sequence $\gamma_1$ and the mean point sequence decreases. This behaviour seems to rule out the usage of most useful techniques from [16]. Instead, we manage to exploit the specific properties of DBA, as it requires the assignments between the points of an input sequence and the mean point sequence to respect the ordering along the mean point sequence and along the input sequence. In the potential function argument that we use to bound the number of iterations, all of the cost reduction comes from the movement of the center vertices in individual update steps of the algorithm. It would be interesting to see if there is a lucrative way to use the cost reduction in the assignment step of the algorithm or by analyzing the cost reduction of multiple consecutive iterations together. This has been shown to be difficult because of the non-monotonic and global behaviour of the assignment in comparison to the monotonic and local behaviour of the assignment of the $k$-means algorithm.

# Bibliography


[1] Peyman Afshani and Anne Driemel. On the complexity of range searching among curves. *CoRR*, 2017. doi:10.48550/ARXIV.1707.04789.

[2] 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, SODA*, 2018. doi:10.1137/1.9781611975031.58.

[3] 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. doi:10.1145/3196959.3196972.

[4] Pankaj K. Agarwal, Sariel Har-Peled, Nabil H. Mustafa, and Yusu Wang. Near-Linear Time Approximation Algorithms for Curve Simplification. *Algorithmica*, 42(3):203–219, 2005. doi:10.1007/s00453-005-1165-y.

[5] Ravi Aggarwal, Viknesh Sounderajah, Guy Martin, Daniel SW Ting, Alan Karthikesalingam, Dominic King, Hutan Ashrafian, and Ara Darzi. Diagnostic accuracy of deep learning in medical imaging: a systematic review and meta-analysis. *NPJ digital medicine*, 4(1):65, 2021. doi:10.1038/s41746-021-00438-z.

[6] Hugo Akitaya, Frederik Brüning, Erin Chambers, and Anne Driemel. Covering a Curve with Subtrajectories. *CoRR*, 2021. doi:10.48550/ARXIV.2103.06040.

[7] Hugo Akitaya, Frederik Brüning, Erin Chambers, and Anne Driemel. Covering a Curve with Subtrajectories. In *37th European Workshop on Computational Geometry (EuroCG 2021)*, 2021.

[8] 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. doi:10.57717/cgt.v2i1.7.

[9] Ehab A. AlBadawy, Ashirbani Saha, and Maciej A. Mazurowski. Deep learning for segmentation of brain tumors: Impact of cross-institutional training and testing. *Medical Physics*, 45(3):1150–1158, 2018. doi:10.1002/mp.12752.

[10] Noga Alon. Tools from higher algebra. *Handbook of combinatorics*, 2:1749–1783, 1996.

[11] Helmut Alt, Oswin Aichholzer, and Günter Rote. Matching shapes with a reference point. In *Proceedings of the Tenth Annual Symposium on Computational Geometry*, SCG '94, pages 85–92, 1994. doi:10.1145/177424.177555.

[12] Helmut Alt, Bernd Behrends, and Johannes Blömer. Approximate matching of polygonal shapes. In *Proceedings of the seventh annual symposium on Computational geometry*, pages 186–193, 1991. doi:10.1145/109648.109669.

[13] Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5:75–91, 1995. doi:10.1142/S0218195995000064.

[14] Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999. doi:10.1017/CBO9780511624216.

[15] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012. doi:10.4086/toc.2012.v008a006.

[16] David Arthur, Bodo Manthey, and Heiko Röglin. Smoothed Analysis of the k-Means Method. *J. ACM*, 58(5), 2011. doi:10.1145/2027216.2027217.

[17] David Arthur and Sergei Vassilvitskii. How Slow is the k-Means Method? In *Proc. Symp. on Computational Geometry*, SCG '06, pages 144–153, 2006. doi:10.1145/1137856.1137880.

[18] Shai Ben-David and Michael Lindenbaum. Localization vs. Identification of Semi-Algebraic Sets. In *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory, COLT 1993*, pages 327–336, 1993. doi:10.1145/168304.168364.

[19] Donald J. Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *Knowledge Discovery in Databases: Papers from the 1994 AAAI Workshop*, pages 359–370. AAAI Press, 1994.

[20] Martin Breunig, Patrick Erik Bradley, Markus Jahn, Paul Kuper, Nima Mazroob, Norbert Rösch, Mulhim Al-Doori, Emmanuel Stefanakis, and Mojgan Jadidi. Geospatial Data Management Research: Progress and Future Directions. *ISPRS International Journal of Geo-Information*, 9(2), 2020. doi:10.3390/ijgi9020095.

[21] Markus Brill, Till Fluschnik, Vincent Froese, Brijnesh Jain, Rolf Niedermeier, and David Schultz. Exact mean computation in dynamic time warping spaces. *Data Mining and Knowledge Discovery*, 33(1):252–291, 2019. doi:10.1007/S10618-018-0604-8.

[22] Hervé Brönnimann, Bernard Chazelle, and Jirí Matousek. Product Range Spaces, Sensitive Sampling, and Derandomization. *SIAM J. Comput.*, 28(5):1552–1575, 1999. doi:10.1137/S0097539796260321.

[23] Hervé Brönnimann and Michael T Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete & Computational Geometry*, 14(4):463–479, 1995. doi:10.1007/BF02570718.

[24] Frederik Brüning. DBA implementation. *GitHub*, 2022. https://github.com/FrederikBruening/DBA.

[25] 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)*, volume 244, pages 28:1–28:16, 2022. doi:10.4230/LIPIcs.ESA.2022.28.

[26] Frederik Brüning, Jacobus Conradi, and Anne Driemel. Faster Approximate Covering of Subcurves under the Fréchet Distance. *CoRR*, 2022. doi:10.48550/ARXIV.2204.09949.

[27] Frederik Brüning and Anne Driemel. Simplified and Improved Bounds on the VC-Dimension for Elastic Distance Measures. *CoRR*, 2023. doi:10.48550/ARXIV.2308.05998.

[28] Frederik Brüning and Anne Driemel. Simplified and Improved Bounds on the VC-Dimension for Elastic Distance Measures. In *40th European Workshop on Computational Geometry (EuroCG 2024)*, 2024.

[29] Frederik Brüning, Anne Driemel, Alperen Ergür, and Heiko Röglin. On the number of iterations of the DBA algorithm. In *39th European Workshop on Computational Geometry (EuroCG 2023)*, 2023.

[30] Frederik Brüning, Anne Driemel, Alperen Ergür, and Heiko Röglin. On the number of iterations of the DBA algorithm. In *Proceedings of the 2024 SIAM International Conference on Data Mining (SDM)*, 2024. doi:10.1137/1.9781611978032.20.

[31] Frederik Brüning, Anne Driemel, Alperen Ergür, and Heiko Röglin. On the number of iterations of the DBA algorithm. *CoRR*, 2024. doi:10.48550/ARXIV.2401.05841.

[32] 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*, SIGSPATIAL '17, 2017. doi:10.1145/3139958.3139964.

[33] 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. doi:10.1145/3423334.3431451.

[34] 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. doi:10.1142/S0218195911003652.

[35] 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, SODA 2019*, pages 2922–2938, 2019. doi:10.1137/1.9781611975482.181.

[36] Kevin Buchin, Anne Driemel, and Martijn Struijs. On the Hardness of Computing an Average Curve. In *17th Scandinavian Symp. and Workshops on Algorithm Theory, SWAT 2020*, volume 162, pages 19:1–19:19, 2020. doi:10.4230/LIPIcs.SWAT.2020.19.

[37] Maike Buchin, Anne Driemel, and Dennis Rohde. Approximating (k, $l$)-Median Clustering for Polygonal Curves. *ACM Trans. Algorithms*, 19(1):4:1–4:32, 2023. doi:10.1145/3559764.

[38] Maike Buchin, Bernhard Kilgus, and Andrea Kölzsch. Group diagrams for representing trajectories. *International Journal of Geographical Information Science*, 34(12):2401–2433, 2020. doi:10.1080/13658816.2019.1684498.

[39] 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 2022*, pages 167–180, 2022. doi:10.1007/978-3-030-95018-7_14.

[40] Maike Buchin and Carola Wenk. Inferring movement patterns from geometric similarity. *Journal of Spatial Information Science*, 21(1):63–69, 2020. doi:10.5311/JOSIS.2020.21.724.

[41] Laurent Bulteau, Vincent Froese, and Rolf Niedermeier. Tight hardness results for consensus problems on circular strings and time series. *SIAM Journal on Discrete Mathematics*, 34(3):1854–1883, 2020. doi:10.1137/19M1255781.

[42] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. The UCR time series classification archive, 2015. www.cs.ucr.edu/~eamonn/time_series_data/.

[43] Siu-Wing Cheng and Haoqiang Huang. Solving Fréchet Distance Problems by Algebraic Geometric Methods. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4502–4513, 2024. doi:10.1137/1.9781611977912.158.

[44] Vasek Chvatal. A Greedy Heuristic for the Set-Covering Problem. *Mathematics of Operations Research*, 4(3):233–235, 1979. doi:10.1287/MOOR.4.3.233.

[45] Kenneth L. Clarkson. Algorithms for polytope covering and approximation. In *Algorithms and Data Structures*, pages 246–252, 1993. doi:10.1007/3-540-57155-8_252.

[46] Kenneth L Clarkson. Las Vegas algorithms for linear and integer programming when the dimension is small. *Journal of the ACM (JACM)*, 42(2):488–499, 1995. doi:10.1145/201019.201036.

[47] Jacobus Conradi and Anne Driemel. Finding complex patterns in trajectory data via geometric set cover. *CoRR*, 2023. doi:10.48550/ARXIV.2308.14865.

[48] 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)*, volume 293, pages 42:1–42:17, 2024. doi:10.4230/LIPIcs.SoCG.2024.42.

[49] Felipe Cucker, Alperen A. Ergür, and Josue Tonelli-Cueto. Plantinga-Vegter Algorithm takes Average Polynomial Time. In *Proceedings of the 2019 on International Symposium on Symbolic and Algebraic Computation*, pages 114–121, 2019. doi:10.1145/3326229.3326252.

[50] Mark de Berg, Atlas F. Cook IV, and Joachim Gudmundsson. Fast Fréchet queries. *Computational Geometry*, 46(6):747–755, 2013. doi:10.1016/J.COMGEO.2012.11.006.

[51] Lex de Kogel, Marc van Kreveld, and Jordi L. Vermeulen. Abstract Morphing Using the Hausdorff Distance and Voronoi Diagrams. In *30th Annual European Symposium on Algorithms (ESA 2022)*, volume 244, pages 74:1–74:16, 2022. doi:10.4230/LIPIcs.ESA.2022.74.

[52] Alexander De Luca, Alina Hang, Frederik Brudy, Christian Lindner, and Heinrich Hussmann. Touch me once and i know it's you! Implicit authentication based on touch screen patterns. *Conference on Human Factors in Computing Systems - Proceedings*, 2012. doi:10.1145/2207676.2208544.

[53] Michel Marie Deza and Elena Deza. *Encyclopedia of Distances*. Springer Berlin Heidelberg, 2009. doi:10.1007/978-3-642-00234-2_1.

[54] Anders Drachen, Rafet Sifa, Christian Bauckhage, and Christian Thurau. Guns, swords and data: Clustering of player behavior in computer games in the wild. In *2012 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 163–170, 2012. doi:10.1109/CIG.2012.6374152.

[55] Anne Driemel and Sariel Har-Peled. Jaywalking Your Dog: Computing the Fréchet Distance with Shortcuts. *SIAM Journal on Computing*, 42(5):1830–1866, 2013. doi:10.1137/120865112.

[56] Anne Driemel, Amer Krivosija, and Christian Sohler. Clustering time series under the Fréchet distance. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016*, pages 766–785, 2016. doi:10.1137/1.9781611974331.ch55.

[57] 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. doi:10.1007/s00454-021-00318-z.

[58] Andrew T. Duchowski. A breadth-first survey of eye-tracking applications. *Behavior Research Methods, Instruments, & Computers*, 34(4):455–470, 2002. doi:10.3758/BF03195475.

[59] Thomas Eiter and Heikki Mannila. Computing discrete Fréchet distance. *Technical Report CD-TR 94/64*, 1994.

[60] Alperen A. Ergür, Grigoris Paouris, and J. Maurice Rojas. Smoothed analysis for the condition number of structured real polynomial systems. *Mathematics of Computation*, 2021. doi:10.1090/MCOM/3647.

[61] Munazza Fatima, Kara J. O'Keefe, Wenjia Wei, Sana Arshad, and Oliver Gruebner. Geospatial Analysis of COVID-19: A Scoping Review. *International Journal of Environmental Research and Public Health*, 18(5), 2021. doi:10.3390/ijerph18052336.

[62] Dan Feldman. Introduction to Core-sets: an Updated Survey. *CoRR*, 2020. doi:10.48550/ARXIV.2011.09384.

[63] 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*, STOC '11, pages 569–578, 2011. doi:10.1145/1993636.1993712.

[64] Germain Forestier, Florent Lalys, Laurent Riffaud, Brivael Trelhu, and Pierre Jannin. Classification of surgical processes using dynamic time warping. *Journal of biomedical informatics*, 45(2):255–264, 2012. doi:10.1016/j.jbi.2011.11.002.

[65] Maurice Fréchet. Sur quelques points du calcul fonctionnel. *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, 22(1):1–72, 1906. doi:10.1007/BF03018603.

[66] Dariu M. Gavrila and Larry S. Davis. Towards 3-D model-based tracking and recognition of human movement: a multi-view approach. In *International Workshop on Automatic Face- and Gesture-Recognition. IEEE*, pages 272–277, 1995.

[67] Tom Gerson and Thoralf Noack. Generating a Vessel Route Model from AIS Data Using the Fréchet Distance. In *Marine Traffic Engineering Conference 2022*, 2022.

[68] Paul Goldberg and Mark Jerrum. Bounding the Vapnik-Chervonenkis dimension of concept classes parameterized by real numbers. In *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory, COLT 1993*, pages 361–369, 1993. doi:10.1145/168304.168377.

[69] Paul Goldberg and Mark Jerrum. Bounding the Vapnik-Chervonenkis dimension of concept classes parameterized by real numbers. *Machine Learning*, 18:131–148, 1995. doi:10.1007/BF00993408.

[70] Sophie Goliber, Taryn Black, Ginny Catania, James M Lea, Helene Olsen, Daniel Cheng, Suzanne Bevan, Anders Bjørk, Charlie Bunce, Stephen Brough, et al. TermPicks: a century of Greenland glacier terminus data for use in scientific and machine learning applications. *The Cryosphere*, 16(8):3215–3233, 2022. doi:10.5194/tc-16-3215-2022.

[71] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985. doi:10.1016/0304-3975(85)90224-5.

[72] W. Eric L. Grimson. On the recognition of parameterized 2D objects. *International Journal of Computer Vision*, 2(4):353–372, 1989. doi:10.1007/BF00133555.

[73] Jeffrey P. Guenette, Nir Ben-Shlomo, Jagadeesan Jayender, Ravi Teja Seethamraju, V. Kimbrell, N.-A. Tran, Raymond Y Huang, C. J. Kim, J. I. Kass, C Eduardo Corrales, and Thomas C. Lee. MR Imaging of the Extracranial Facial Nerve with the CISS Sequence. *American Journal of Neuroradiology*, 40(11):1954–1959, 2019. doi:10.3174/ajnr.A6261.

[74] Sariel Har-Peled. *Geometric approximation algorithms*. American Mathematical Soc., 2011.

[75] Sariel Har-Peled and Bardia Sadri. How Fast Is the k-Means Method? *Algorithmica*, 41(3):185–202, 2005. doi:10.1007/s00453-004-1127-9.

[76] Sariel Har-Peled and Micha Sharir. Relative (p, $\epsilon$)-Approximations in Geometry. *Discrete & Computational Geometry*, 45(3):462–496, 2011. doi:10.1007/s00454-010-9248-1.

[77] Jan-Henrik Haunert and Alexander Wolff. Optimal and topologically safe simplification of building footprints. In *Proceedings of the 18th sigspatial international conference on advances in geographic information systems*, pages 192–201, 2010. doi:10.1145/1869790.1869819.

[78] Felix Hausdorff. *Grundzüge der Mengenlehre*, volume 7. von Veit, 1914.

[79] Felix Hausdorff. *Mengenlehre*, volume 7. Gruyter, 1927.

[80] David Haussler and Emo Welzl. Epsilon-nets and simplex range queries. *Discrete & Computational Geometry*, 2(2):127–151, 1987. doi:10.1007/BF02187876.

[81] Adam Hilbert, Vince I. Madai, Ela M. Akay, Orhun U. Aydin, Jonas Behland, Jan Sobesky, Ivana Galinovic, Ahmed A. Khalil, Abdel A. Taha, Jens Wuerfel, Petr Dusek, Thoralf Niendorf, Jochen B. Fiebach, Dietmar Frey, and Michelle Livne. Brave-net: Fully automated arterial brain vessel segmentation in patients with cerebrovascular disease. *Frontiers in Artificial Intelligence*, 3, 2020. doi:10.3389/frai.2020.552258.

[82] Kenneth Holmqvist. *Eye tracking: a comprehensive guide to methods and measures*. Oxford University Press, 2011.

[83] Daniel P. Huttenlocher, Gregory A. Klanderman, and William J. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, 1993. doi:10.1109/34.232073.

[84] Mary Inaba, Naoki Katoh, and Hiroshi Imai. Variance-based k-clustering algorithms by Voronoi diagrams and randomization. *IEICE Trans. on Information and Systems*, 83(6):1199–1206, 2000.

[85] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE transactions on pattern analysis and machine intelligence*, 36(7):1325–1339, 2013. doi:10.1109/TPAMI.2013.248.

[86] Deepti Joshi, Leen-Kiat Soh, and Ashok Samal. Redistricting Using Constrained Polygonal Clustering. *IEEE Transactions on Knowledge and Data Engineering*, 24(11):2065–2079, 2012. doi:10.1109/TKDE.2011.140.

[87] Sarang Joshi, Raj Varma Kommaraji, Jeff M. Phillips, and Suresh Venkatasubramanian. Comparing Distributions and Shapes Using the Kernel Distance. In *Proceedings of the Twenty-Seventh Annual Symposium on Computational Geometry*, SoCG '11, pages 47–56, 2011. doi:10.1145/1998196.1998204.

[88] Maged N. Kamel Boulos and John P. Wilson. Geospatial techniques for monitoring and mitigating climate change and its effects on human health. *International Journal of Health Geographics*, 22(1):2, 2023. doi:10.1186/s12942-023-00324-9.

[89] Marek Karpinski and Angus Macintyre. Polynomial Bounds for VC Dimension of Sigmoidal Neural Networks. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '95, pages 200–208, 1995. doi:10.1145/225058.225118.

[90] Eamonn J. Keogh and Chotirat A. Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and information systems*, 7(3):358–386, 2005. doi:10.1007/S10115-004-0154-9.

[91] János Komlós, János Pach, and Gerhard Woeginger. Almost tight bounds for $\varepsilon$-nets. *Discrete & Computational Geometry*, 7:163–173, 1992. doi:10.1007/BF02187833.

[92] Tobias Kremer, Elmar Schömer, Christian Euler, and Michael Riemer. Cluster Analysis Tailored to Structure Change of Tropical Cyclones Using a Very Large Number of Trajectories. *Monthly Weather Review*, 148(10):4209 – 4229, 2020. doi:10.1175/MWR-D-19-0408.1.

[93] Jae-Gil Lee and Minseo Kang. Geospatial big data: Challenges and opportunities. *Big Data Research*, 2(2):74–81, 2015. doi:10.1016/j.bdr.2015.01.003.

[94] Songnian Li, Suzana Dragicevic, Francesc Antón Castro, Monika Sester, Stephan Winter, Arzu Coltekin, Christopher Pettit, Bin Jiang, James Haworth, Alfred Stein, and Tao Cheng. Geospatial big data handling theory and methods: A review and research challenges. *ISPRS Journal of Photogrammetry and Remote Sensing*, 115:119–133, 2016. doi:10.1016/j.isprsjprs.2015.10.012.

[95] Yi Li, Philip M. Long, and Aravind Srinivasan. Improved Bounds on the Sample Complexity of Learning. *Journal of Computer and System Sciences*, 62(3):516–527, 2001. doi:10.1006/JCSS.2000.1741.

[96] Michael Lindenbaum and Shai Ben-David. Applying VC-dimension analysis to 3D object recognition from perspective projections. In *Proceedings of the 12th National Conference on Artificial Intelligence*, pages 985–990, 1994.

[97] Michael Lindenbaum and Shai Ben-David. Applying VC-dimension analysis to object recognition. In *Computer Vision - ECCV'94, Third European Conference on Computer Vision*, pages 239–250, 1994. doi:10.1007/3-540-57956-7_29.

[98] Stuart P. Lloyd. Least Squares Quantization in PCM. *IEEE Trans. on information theory*, 28(2):129–137, 1982.

[99] Rick Lumpkin and Luca Centurioni. NOAA Global Drifter Program quality-controlled 6-hour interpolated data from ocean surface drifting buoys. NOAA National Centers for Environmental Information. Dataset, 2010. doi:10.25921/7ntx-z961.

[100] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The M5 competition: Background, organization, and implementation. *International Journal of Forecasting*, 38(4):1325–1336, 2022. doi:10.1016/j.ijforecast.2021.07.007.

[101] Bodo Manthey and Heiko Röglin. Improved smoothed analysis of the k-means method. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 461–470, 2009. doi:10.1137/1.9781611973068.51.

[102] John Milnor. On the Betti numbers of real varieties. *Proceedings of the American Mathematical Society*, 15(2):275–280, 1964.

[103] Nabil H. Mustafa and Kasturi R. Varadarajan. Epsilon-approximations and epsilon-nets. *CoRR*, 2017. doi:10.48550/ARXIV.1702.03676.

[104] Abhinandan Nath and Erin Taylor. k-median clustering under discrete Fréchet and Hausdorff distances. In *36th International Symposium on Computational Geometry (SoCG 2020)*, volume 164, page 58, 2020. doi:10.4230/LIPICS.SOCG.2020.58.

[105] Marcus G. Pandy. Computer Modeling and Simulation of Human Movement. *Annual Review of Biomedical Engineering*, 3(Volume 3, 2001):245–273, 2001. doi:10.1146/annurev.bioeng.3.1.245.

[106] François Petitjean. DBA. *GitHub repository*, 2014. https://github.com/fpetitjean/DBA.git.

[107] François Petitjean, Alain Ketterlin, and Pierre Gançarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition*, 44(3):678–693, 2011. doi:10.1016/j.patcog.2010.09.013.

[108] Ivan Georgievich Petrovskii and Olga Arsen'evna Oleinik. On the topology of real algebraic surfaces. *Izvestiya Rossiiskoi Akademii Nauk. Seriya Matematicheskaya*, 13(5):389–402, 1949.

[109] Dimitrie Pompeiu. Sur la continuité des fonctions de variables complexes. *Annales de la Faculté des sciences de l'Université de Toulouse pour les sciences mathématiques et les sciences physiques*, 7(3):265–315, 1905.

[110] Sen Qiao, Yilin Wang, and Jian Li. Real-time human gesture grading based on OpenPose. In Qingli Li, Lipo Wang, Mei Zhou, Li Sun, Song Qiu, and Hongying Liu, editors, *10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics, CISP-BMEI 2017*, pages 1–6. IEEE, 2017. doi:10.1109/CISP-BMEI.2017.8301910.

[111] Sameera V Mohd Sagheer and Sudhish N George. A review on medical image denoising algorithms. *Biomedical signal processing and control*, 61:102036, 2020. doi:10.1016/J.BSPC.2020.102036.

[112] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 26:43–49, 1978. doi:10.1109/TASSP.1978.1163055.

[113] Md. Nazirul Islam Sarker, Bo Yang, Yang Lv, Md. Enamul Huq, and Md. Kamruzzaman. Climate Change Adaptation and Resilience through Big Data. *International Journal of Advanced Computer Science and Applications*, 2020. doi:10.14569/IJACSA.2020.0110368.

[114] Norbert Sauer. On the Density of Families of Sets. *J. Comb. Theory, Ser. A*, 13(1):145–147, 1972. doi:10.1016/0097-3165(72)90019-2.

[115] David Schultz and Brijnesh Jain. Nonsmooth analysis and subgradient methods for averaging in dynamic time warping spaces. *Pattern Recognition*, 74:340–358, 2018. doi:10.1016/J.PATCOG.2017.08.012.

[116] Young-Woo Seo, Chris Urmson, David Wettergreen, and Jin-Woo Lee. Building lane-graphs for autonomous parking. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 6052–6057, 2010. doi:10.1109/IROS.2010.5650331.

[117] Skyler Seto, Wenyu Zhang, and Yichen Zhou. Multivariate time series classification using dynamic time warping template selection for human activity recognition. In *IEEE Symposium Series on Computational Intelligence*, pages 1399–1406, 2015. doi:10.1109/SSCI.2015.199.

[118] Agram Piyush Shanker and A.N. Rajagopalan. Off-line signature verification using DTW. *Pattern Recognition Letters*, 28:1407–1414, 2007. doi:10.1016/j.patrec.2007.02.016.

[119] Roniel S. De Sousa, Azzedine Boukerche, and Antonio A. F. Loureiro. Vehicle Trajectory Similarity: Models, Methods, and Applications. *ACM Comput. Surv.*, 53(5), 2020. doi:10.1145/3406096.

[120] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM (JACM)*, 51(3):385–463, 2004. doi:10.1145/990308.990310.

[121] Rachel St John, Sándor F. Tóth, and Zelda B. Zabinsky. Optimizing the Geometry of Wildlife Corridors in Conservation Reserve Design. *Operations Research*, 66(6):1471–1485, 2018. doi:10.1287/opre.2018.1758.

[122] Han Su, Shuncheng Liu, Bolong Zheng, Xiaofang Zhou, and Kai Zheng. A survey of trajectory distance measures and performance evaluation. *The VLDB Journal*, 29(1):3–32, 2020. doi:10.1007/S00778-019-00574-9.

[123] Thanchanok Teeraratkul, Daniel O'Neill, and Sanjay Lall. Shape-based approach to household electric load curve clustering and prediction. *IEEE Transactions on Smart Grid*, 9(5):5196–5206, 2017. doi:10.1109/TSG.2017.2683461.

[124] Holger Teichgraeber and Adam R Brandt. Clustering methods to find representative periods for the optimization of energy systems: An initial framework and comparison. *Applied energy*, 239:1283–1293, 2019. doi:10.1016/j.apenergy.2019.02.012.

[125] René Thom. Sur l'homologie des variétés algébriques réelles. *Differential and combinatorial topology*, pages 255–265, 1965.

[126] Csaba D Toth, Joseph O'Rourke, and Jacob E Goodman. *Handbook of discrete and computational geometry*. CRC press, 2017.

[127] Marc van Kreveld, Tillmann Miltzow, Tim Ophelders, Willem Sonke, and Jordi L. Vermeulen. Between shapes, using the Hausdorff distance. *Computational Geometry*, 100, 2022. doi:10.1016/j.comgeo.2021.101817.

[128] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer New York, NY, 2000. doi:10.1007/978-1-4757-3264-1.

[129] Vladimir N. Vapnik and Alexey Ya. Chervonenkis. On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, 1971. doi:10.1137/1116025.

[130] Andrea Vattani. k-means requires exponentially many iterations even in the plane. *Discrete & Computational Geometry*, 45(4):596–616, 2011. doi:10.1007/s00454-011-9340-1.

[131] Sheng Wang, Zhifeng Bao, J. Shane Culpepper, and Gao Cong. A Survey on Trajectory Data Management, Analytics, and Learning. *ACM Computing Surveys*, 54(2):39:1–39:36, 2022. doi:10.1145/3440207.

[132] Hugh E Warren. Lower bounds for approximation by nonlinear manifolds. *Transactions of the American Mathematical Society*, 133(1):167–178, 1968. doi:10.2307/1994937.

[133] Shiqing Wei, Shunping Ji, and Meng Lu. Toward Automatic Building Footprint Delineation From Aerial Images Using CNN and Regularization. *IEEE Transactions on Geoscience and Remote Sensing*, 58(3):2178–2189, 2020. doi:10.1109/TGRS.2019.2954461.

[134] Linda Young, Kiyoshi Ueda, Markus Gühr, Philip H Bucksbaum, Marc Simon, Shaul Mukamel, Nina Rohringer, Kevin C Prince, Claudio Masciovecchio, Michael Meyer, et al. Roadmap of ultrafast x-ray atomic and molecular physics. *Journal of Physics B: Atomic, Molecular and Optical Physics*, 51(3):032003, 2018. doi:10.1088/1361-6455/aa9735.

[135] Manzhu Yu, Chaowei Yang, and Yun Li. Big Data in Natural Disaster Management: A Review. *Geosciences*, 8(5), 2018. doi:10.3390/geosciences8050165.

[136] Guan Yuan, Penghui Sun, Jie Zhao, Daxing Li, and Canwei Wang. A review of moving object trajectory clustering algorithms. *Artifcial Intelligence Review*, 47(1):123–144, 2017. doi:10.1007/S10462-016-9477-7.