Subtrajectory Clustering: Finding Set Covers for Set Systems of Subcurves

Hugo A. Akitaya
Department of Computer Science, University of Massachusetts Lowell, USA

Frederik Brüning
Department of Computer Science, University of Bonn, Germany

Erin Chambers
Department of Computer Science, Saint Louis University, USA.

Anne Driemel
Hausdorff Center for Mathematics, University of Bonn, Germany

Abstract

We study subtrajectory clustering under the Fréchet distance. Given one or more trajectories, the task is to split the trajectories into several parts, such that the parts have a good clustering structure. We approach this problem via a new set cover formulation, which we think provides a natural formalization of the problem as it is studied in many applications. Given a 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 ℓ such that every point on P is covered by a subtrajectory that has small Fréchet distance to one of the k center curves ($\leq \Delta$). In many application scenarios, one is interested in finding clusters of small complexity, which is controlled by the parameter ℓ. Our main result is a bicriterial approximation algorithm: if there exists a solution for given parameters k, ℓ, and Δ, then our algorithm finds a set of k' center curves of complexity at most ℓ with covering radius Δ' with $k' \in O(k\ell^2 \log(k\ell))$, and $\Delta' \leq 19\Delta$. Moreover, within these approximation bounds, we can minimize k while keeping the other parameters fixed. If ℓ is a constant independent of n, then, the approximation factor for the number of clusters k is $O(\log k)$ and the approximation factor for the radius Δ is constant. In this case, the algorithm has expected running time in $\tilde{O}\left(km^2 + mn\right)$ and uses space in $O(n+m)$, where $m = \lceil \frac{L}{\Delta} \rceil$ and L is the total arclength of the curve P.

Keywords and phrases Clustering, Set cover, Fréchet distance, Approximation algorithms

Digital Object Identifier 10.57717/cgt.v2i1.7

1 Introduction

Trajectories appear in many different applications in the form of recorded sequences of positions of moving objects. A trajectory is usually modelled as a piecewise linear curve by interpolating between two consecutive location measurements. Standard examples include trajectories of migrating animals, sports players on the field, and vehicles in traffic [37, 38]. Other examples include time series data from sensor measurements tracking the movement of a hand for gesture analysis [36], or the focal point of attention during eye tracking [33, 28]. One particular question in trajectory analysis which has gotten much attention relates to clustering this type of data; typically, one wishes to extract patterns that summarize the data well. This necessitates a notion of similarity (or dissimilarity) to compare and evaluate simplified representations of curves. The Fréchet distance is one such measure, which in addition to geometric closeness also takes the flow of the curve into account; see Section 1.3 for the precise definition.

In this paper, we consider the problem of subtrajectory clustering. The main difference to standard trajectory clustering is that the input curves may be broken into subcurves by the clustering algorithm. Indeed, this approach is well motivated, as trajectory data is often collected over longer periods of time, and the start and ending of the trajectories often do

not carry any particular meaning. In a sense, then, any particular trajectory might naturally break down into subtrajectories, for example when a car's route involves several independent stops as opposed to a single continuous trip. A goal of subtrajectory clustering is to find patterns within the trajectory data and to let an algorithm find the starting and ending points of these patterns by means of solving an optimization problem. There is much work on different variants of this subtrajectory clustering problem and many heuristics have been proposed, see also the surveys [40, 19, 39] and references therein. However, there does not seem to be a rigorous and commonly agreed upon definition of the underlying optimization problem.

The purpose of this paper is to propose a class of problems that capture the nature of the subtrajectory clustering problem and provide algorithmic solutions with provable guarantees. Our work draws from ideas and techniques developed in works on the (k, ℓ)-clustering variant for trajectories [26, 16, 35, 17], where the complexity of centers is restricted by a parameter. We develop algorithmic techniques that build upon fundamental work on computing hitting sets of set systems for low VC-dimension. In particular, we use the set cover framework algorithm by Brönniman and Goodrich [11]. This framework algorithm is related to the multiplicative weights update method [10] and has been used in numerous applications. In computational geometry, it has been used for projective clustering [6] and the art gallery problem [30]. The basic sampling technique underlying these algorithms goes back to Clarkson [22, 23], see also the survey by Agarwal and Sharir [7]. To the best of our knowledge, the framework has not been applied to subtrajectory clustering before. We remark that the algorithm by Brönniman and Goodrich [11] has been revisited and improved several times [5, 20], but our methods do not seem to profit from these improvements.

1.1 Related work

One of the earlier works on clustering subtrajectories is by Lee, Han and Whang [34]. They were interested in computing a small set of line segments that describe the geometry of the input trajectories well. Their algorithm works in two phases: (i) a partition phase where they employ the minimum-description-length (MDL) principle and (ii) a grouping phase where they use a density based clustering algorithm similar to DBSCAN [29].

In general, it is not obvious how to combine the two phases—partitioning and grouping—into one optimization problem. Buchin et al. [15] focus on the problem of finding one single cluster of subtrajectories that are similar to each other. More specifically, they define a subtrajectory cluster with parameters s, Δ, and ℓ, as a set of s pairwise disjoint subtrajectories with pairwise Fréchet distance at most Δ and such that at least one of the subtrajectories has complexity at least ℓ. They define three optimization problems that each optimize one of the three parameters while keeping the other two fixed. The decision problem where all three parameters are fixed is shown NP-complete via reduction from the MaxClique problem. They give 2-approximation algorithms:

- **(i)** for finding the longest subtrajectory cluster (max ℓ) and
- **(ii)** for finding the subtrajectory cluster with the maximum number of subtrajectories (max s).

In subsequent work, these algorithms have been used as building blocks in several heuristic algorithms for map construction [13, 14], where the task is to infer an underlying road map from a set of trajectories.

A natural way to define a global optimization criterion for subtrajectory clustering is by using the set cover problem: given a set of elements X and a set of subsets $\mathcal{R} \subseteq 2^X$, select a minimum number of sets from $\mathcal{R}$, such that their union covers all of X.

Indeed, set cover formulations are used implicitly and explicitly in many algorithms for subtrajectory clustering. Buchin, Kilgus and Kölzsch [18] study migration patterns of animals. They want to derive an augmented geometric graph (a so-called group diagram) that captures the common movement of a group of migrating animals. An input trajectory is represented in the group diagram if there exists a path in the graph that is similar to it under some predefined similarity measure, such as the Fréchet distance. Their algorithm constructs a set cover instance by extracting a linear number of subtrajectory clusters using the algorithm of [15] (see the discussion above). Overall, the algorithm takes time in $O(k^3N^3)$ given k trajectories, each of at most N vertices. However, they use a preprocessing phase that introduces additional vertices which may increase N quadratically in the worst case leading to an overall running time of $O(k^3N^6)$. The approximation factor for the number of clusters selected is $O(\log kN)$.

Agarwal et al. [3] proposed a problem formulation based on facility location for subtrajectory clustering under the discrete Fréchet distance. Their objective function is based on the number of centers, the distances of subtrajectories to their assigned cluster centers and penalties for subtrajectories that are not included in any cluster. They also consider a set cover problem as an intermediate step of their algorithm, but their formulation leads to a set system of exponential size. They present $O(\log n)$-approximation algorithms, where n is the total number of vertices of the input curves. Their algorithm runs in $O(|B|n^3)$ if B is a set of candidate center curves given with the input. They show how to generate a suitable set B of size $O(n^2)$, and how to reduce the size to $O(n)$ at the expense of an additional $O(\log n)$-factor in the approximation quality.

1.2 Organization

In the remainder of this section, we give some preliminary definitions in Section 1.3, we define the problem statement in Section 1.4 and a modified problem statement in Section 1.5. We give an overview of our main results in Section 1.6 and discuss other problem variants in Section 8. We then discuss our main techniques in Section 2. In Sections 3 and 4 we discuss solutions to the modified problem. In Sections 5 we discuss our solution to the main problem stated in Section 1.4. In Sections 6 and 7 we discuss additional results.

1.3 Preliminaries

A sequence of n points $p_1, \ldots, p_n \in \mathbb{R}^d$ defines a ***polygonal curve*** P by linearly interpolating consecutive points, that is, for each i, we obtain the ***edge*** $\{tp_i + (1-t)p_{i+1} | t \in [0,1]\}$. We may think of P as resulting from the concatenation of the edges in the given order as a parametrized curve, that is, a function $P : [0,1] \mapsto \mathbb{R}^d$. Note that for any such parametrized curve there exist real values $s_1 \leq \cdots \leq s_n$, such that $P(s_i) = p_i$. We call the ordered set of the p_i the ***vertices*** of P and we denote it with $V(P)$. We call the number of vertices n the ***complexity*** of the curve. For any two $[a,b] \subseteq [0,1]$ we denote with $P[a,b]$ the ***subcurve*** of P that starts at $P(a)$ and ends at $P(b)$. Let $\mathbb{X}_\ell^d = (\mathbb{R}^d)^\ell$, and think of the elements of this set as the set of all polygonal curves of ℓ vertices in $\mathbb{R}^d$. For two parametrized curves P and Q, we define their ***Fréchet distance*** as

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

where γ ranges over all strictly monotone increasing functions. A curve $Q \in \mathbb{X}_\ell^d$ is called an ***ℓ-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 ℓ-simplification for a polygonal curve of n vertices.

Let X be a set. We call a set $\mathcal{R}$ where any $r \in \mathcal{R}$ is of the form $r \subseteq X$ and $X = \bigcup_{r\in\mathcal{R}} r$ a ***set system*** with ***ground set*** X. Let $\mathcal{R}$ be a set system 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. In addition to the set cover problem, we define the ***hitting set problem***. Let $\mathcal{R}$ be a set system with ground set X. 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 set system ***dual*** to $\mathcal{R}$. The set system $\mathcal{R}^*$ 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 set system of $\mathcal{R}^*$ is again $\mathcal{R}$. The hitting set problem for $\mathcal{R}$ is equivalent to the set cover problem for the dual set system $\mathcal{R}^*$. 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 $\mathcal{R}$ is the maximal size of a set A that is shattered by $\mathcal{R}$. When stating asymptotic bounds, we may use the $\widetilde{O}(\cdot)$ notation hiding polylogarithmic factors to simplify the exposition.

1.4 Problem definition

Let $P : [0,1] \to \mathbb{R}^d$ be a parametrized curve[1] and let $\ell \in \mathbb{N}$ and $\Delta \in \mathbb{R}$ be fixed parameters. Define the Δ-***coverage*** of a set of center curves $C \subset \mathbb{X}_\ell^d$ as follows:

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

Note that this corresponds to the part of the curve P that is covered by the set of all subtrajectories that are within Fréchet distance Δ to some curve in C. The problem we study in this paper is to find a set $C \subset \mathbb{X}_\ell^d$ of minimum size such that the Δ-coverage of C covers all of P. 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 ℓ, which is given as a parameter.[2]

1.5 Set system formulation

Our approach to the subtrajectory clustering problem (see Section 1.4) works via set covers of suitable set systems. To this end, we will first define a ***discrete variant*** of the problem. 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 $1 \le i \le m$ 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

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

[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.

[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.

Analogous to the problem definition in Section 1.4 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 set system $\mathcal{R}$ 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

$$r_Q = \{z \in X \mid \exists i \leq z < j \text{ with } d_F(Q, P[t_i, t_j]) \leq \Delta\} \tag{1}$$

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 set system $\mathcal{R}$. Stating the problem in terms of set systems allows us to draw from a rich background of algorithmic techniques for computing set covers (see Section 2).

We first discuss how solutions to the discrete problem help solving the initial problem defined in Section 1.4. We can choose breakpoints for the input curve P, such that the distance between two consecutive breakpoints is at most $\epsilon\Delta$ for any fixed $\epsilon > 0$. This is always possible with $m = \lceil \frac{L}{\epsilon\Delta} \rceil$ breakpoints, where L 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 1.** *Assume there exists a set $C^* \subset \mathbb{X}_\ell^d$ of size k, such that $\psi(P, C^*) \leq \Delta$. Then we have $\phi(P, C^*) \leq (1+\epsilon)\Delta$. Additionally for each $C \subset \mathbb{X}_\ell^d$ with $\phi(P, C) \leq (1+\epsilon)\Delta$ we have $\psi(P, C^*) \leq (1+\epsilon)\Delta$.*

Proof. We show that for any set of center curves $C \subset \mathbb{X}_\ell^d$ we have $\psi(P, C) \leq \phi(P, C) \leq (1+\epsilon)\psi(P, C)$. 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 $\epsilon\Delta$. 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. ◀

1.6 Main results

We study the problem of subtrajectory clustering in the concrete form as defined in Section 1.4. 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 8). 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 Δ.

In Sections 3 and 4 we describe our approach for the discrete variant of the subtrajectory clustering problem defined in Section 1.5, before we turn to the main problem in Section 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 2.** *Let $P : [0, 1] \to \mathbb{R}^d$ be a polygonal curve of complexity n with breakpoints $0 \leq t_1, \ldots, t_m \leq 1$ 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 set system that preserves optimal solutions up to approximation and at the same time allows for efficient set system oracles. A set system

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 set system 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 et al. [4] to define suitable proxy curves. This is described in Section 4 and leads to the following result.

▶ **Theorem 3.** *Let $P : [0,1] \to \mathbb{R}^d$ be a polygonal curve of complexity n with breakpoints $0 \le t_1, \dots, t_m \le 1$. Assume there exists a set $C^* \subset \mathbb{X}_\ell^d$ of size $k \le m$, such that $\phi(P, C^*) \le \Delta$. Then there exists an algorithm that computes a set $C \subset \mathbb{X}_\ell^d$ of size $O(k \log(m) \log^2(m))$ such that $\phi(P, C) \le 50\Delta$. 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 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 1.4). We use the techniques developed in Section 4, but we obtain better approximation factors and running times, compared to a naive application of Lemma 1. The improved running time results from the fact that we do not need to keep track of breakpoints explicitly in the set system oracle. Crucial to obtaining better approximation factors is the analysis of the VC-dimension of the dual set system. We obtain the following theorem.

▶ **Theorem 4** (Main Theorem)**.** *Let $P : [0,1] \to \mathbb{R}^d$ be a polygonal curve of complexity n, let $\ell \in \mathbb{N}$ and $\Delta, \varepsilon > 0$ be parameters. Assume there exists a set $C^* \subset \mathbb{X}_\ell^d$ of size k, such that $\psi(P, C^*) \le \Delta$. Let $m = \lceil \frac{L}{\epsilon\Delta} \rceil$ and $\delta = O(d^2\ell^2 \log(d\ell))$, there exists an algorithm that computes a set $C \subset \mathbb{X}_\ell^d$ of size $O(k\delta \log(\delta k))$, such that $\psi(P, C) \le (18 + \epsilon)\Delta$. The algorithm has expected running time in $\widetilde{O}\left(km^2 + mn\right)$ and uses space in $O(n + m)$, where we assume that ℓ and d are constants independent of n.*

In particular, in the above theorem, when the complexity of center curves ℓ and the ambient dimension d are constants, the VC-dimension δ is constant, and the approximation factor for the size of the set cover is $O(\log k)$. For a comparison, using Theorem 3 and Lemma 1 directly would result in an approximation factor of $O(\log \frac{L}{\Delta} \log^2 \frac{L}{\Delta})$ which could be large even if ℓ and d are small. We summarize our results in Table 1.

Size k'	Δ'	Running time	Space	Setting	Reference
$O(k \log^2(m))$	6Δ	$\widetilde{O}\left(km^2 + mn\right)$	$O(n + m^2)$	$\ell = 2$, discrete	Thm. 2
$O(k \log(m) \log^2(m))$	50Δ	$\tilde{O}\left(km^2 + mn\right)$	$O(n + m^2)$	$\ell \ge 1$, discrete	Thm. 3
$O(k \log(k))$	19Δ	$\tilde{O}(k\lceil \frac{\lambda}{\Delta} \rceil^2 + \lceil \frac{\lambda}{\Delta} \rceil)n)$	$O(n + \lceil \frac{\lambda}{\Delta} \rceil)$	$\ell \ge 1$, contin.	Thm. 4

Table 1 For optimal $C \subset (\mathbb{R}^d)^\ell$ of size k, covering $P \in (\mathbb{R}^d)^n$ under distance Δ, we design bicriteria-approximation algorithms that compute $C' \subset \mathbb{X}_{\ell'}^d$ of size k', covering P under distance Δ'. Here, we assume that ℓ and d are constant, n is the complexity of P and λ is the arclength of P.

The improved approximation factors that are obtained in the continuous case in Theorem 4 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 6 we study lower bounds to the VC-dimension for two natural problem variants. We study the dual set system (i) in the discrete case and (ii) the set system 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 47. 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 46 for the exact statement. Thus, ultimately, our modified set system with proxy curves not only makes the algorithm faster, but also has the benefit of a significantly lower VC-dimension, compared to the exact set system inherent to the problem.

Finally, we also investigate the question of hardness for the discrete problem defined in Section 1.5. If the complexity of center curves ℓ can be large, then NP-hardness follows from the hardness of the shortest common superstring problem, see also the result by Buchin et al. [16] on (k, ℓ)-center clustering under the Fréchet distance. In particular, in this case the problem is also hard to approximate. In Section 7 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.

1.7 Subsequent work

Subsequent to our work, Driemel, Brüning and Conradi [12] showed several improvements to Theorem 4. Using the same general approach combined with a new auxiliary set system, which is based on a different simplification algorithm, they show that one can generate a set of candidates of size independent of the relative arclength. This approach circumvents the use of breakpoints altogether. As a result, they show that there exists an algorithm with expected running time in $\widetilde{O}(k^2n + kn^3)$ using space in $\widetilde{O}(kn + n^3)$ for the continuous case. Secondly, they show that, when using breakpoints to approximate the curve and using implicit weight updates, one can improve the running time with respect to the relative arclength resulting in an algorithm of expected running time in $\widetilde{O}(nk^3 \log^4 \lceil \frac{\lambda}{\Delta} \rceil)$ and using space in $\widetilde{O}(nk \log \lceil \frac{\lambda}{\Delta} \rceil)$. Furthermore, in both algorithms, they improve the dependency on ℓ in the term that bounds the approximation factor (with respect to the number of clusters k) from quadratic to linear. Refer to [12] for the exact theorem statements.

Another paper that appeared subsequently is the paper by Gudmundsson and Wang [31]. This is a follow-up to the line of work initiated by Buchin et al. [15] where the focus is on detecting a single subtrajectory cluster that satisfies certain criteria: a minimum number of disjoint subtrajectories, minimum length, and maximum Fréchet distance between any pair of subtrajectories within the cluster. They consider the discrete and the continuous Fréchet distance and show tight bounds on the fine-grained complexity of this problem which exhibit a separation between the discrete and the continuous case.

2 Setup of techniques

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

We start in Section 2.1 with a simple algorithm that illustrates our general approach in a nutshell: we derive an auxiliary set system that has a simpler structure and smaller size compared to the set system of Section 1.5, while preserving optimal solutions up to approximation. A preliminary result that follows by applying the greedy set cover algorithm is stated in Theorem 6. Then, in Section 2.2 we recapitulate the algorithmic framework by Brönniman and Goodrich [11] which we use in our main algorithm. In order to obtain efficient algorithms from this framework, we need to adapt the framework to our specific needs. The approximation quality of the resulting algorithm strongly depends on the VC-dimension of the dual set system. Therefore, we aim for auxiliary set systems with constant VC-dimension. Alas, this is not always possible when breakpoints are given with the input. We discuss this in Section 2.3.

2.1 A set system 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 Δ 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 set system in a way that preserves the initial structure up to approximation but allows for more efficient algorithms for the clustering problem.

Let $\mathcal{S} = \{(i,j) \in \mathbb{N}^2 \mid 1 \leq i < j \leq m\}$. For any $(i,j) \in \mathcal{S}$ let $\mu_\ell(P[t_i, t_j])$ denote the ℓ-simplification of the corresponding subcurve of P. Consider a set system $\widetilde{\mathcal{R}}_0$ defined on the ground set $X = \{1, \ldots, m-1\}$, 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

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

We will see (Lemma 5, below), that $\widetilde{\mathcal{R}}_0$ approximates the structure of $\mathcal{R}$ as defined in (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 [21]. Applying this algorithm to the set system $\widetilde{\mathcal{R}}_0$, we obtain a set C consisting of ℓ-simplifications $\mu_\ell(P[t_i, t_j]))$ for each $r_{i,j}$, such that $\phi(P, C) \leq 3\Delta$.

2.1.0.1 Building the incidence matrix

To this end, we compute the binary incidence matrix M of the set system $\widetilde{\mathcal{R}}_0$ 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 μ, we compute the Δ-free space with the curve P, which is defined as the level set

$$FD_\delta(P, \mu) = \left\{(x,y) \in [0,1]^2 \mid \|P(x) - \mu(y)\| \leq \delta\right\}.$$

Computing the associated diagram can be done in $O(n\ell)$ time and space [8]. Note that the simplification μ corresponds to the vertical axis of the Δ-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 [8]. For all $i' \leq q < j'$, we set the entry corresponding to q and μ 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.

2.1.0.2 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 5.** *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 6.** *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 ℓ-simplification of a polygonal curve of n vertices.*

Proof. The algorithm builds the incidence matrix of the set system 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 5 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 $\phi(P, C) \leq 3\Delta$. ◀

2.2 The framework for the set cover algorithm

For obtaining our main results we use an idea that goes back to Clarkson [22, 23] and was later also applied and extended by Brönniman and Goodrich [11] for set systems of low VC-dimension. We first describe the framework algorithm and then show how to adapt it to our needs. The idea of the framework algorithm is best explained by taking the point of view that it computes a hitting set of the dual set system. We need the following definition of an ε-net.

▶ **Definition 7.** *Let $\mathcal{R}$ be a set system with finite ground set X and with an additive weight function w on X. An **ε-net** is a subset $S \subset X$, such that every set of $\mathcal{R}$ of weight at least $\varepsilon \cdot w(X)$ contains at least one element of S.*

Note that, if $w(x) = 1$ for each $x \in X$, then an ε-net is a hitting set for the "heavy" sets of $\mathcal{R}$ that contain at least an ε-fraction of the ground set. The framework algorithm needs the following subroutines.

▶ **Definition 8** ([11])**.** *A **net finder** of size s for a set system $(X, \mathcal{R})$ is an algorithm A that, given $\rho \in \mathbb{R}$ and a weight function w on X, returns an $(1/\rho)$-net of size $s(\rho)$ for $(X, \mathcal{R})$ with weight w. Also, a **verifier** is an algorithm B 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 $r \cap H = \emptyset$.*

2.2.0.1 Framework

Given these two subroutines and a finite set system $(X, \mathcal{R})$, the algorithm proceeds as follows. Assume we know there exists a hitting set of size k. The algorithm calls the net finder to compute an ε-net S of $\mathcal{R}$ (for a specific value of ε). Then, the algorithm calls the verifier to test if S is also a hitting set for $\mathcal{R}$. If yes, we return S. If no, then the verifier returns a witness set r that does not contain any element of S. We double the weight of each element of r. Then, we repeat until we find 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 after $4k \log(\frac{|X|}{k})$ weight-updates if no hitting set has been found. If this happens, we double the value of k, reset the weights, and start over.

In the following, we describe an adaptation of this framework algorithm that suits our needs. Our adaptation uses the following definition of a set system oracle. The resulting theorem is stated below.

▶ **Definition 9** (Set system oracle). *For a given set system $\mathcal{R}$ with ground set X a **set system oracle** is a data structure $\mathcal{D}$ that can be queried with any $r \in \mathcal{R}$ and $z \in X$ and answers whether $z \in r$. We denote with $T_P(\mathcal{D})$ the preprocessing time to build the data structure $\mathcal{D}$ for the oracle and with $T_Q(\mathcal{D})$ the time needed to answer the query. We denote with $S_O(\mathcal{D})$ the space required by the data structure.*

▶ **Theorem 10.** *For a given finite set system $(X, \mathcal{R})$ with finite VC-dimension δ, assume there exists a hitting set of size k. Then, there exists an algorithm that computes a hitting set of size $k' \in O(\delta k \log \delta k)$ with expected running time in $O\big((k'|\mathcal{R}| + |X|)\, k \log(|X|)\, T_Q(\mathcal{D}) + T_P(\mathcal{D})\big)$ and using space in $O\big(|X| + S_O(\mathcal{D})\big)$.*

In the remainder of this section we show how to prove Theorem 10 following Brönniman and Goodrich [11]. Theorem 10 and the proof of it differ slightly from their results because we do not use a deterministic net finder. Let $\mathcal{R}$ be a set system with finite ground set X and finite VC-dimension δ. An effective way to implement the net-finder is via a random sample from the ground set, as guaranteed by the ε-net theorem [32] by Haussler and Welzl.

▶ **Theorem 11** ([32]). *For any $(X, \mathcal{R})$ of finite VC-dimension δ, finite $A \subseteq X$ and $0 < \epsilon$, $\alpha < 1$, if N is a subset of A obtained by at least*

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

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

Thus, the net-finder can be implemented to run in $O(|X|)$ time and $O(|X|)$ space, by taking a sample from X where the weights correspond to probabilities. We call this the ***probabilistic net-finder***. While verifying that a set is an ε-net could be costly in our setting, we can observe that this is actually not necessary. Indeed, we can modify the behaviour of the verifier as follows.

▶ **Definition 12** (Extended verifier). *Given a set $S \subseteq X$, the extended verifier returns one of the following:*

(i) *S is a hitting set.*

(ii) *A witness set r with $r \cap S = \emptyset$, and $w(r) \leq \varepsilon \cdot w(X)$.*

(iii) *A witness set r with $r \cap S = \emptyset$, and $w(r) > \varepsilon \cdot w(X)$.*

To implement the extended verifier we assume that we have a set system oracle $\mathcal{D}$ for $(X, \mathcal{R})$. After reprocessing the oracle, the extended verifier can be implemented to run in

$$O(|S| \cdot |\mathcal{R}| \cdot T_Q(\mathcal{D}) + |X| \cdot T_Q(\mathcal{D}))$$

time by using $|\mathcal{R}|$ linear scans over S, one for each set in $\mathcal{R}$. We determine for every set $r \in \mathcal{R}$ whether it is hit by an element of S, by calling the set system oracle on r and the corresponding set and elements in S. If we find a set that is not hit by any of the elements in S, we compute its weight explicitly by using $|X|$ calls to $T_Q(\mathcal{D})$ and return the appropriate answer (ii) or (iii). In case (ii), we return the witness set that we have just computed explicitly, that is, we return all elements of this set, in order for the reweighting to be applied. If we do not find such a set, then S is a hitting set and we return (i).

2.2.0.2 Algorithm.

Using the above implementation of the probabilistic net-finder and the extended verifier, the algorithm for computing a hitting set for a given parameter k now proceeds as follows. In each iteration we use the probabilistic net-finder to sample a candidate set $S \subseteq X$. The sample size is chosen large enough that S is an ε-net with probability greater $\frac{1}{2}$ for $\epsilon = \frac{1}{2k}$. Given S, we apply the extended verifier. If the verifier returns that S is a hitting set (case (i)) then the algorithm terminates with S as a result. If the verifier returns a witness set r with $r \cap S = \emptyset$, and $w(r) \leq \varepsilon \cdot w(X)$ (case (ii)) then we double the weight of each element of r. The algorithm keeps track of the weight of each element and the total weight of the ground set. If the verifier returns a witness set r with $r \cap S = \emptyset$, and $w(r) > \varepsilon \cdot w(X)$ (case (iii)) then S is not an ε-net and we repeat the sample without performing a weight-update. If after $4k \log(\frac{|X|}{k})$ steps that performed a weight-update the algorithm did not find a hitting set, we return that there is no hitting set of size k.

Proof of Theorem 10. We first build a data structure $\mathcal{D}$ for the oracle in $T_P(\mathcal{D})$ time. Then, we use the above algorithm in a doubling search on k, starting with $k = 2$.

In the remaining proof, we analyse the running time of the algorithm described above for a fixed k in detail. In each iteration of the algorithm, the computed random sample of size $O(k\delta \log(\delta k))$ is an ϵ-net with probability greater $\frac{1}{2}$. Therefore, the expected number of iterations until we find an ϵ-net between any two weight-update steps is at most 2. Once we find an ϵ-net, we are either in case (i) or (ii). As soon as we are in case (i) the algorithm terminates and outputs a hitting set of size $O(\delta k \log(\delta k))$. By construction, the number of times we can be in case (ii) is bounded by $4k \log(\frac{|X|}{k})$, as this is the maximum number of weight-update steps before the algorithm terminates. By the analysis in [11], 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 of being in case (ii). 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 iterations in case (ii). Then we have after i iterations in case (ii) 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 $\epsilon = \frac{1}{2k}$, 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 $i \leq 4k \log(\frac{|X|}{k})$. Combining this result with the expected number of iterations until we find an ϵ-net, we conclude that the expected number of iterations before the algorithm terminates is smaller than $8k \log(\frac{|X|}{k})$.

In each iteration, the algorithm computes a random sample in $O(|X|)$ time and applies the extended verifier in $O(|\mathcal{R}|k\delta \log(k) T_Q(\mathcal{D}) + |X| T_Q(\mathcal{D}))$ time. If a reweighting needs to be applied (case (ii)) this can be done in $O(|X|)$ time. So each iteration of the algorithm has a running time of $O(|\mathcal{R}|\delta k \log(\delta k) T_Q(\mathcal{D}) + |X| T_Q(\mathcal{D}))$. In total we get an expected running time of

$$O(k \log(\frac{|X|}{k}) T_Q(\mathcal{D})(\delta k \log(\delta k)|\mathcal{R}| + |X|) + T_P(\mathcal{D})).$$

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 theorem follows by the observation that both the net-finder and the verifier need $O(|X|)$ space. ◀

2.3 Bounding the VC-dimension

In order to use Theorem 10 of Section 2.2, we need to bound the VC-dimension of the dual set system. In our case, this will be a set system that has similar structure as a set system of metric balls under the Fréchet distance studied by Driemel et al. [27]. In a nutshell, they showed a bound of $O(d^2 s^2 \log(ds))$ for polygonal curves in $\mathbb{R}^d$ of complexity at most s. Using this result directly would not gain us any useful bounds, as the subcurves $P[t_i, t_j]$ in the definition of the set system may have linear complexity in n—even for the simpler variant of Section 2.1. In fact, it turns out that the VC-dimension of the dual set system for the main problem defined in Section 1.4 does indeed inherently depend on n, as we show in Theorem 46 in Section 6.1.

In Section 5 we instead define an auxiliary set system that preserves solutions up to approximation and—more importantly—which has low VC-dimension in the dual. We show this by using the approach of Driemel et al. [27]. They derive a set of geometric predicates which specify sufficient information for evaluating whether the Fréchet distance is below a certain threshold. Based on this, they define a composite set system that uses the geometric predicates as building blocks. The VC-dimension can then be bounded using standard composition arguments in combination with a theorem by Anthony and Bartlett [9]. Our analysis of the VC-dimension is given in Section 5.3 and relies on the same set of geometric predicates. We relate these predicates to the distance evaluation of a certain type of partial Fréchet distance with specific conditions that occur in our set system with proxy curves. The result is stated in Theorem 40 and implies that the VC-dimension is constant, if the complexity of the center curves ℓ 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 set system already gives a bound of $O(\log m)$,

however this depends on the number of breakpoints m and can be large even if ℓ is small. We study this problem in Section 6.2. For the set system defined in Section 1.5 we show a lower bound of $\Omega(\log m)$ even in the case that $d = 1$ and $\ell = 2$ (see Theorem 47). Technically, this does not rule out the existence of an auxiliary set system with low VC-dimension in the dual. However, it is not clear what such a set system would look like as Theorem 47 makes only few assumptions on the set system. 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 to what we can currently achieve in the discrete case, when breakpoints are given with the input.

3 Warm-up — Clustering with line segments

In this section, we show how to apply Theorem 10 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. In contrast to the solution described in Section 2.1, our algorithm finds an approximate set cover without computing the set system explicitly leading to better running times.

3.1 The set system

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

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

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

$$d_F(\overline{P(t_x)P(t_z)}, P[t_x, t_z]) \leq 4\Delta. \tag{2}$$

If x satisfies (2), then we decrement x and continue the scan. If $x = 0$ or if x does not satisfy (2), 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 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 1 shows an example of z, x_z and y_z.

3.2 Analysis of the approximation error

In this section we show how we use a set cover of the set system $\widetilde{\mathcal{R}}_2$ to construct an approximate solution for our clustering problem and analyse the resulting approximation error. In particular, we prove Lemma 14 and Lemma 15. To prove them, we first prove the following simple lemma.

▶ **Lemma 13.** *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 $d_F(\pi(I), P[t_{i_1}, t_{i_{m'}}]) \leq 2\alpha$.*

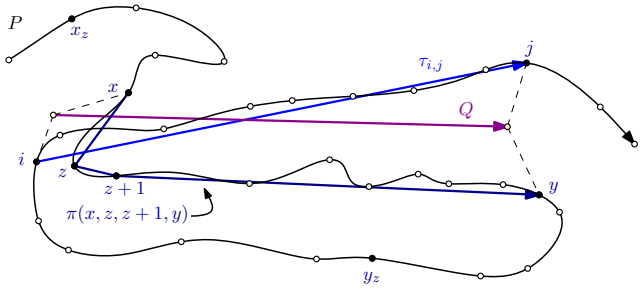


Figure 1 Example of a curve P and index z, such that $z \in r_{i,j}$ for some $r_{i,j} \in \widetilde{\mathcal{R}}_2$. Also shown is a line segment Q, such that $z \in r_Q$ of the initial set system $\mathcal{R}$. After preprocessing, we can test $z \in r_{i,j}$ in constant time.

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

$$d_F(\overline{P(t_{i'})P(t_{j'})}, P[t_{i'}, t_{j'}]) \leq d_F(\overline{P(t_{i'})P(t_{j'})}, Q[a_{i'}, b_{j'}]) + d_F(Q[a_{i'}, b_{j'}], P[t_{i'}, t_{j'}]) \leq 2\alpha.$$

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 14.** *Assume there exists a set cover for $\mathcal{R}$ with parameter Δ. 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

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

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 Δ, 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 13, 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 15.** *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 13, 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 13 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 $r_{i,j} \in \widetilde{\mathcal{R}}_2$. ◀

3.3 The algorithm

We intend to use the algorithm of Theorem 10 to find a set cover of the set system $\widetilde{\mathcal{R}}_2$, since such a set cover gives a 6-approximation for our clustering problem; see Section 2.2.0.1 for details on the algorithm. The algorithm requires a set system oracle for $\widetilde{\mathcal{R}}_2$. In this section, we describe such a set system 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.

3.3.0.1 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 set system in Section 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

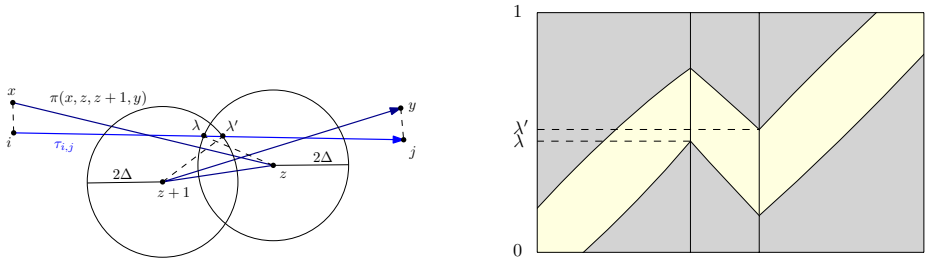


Figure 2 Illustration of Observation 16. The figure on the right shows the 2Δ-free-space diagram of the two curves on the left. A monotone path from the bottom left to the upper right corner of the diagram is feasible iff the three conditions stated in the observation are satisfied. We slightly abuse notation by referring to the vertex $P(t_z)$ with z in all figures, when context is clear.

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.

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

Otherwise, the algorithm returns "no".

3.3.0.3 Correctness.

The above described set system oracle returns the correct answer. Correctness is implied by the following observation, which follows from the analysis of Alt and Godau [8]. See also Figure 2.

▶ **Observation 16.** *$d_F(\pi(x, z, z + 1, y), \overline{P(t_i)P(t_j)}) \leq 2\Delta$ 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 $v = P(t_j)$.

3.3.0.4 Running time.

Next, we analyse the running time of constructing an oracle for the case $\ell = 2$ and query time $O(1)$. In particular we analyse 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 [8]. 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 2Δ 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 17.** *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.*

3.4 The result

For the set system $(Z, \widetilde{\mathcal{R}}_2)$, we have $|Z| = m$ and $|\widetilde{\mathcal{R}}_2| = O(m^2)$. Thus, the VC-dimension δ of the dual set system is trivially bounded by $O(\log m)$. We combine this with the result for constructing the oracle in Theorem 17 and apply Theorem 10 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 18.** *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 14 and Lemma 15.

▶ **Theorem 2.** *Let $P : [0,1] \to \mathbb{R}^d$ be a polygonal curve of complexity n with breakpoints $0 \leq t_1, \ldots, t_m \leq 1$ 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 The main algorithm

In this section we extend the scheme described in Section 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 set system for approximation which allows for an efficient implementation of a set system oracle. The main idea is to replace the edges of the proxy curve π from Section 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 set system. We will later show how to use the main elements of this algorithm for the continuous case in Section 5.

4.1 Simplifications

We begin by introducing the following slightly different notion of simplification. A curve $Q \in \mathbb{X}_\ell^d$ is an (ϵ, ℓ)-***simplification*** of a curve P if Q has at most ℓ vertices and its Fréchet distance to P is at most ϵ. We call the simplification ***vertex-restricted*** if $V(Q) \subseteq V(P)$

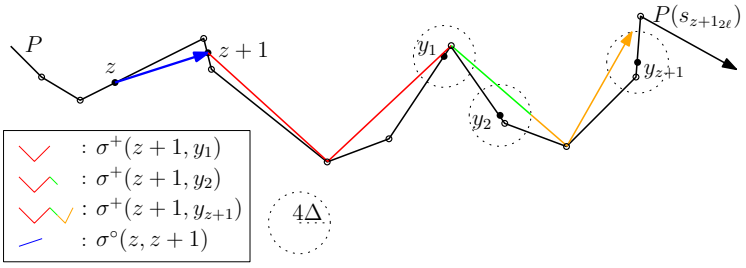


Figure 3 Example of the generated $(4\Delta, 2\ell)$-simplifications for a curve P with breakpoints $z, z+1, y_1, y_2$ and y_{z+1} in the case $\ell = 2$.

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 $i,j \in \{1,\dots,m\}$ that we will use in the definition of the set system in the next section. Concretely, the simplifications will be defined as the output of the algorithm by Agarwal et. al. [4]. In a nutshell, their algorithm works the following way: Let P be a curve with vertices $P(s_1),\dots,P(s_n)$. Let $f(\frac{\epsilon}{2})$ denote the minimum number of vertices in a vertex-restricted $(\frac{\epsilon}{2}, n)$-simplification of P. To compute a vertex-restricted $(\epsilon, f(\frac{\epsilon}{2}))$-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 $d_F(\overline{P(s_i)P(s_{i+2^j})}, P[s_i, s_{i+2^j}]) \leq \epsilon$. After determining j it finds with a binary search the last integer $r \in [2^j, 2^{j+1}]$ such that $d_F(\overline{P(s_i)P(s_{i+r})}, P[s_i, s_{i+r}]) \leq \epsilon$. The algorithm terminates when it reaches $P(s_n)$.

4.1.0.1 Generating simplifications.

We now describe how to generate a set of simplifications that will be used in the definition of our set system in Section 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 $z \in \{1,\dots,m\}$ we apply the algorithm with $\epsilon = 4\Delta$ on $P[t_z, 1]$ to get a simplification P_z^+. We stop the algorithm early if the complexity of the simplification reaches 2ℓ. If $|P_z^+| = 2\ell$ let $P(s_{z_{2\ell}})$ be the 2ℓ-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 $P(s_{z_{2\ell}})$. 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 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 set system later.

▶ **Lemma 19.** *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 20.** *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 21.** *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 19 follows directly from the following lemma. Lemma 20 and Lemma 21 follow by using symmetric arguments.

▶ **Lemma 22.** *Consider the generating process described in Section 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 Agarwal et. al. 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 (Δ, ℓ)-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 γ 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. ◀

4.2 The set system

We are now ready to define the new set system $\widetilde{\mathcal{R}}_3$ with ground set $Z = \{1, \ldots, m-1\}$. The set system 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. Note that, by Lemma 21 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 set system is well-defined as implied by the Lemmas 19, 20 and 21.

4.3 Analysis of the approximation error

We show correctness in the same schema as in Section 3.2. In particular, we prove Lemma 23 and Lemma 24.

▶ **Lemma 23.** *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 $\phi(P, C) \leq 14\Delta$.*

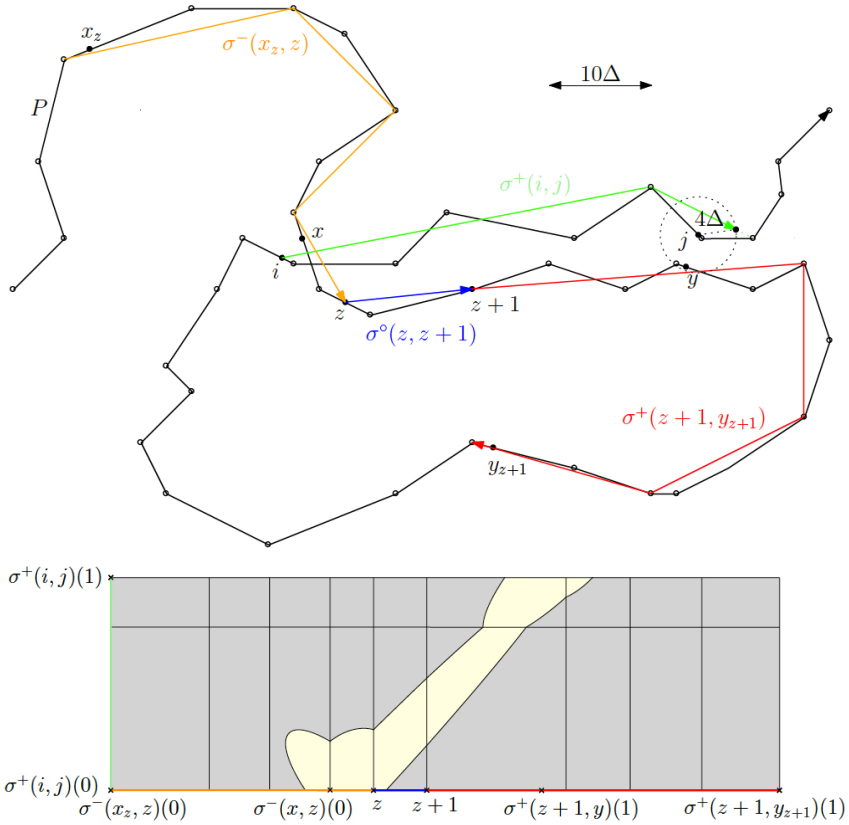


Figure 4 Example of a curve P such that $z \in r_{i,j}$ for some $r_{i,j} \in \widetilde{\mathcal{R}}_3$. Also shown is the 10Δ-free space diagram of $\kappa_z(x, y)$ and $\sigma^+(i, j)$. Simplification $\sigma^+(i, j)$ demonstrates that the simplifications do not have to be vertex-restricted.

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 $d_F(\sigma^-(x,z), P[t_x,t_z])$, $d_F(\sigma^\circ(z,z+1), P[t_z,t_{z+1}])$ 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 $C' \subseteq \mathbb{X}^d_{2\ell}$. 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 ℓ. 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 14Δ. So, if we split each curve $c \in C'$ as described above, we obtain a set $C \subseteq \mathbb{X}^d_\ell$ with $|C| = 3|C'|$ and $\phi(P, C) \leq 14\Delta$. ◀

▶ **Lemma 24.** *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 ℓ since Q has complexity at most ℓ. By the Lemmas 20 and 19, 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 The approximation oracle

To find a set cover of the set system $\widetilde{\mathcal{R}}_3$ we want to use the framework described in Section 2.2.0.1. But to apply Theorem 10 directly we would need to implement an oracle that answers for an element of the ground set $Z = \{1,\dots,m-1\}$ 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 4.5) we then show how to apply Theorem 10.

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 $\sigma^+(i,j)$ 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 25.** *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 start on an active edge of $\sigma^-(x_z, z)$ and end 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 5 for an example.

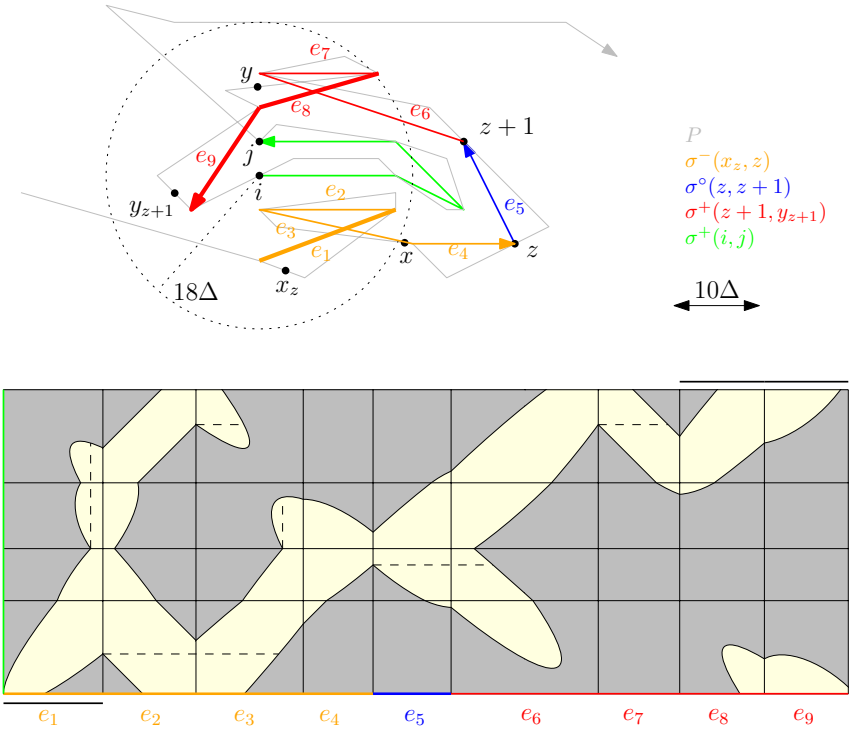


Figure 5 Example of a curve P and breakpoints x_z, x, z, $z+1$, y, y_{z+1}, i and j. The active edges are e_1, e_8 and e_9 since there are breakpoints corresponding to these edges within distance 18Δ to i or j respectively. There is, however, no strictly monotone path from e_1 on the bottom to e_8 or e_9 on the top in the 10Δ-free space of $\sigma^+(i,j)$ and $\kappa_z(x_z, y_{z+1})$. So we have $z \notin r_{i,j}$.

4.4.0.1 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 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 analyse its running time.

4.4.0.2 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 Agarwal et. al. [4] up to complexity 2ℓ. 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 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 [8] in the following way.

The free space diagram of the 10Δ-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 $B^R_{s^*+1,t} \neq \emptyset$.

4.4.0.3 Correctness.

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

▶ **Lemma 26.** *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 $d_F(\kappa_z(x, y), \sigma^+(i, j)) \leq 46\Delta$*
- **(ii)** *"No", then we have $z \notin r_{i,j}$.*

Proof. i) Consider the 10Δ-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 10Δ-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 46Δ-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 choice 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 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
  $$\max(d(\sigma^+(i, j)(0), \sigma^-(x_e, z)(0)), d(\sigma^+(i, j)(0), p_e)) \leq 26\Delta$$

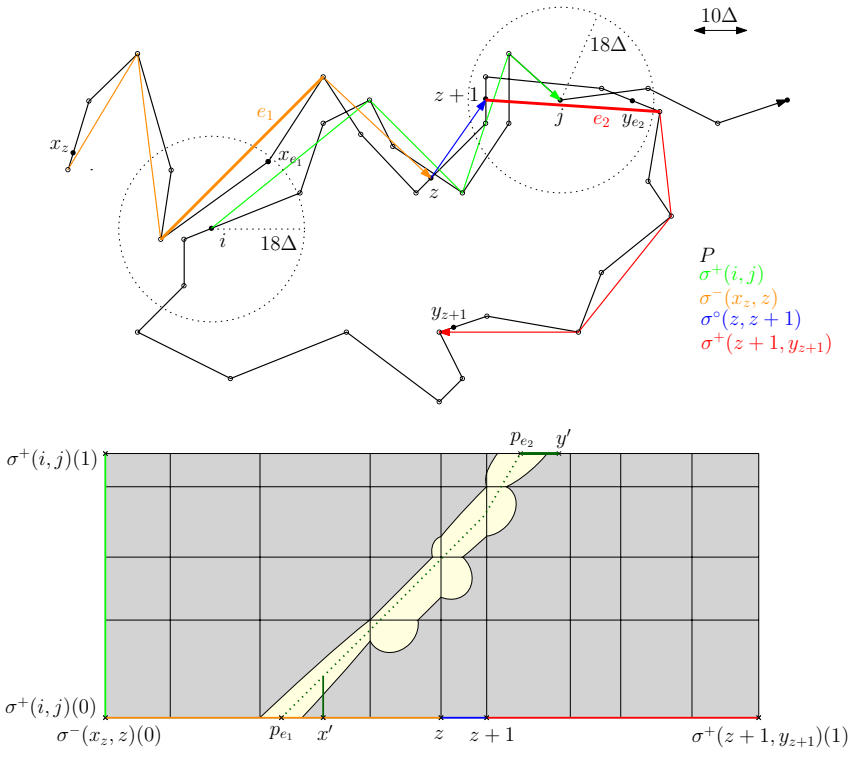


Figure 6 Example for a curve P and breakpoints z, i, j such that the approximation oracle returns "Yes" for the query $z \in r_{i,j}$. The path from p_{e_1} to p_{e_2} in the 10Δ-free space diagram is a monotone increasing path from the active edge e_1 to the active edge e_2. The edges are active since $d(P(t_i), P(t_{x_{e_1}})) \leq 18\Delta$ 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 26.

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 46Δ-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 10Δ-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 10Δ-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 46Δ-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 46Δ-free space of $\kappa_z(x,y)$ and $\sigma^+(i,j)$. For an example of such a path see Figure 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 18Δ, and analogously $d(P(t_j), P(t_y)) \leq 18\Delta$. ◀

4.4.0.4 Running time.

First we analyse the preprocessing time needed to build the data structure for the oracle then we analyse the query time of the oracle.

Since one application of the algorithm of Agarwal et. al. [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 $z\in\{1,\dots,m\}$. To construct the pointers from each edge to the first and last breakpoint on the edge we need 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 2ℓ 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 27.** *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 $O(\ell^2)$.*

4.5 Applying the framework for computing a set cover

In order to apply Theorem 10 directly, we technically need to define a set system based on our data structure. Concretely, we define a new set system 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

$I(z,(i,j))=1$ if the oracle answers "Yes"

$I(z,(i,j))=0$ if the oracle answers "No"

Let $\widetilde{\mathcal{R}}_4$ be the set system consisting of sets of the form

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

With Theorem 27 we immediately get

▶ **Theorem 28.** *One can build a data structure of size $O(m\ell + m^2)$ 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.*

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 24 this directly implies

▶ **Lemma 29.** *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 set system $\widetilde{\mathcal{R}}_4$ we further can derive a lemma corresponding to Lemma 23 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 30.** *Assume there exists a set cover for $\mathcal{R}$ with parameter Δ. 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 10 to the set system $\widetilde{\mathcal{R}}_4$ given by the approximation oracle we merely lose a constant approximation factor for our clustering problem in comparison to the direct application on the set system $\widetilde{\mathcal{R}}_3$. This leads to the following result.

4.6 The result

▶ **Lemma 31.** *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 $\widetilde{O}\left(k\ell^2m^2 + mn\right)$ 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 10 to compute a set cover of $(Z, \widetilde{\mathcal{R}}_4)$. For Theorem 10, we use Theorem 28, $|Z| = m - 1$ and $|\widetilde{\mathcal{R}}_4| = O(m^2)$. Again, the VC-dimension of the dual set system is bounded by $O(\log m)$. ◀

▶ **Theorem 3.** *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 $O(k\log(m)\log^2(m))$ 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 31, Lemma 29 and Lemma 30. ◀

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 Δ 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 1.4.

In this section, we present an approximation algorithm that applies the algorithmic ideas developed in the previous sections to the discretization described in Section 1.5. A direct application of Theorem 3 using Lemma 1, however, leads to a high dependency on the arclength of the input curve, see also the discussion in Section 1.6. 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 set system and the set system oracle wisely.

5.1 The set system

We will again use the set system $\widetilde{\mathcal{R}}_3$ that was defined in Section 4.2. Here we choose $m = \lceil \frac{L}{\epsilon\Delta} \rceil$ breakpoints to ensure that two consecutive breakpoints have a distance of at most $\epsilon\Delta$. The explicit choice of breakpoints was already described in Section 1.5. For the construction of the approximation oracle we then can take advantage of the fact that two consecutive breakpoints are close to each other. This will allow us to achieve better running time results based on the simpler structure of the oracle. A key factor here is the low VC-dimension of the set system that is dual to the set system which is implicitly given by the oracle.

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 $d_F(P[t_x, t_y], \sigma^+(i,j)) \leq (14+\epsilon)\Delta$
- **(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 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 10Δ-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 32.** *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 $O(\ell^2)$.*

5.2.0.1 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 33.** *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 $d_F(P[t_x, t_y], \sigma^+(i,j)) \leq (14+\epsilon)\Delta$*
- **(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 10Δ-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_ϵ before q has distance at most $\epsilon\Delta$ to q. Therefore we have by triangle inequality

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

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

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

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_ε after u, we therefore get

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

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 concetation we can get the curve

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

which is a subcurve of P with

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

By the use of triangle inequality, we now get

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

(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 set system 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

$\widetilde{I}(z, (i,j)) = 1$ if the oracle answers "Yes"

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

Let $\widetilde{\mathcal{R}}_5$ be the set system consisting of sets of the form

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

With Theorem 32 we immediately get

▶ **Theorem 34.** *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 set system $\widetilde{\mathcal{R}}_5$ 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 24 this directly implies

▶ **Lemma 35.** *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 $d_F(\kappa_z(x,y),\sigma^+(i,j)) \leq (14+\epsilon)\Delta$. Imitating the proof of Lemma 23 we then get

▶ **Lemma 36.** *Assume there exists a set cover for $\mathcal{R}$ with parameter Δ. 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 $\phi(P,C) \leq (18+\epsilon)\Delta$.*

These results imply that a minium set cover of $\widetilde{\mathcal{R}}_5$ can be used to find an approximate solution for our clustering problem. But to apply Theorem 10 for finding a good set cover, we first need to bound the VC-dimension of the dual of $\widetilde{\mathcal{R}}_5$.

5.3 The VC-dimension

In our proof to bound the VC-dimension of the set system $\widetilde{\mathcal{R}}_5$ and its dual set system, we follow Driemel, Phillips, Psarros, and Nusser [27]. In a nutshell, they use composition arguments in combination with a set of geometric predicates that describe the metric ball of the Fréchet distance which were previously introduced by Afshani and Driemel [1, 2] in the context of range searching. We need to adapt the proof slightly, since the set system is defined based on a partial alignment instead of a complete alignment. We first show a lemma that is analogous to Lemma 9 in [1]. In particular, we show that the output $\widetilde{I}(z,(i,j))$ of the approximation oracle can be determined by the truth value of the following predicates $\mathcal{P}_1,\mathcal{P}_2,\mathcal{P}_3,\mathcal{P}_4$ for $V=\sigma^+(i,j)$ and $W=\kappa_z(x_z,y_{z+1})$. 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 define

- **i)** $(\mathcal{P}_1)_{(i,j)}$ (Vertex-edge (vertical)): 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)** $(\mathcal{P}_2)_{(i,j)}$ (Vertex-edge (horizontal)): 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)** $(\mathcal{P}_3)_{(i,j,t)}$ (Monotonicity (vertical)): 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)** $(\mathcal{P}_4)_{(i,j,t)}$ (Monotonicity (horizontal)): 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 $\|p_2-w_i\| \leq 10\Delta$.

To show our claim we use the following lemma.

▶ **Lemma 37.** *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 $\mathcal{P}_1, \ldots, \mathcal{P}_4$, one can determine if there exists a monotone increasing path in the 10Δ-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.*

The proof of Lemma 37 is analogous to the proof of Lemma 9 in [1] and much of the argumentation can be applied verbatim. We include the proof here for the sake of completeness, since there are some subtle differences.

Proof. As in the proof of Lemma 9 in [1], we first introduce the notion of 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 $\mathscr{C} = ((i_1, j_1), (i_2, j_2), \ldots, (i_k, j_k))$ 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 10Δ-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 $\mathscr{C}$ a set of predicates $\mathscr{P}$. The set of predicates is different from the predicates in [1] and consist of the following.

i) $(\mathcal{P}_1)_{(i,j)} \in \mathscr{P}$ iff $(i, j-1), (i, j) \in \mathscr{C}$.
ii) $(\mathcal{P}_2)_{(i,j)} \in \mathscr{P}$ iff $(i-1, j), (i, j) \in \mathscr{C}$.
iii) $(\mathcal{P}_2)_{(a,1)} \in \mathscr{P}$ and $(\mathcal{P}_2)_{(b,\ell_1)} \in \mathscr{P}$
iv) $(\mathcal{P}_3)_{(i,j,k)} \in \mathscr{P}$ iff $(i, j-1), (i, k) \in \mathscr{C}$ and $j < k$.
v) $(\mathcal{P}_3)_{(a,1,k)} \in \mathscr{P}$ iff $(a, k) \in \mathscr{C}$ and $1 < k$.
vi) $(\mathcal{P}_3)_{(b,j,\ell_1-1)} \in \mathscr{P}$ iff $(b, j) \in \mathscr{C}$ and $j < \ell_1 - 1$.
vii) $(\mathcal{P}_4)_{(i,j,k)} \in \mathscr{P}$ iff $(i-1, j), (k, j) \in \mathscr{C}$ 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 38** ([1], Lemma 10)**.** *Let $\mathscr{C}$ be a feasible sequence of cells and consider a monotonicity predicate $\mathcal{P}$ of the set of predicates $\mathscr{P}$ induced by $\mathscr{C}$. 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 38 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 37, we claim that any feasible path induces a feasible sequence of cells and vice versa. Assume there exists a feasible path π that passes through the sequence of cells $\mathscr{C}$. 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 $\mathscr{C}$. The path π is a feasible path that passed through the cell boundary between these two cells. This implies that the there exists a point on the edge $\overline{w_i w_{i+1}}$ which lies within distance 10Δ 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 $\mathscr{C}$ that lies in a fixed column i and consider the set of predicates $\mathscr{P}' \subseteq \mathscr{P}$ that consists of vertical monotonicity predicates $(\mathcal{P}_3)_{(i,j,k)}$ 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 π 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 π. Since π 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 $\mathscr{P}$ are true. We can make a similar argument for the horizontal monotonicity predicates $(\mathcal{P}_4)_{(i,j,k)}$ for a fixed row j. This shows that a feasible path π that passes through the cells of $\mathscr{C}$ implies that the conjunction of induced predicates $\mathscr{P}$ 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 π 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 $\mathscr{P}$ is true, but there exists no feasible path through the sequence of cells $\mathscr{C}$. 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 $\mathscr{P}$ and by Lemma 38 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 sequences of cells implies a feasible path, as claimed. ◀

Lemma 37 now directly implies the following theorem.

▶ **Theorem 39.** *Given the truth values of all predicates $\mathcal{P}_1, \ldots, \mathcal{P}_4$ for two fixed curves $V = \sigma^+(i, j)$ and $W = \kappa_z(x_z, y_{z+1})$, one can determine the value of $\widetilde{I}(z, (i, j))$.*

We use Theorem 39 to determine the following bound on the VC-dimension of $(Z, \widetilde{\mathcal{R}}_5)$.

▶ **Theorem 40.** *Let $Z = \{1, \ldots, m\}$. The VC-dimension of $(Z, \widetilde{\mathcal{R}}_5)$ and its dual set system are both in $O(d^2\ell^2 \log(d\ell))$.*

The proof of Theorem 40 is analogous to the proof of Theorem 8.3 in [27] and included here for the sake of completeness. For the proof we use VC-dimension bounds for the following set systems.

▶ **Definition 41.** *For any two points $s, t \in \mathbb{R}^d$ and $r \in \mathbb{R}_+$ define the stadium centered at $\overline{st}$ as*

$$D_r(\overline{st}) = \{x \in \mathbb{R}^d \mid \exists p \in \overline{st}, \|p - x\| \leq r\}$$

We further define the monotony sets $M_r(\overline{st}) \subseteq \mathbb{X}_2^d$ as the sets where $\{w_1, w_2\} \in M_r(\overline{st})$ if and only if there exist $p_1, p_2 \in \ell$ where ℓ is the line supported by $\overline{st}$ such that

- $\|w_1 - w_2\| \leq r$ *and* $\|p_2 - w_2\| \leq r$*; and*
- $\langle p_1, t-s\rangle \leq \langle p_2, t-s\rangle$ *(p_1 lies in front of p_2 on ℓ if ℓ is oriented from s to t).*

The resulting set systems are then $(\mathbb{R}^d, \mathscr{D})$ *with* $\mathscr{D} = \{D_r(\overline{st}) \mid s,t \in \mathbb{R}^d, r \in \mathbb{R}_+\}$ *and* $(\mathbb{X}_2^d, \mathscr{M})$ *with* $\mathscr{M} = \{M_r(\overline{st}) \mid s,t \in \mathbb{R}^d, r \in \mathbb{R}_+\}$.

▶ **Theorem 42** ([27], Corollary 6.4)**.** *The VC-dimension of* $(\mathbb{R}^d, \mathscr{D})$ *and its dual set system are both in* $O(d^2)$.

▶ **Theorem 43** ([27], Corollary 8.4)**.** *The VC-dimension of* $(\mathbb{X}_2^d, \mathscr{M})$ *and its dual set system are both in* $O(d^2)$.

With the help of these bounds we can now prove Theorem 40.

Proof of Theorem 40. Let $S \subseteq \{1,\dots,m\}$ be a set of t breakpoints and let $(i,j) \in \{1,\dots,m\}^2$. For for $1 \leq k \leq 4$, let further $\mathcal{P}_k((\sigma^+(i,j), \kappa_z(x_z, y_{z+1}))$ be the set of all possible Predicates of the form $(\mathcal{P}_k)_{(\cdot,\cdot)}$ or $(\mathcal{P}_k)_{(\cdot,\cdot,\cdot)}$ between the curves $\sigma^+(i,j)$ and $\kappa_z(x_z, y_{z+1})$. Due to Theorem 39, we have that the set $\{z \in S \mid \widetilde{I}(z,(i,j)) = 1\}$ is uniquely defined by the sets

$$\bigcup_{k=1}^{4} \bigcup_{z \in S} \mathcal{P}_k(\sigma^+(i,j), \kappa_z(x_z, y_{z+1}))$$

Theorem 42 implies that the number of all possible sets $\bigcup_{z\in S} \mathcal{P}_1(\sigma^+(i,j), \kappa_z(x_z, y_{z+1}))$ and the number of all possible sets $\bigcup_{z\in S} \mathcal{P}_2(\sigma^+(i,j), \kappa_z(x_z, y_{z+1}))$ are both bounded by $(t\ell)^{O(d^2\ell)}$. Furthermore, the number of all possible sets $\bigcup_{z\in S} \mathcal{P}_3(\sigma^+(i,j), \kappa_z(x_z, y_{z+1}))$ and the number of all possible sets $\bigcup_{z\in S} \mathcal{P}_4(\sigma^+(i,j), \kappa_z(x_z, y_{z+1}))$ are both bounded by $(t\ell)^{O(d^2\ell^2)}$ by Theorem 43. The ℓ^2 term arises beause we consider $\Theta(\ell^2)$ pairs v_j, v_t for Predicate $\mathcal{P}_3$ (w_i, w_t for Predicate $\mathcal{P}_4$). Hence, we get

$$2^t \leq (t\ell)^{O(d^2\ell^2)} \implies t = O(d^2\ell^2 \log(d\ell)).$$

◀

5.4 The result

We apply Theorem 10 on the dual of $(\{1,\dots,m\}, \widetilde{\mathcal{R}}_5)$ to get the following result for computing a set cover. We use here that $|\widetilde{\mathcal{R}}_5| = O(m^2)$ and that the resut of Theorem 40 that the VC-dimension of $\widetilde{\mathcal{R}}_5^*$ is in $O(d^2\ell^2\log(d\ell))$.

▶ **Lemma 44.** *Let k be the minimum size of a set cover for $\widetilde{\mathcal{R}}_5$. Let further $m = \lceil \frac{L}{\epsilon\Delta} \rceil$ and $\delta = O(d^2\ell^2\log(d\ell)))$, there exists an algorithm that computes a set cover for $\widetilde{\mathcal{R}}_5$ of size $O(k\delta\log(\delta k))$ with expected running time in $\widetilde{O}(k\ell^2\delta m^2 + mn)$ 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** (Main Theorem)**.** *Let $P : [0,1] \to \mathbb{R}^d$ be a polygonal curve of complexity n, let $\ell \in \mathbb{N}$ and $\Delta, \varepsilon > 0$ be parameters. Assume there exists a set $C^* \subset \mathbb{X}_\ell^d$ of size k, such that $\psi(P, C^*) \leq \Delta$. Let $m = \lceil \frac{L}{\epsilon\Delta} \rceil$ and $\delta = O(d^2\ell^2\log(d\ell)))$, there exists an algorithm that computes a set $C \subset \mathbb{X}_\ell^d$ of size $O(k\delta\log(\delta k))$, such that $\psi(P,C) \leq (18+\epsilon)\Delta$. The algorithm has expected running time in $\widetilde{O}\left(km^2 + mn\right)$ and uses space in $O(n+m)$, where we assume that ℓ and d are constants independent of n.*

Proof. The theorem follows immediately by the combination of Lemma 44, Lemma 35 and Lemma 36. ◀

6 Additional lower bounds for the VC-dimension

In this section we derive bounds on the VC-dimension of the dual set systems in the discrete and continuous case. Consider the set system $\mathcal{R}$ from Section 1.5. The dual set system of $\mathcal{R}$ is the set system $\mathcal{R}^*$ 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 set system is the set system $\mathcal{R}_0^*$ 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 45.** *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 $d+1$.*

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 set system 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 $d+1$. ◀

6.1 Continuous case

We derive a lower bound on the VC-dimension of the dual set system $(\mathbb{X}_\ell^d, \mathcal{R}_0^*)$ in the general case.

▶ **Theorem 46.** *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 ℓ and d. Let $m \in \mathbb{N}$. We construct a curve P with at most $O(4^m)$ vertices such that the set system $(\mathbb{X}_2^2, \mathcal{R}_0^*)$ 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 τ_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 τ_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 7.

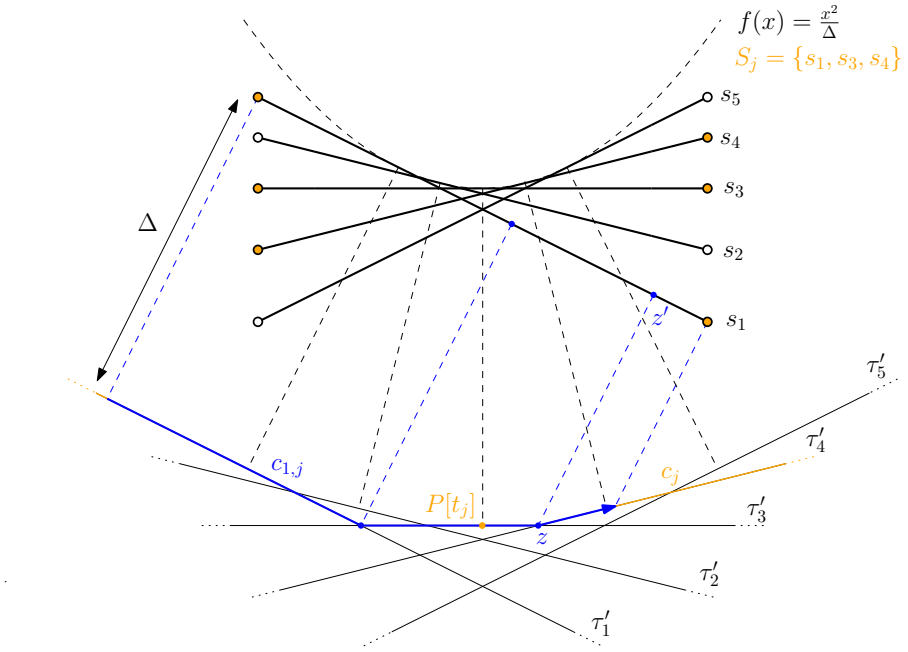


Figure 7 Construction for the case $\ell = 2$ such that the VC-dimension of $(\mathbb{X}^d_\ell, \mathcal{R}^*_0)$ is high.

Consider the power set 2^S. We show that for each subset $S_j \in 2^A$ there exists a curve $c_j \in \mathbb{X}^d_{m+1}$ 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 set system 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 τ'_i be the line parallel to τ_i that lies below τ_i and has distance Δ to τ_i. For $S_j \in 2^S$ we define with o_j the upper contour set of the lines τ'_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 Δ 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 τ'_i. Therefore the ray starting at z' and containing $\overline{z'z}$ hits z before or at the same time as it hits τ'_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 Δ 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 $s_i \in r_{t_j} \cap S$. ◀

6.2 Discrete case

Now we consider the set system $(\mathbb{X}_\ell^d, \mathcal{R}^*)$ that is dual to the set system $\mathcal{R}$, which was introduced in Section 1.5 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 ℓ. 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 set system. It remains to show the lower bound.

▶ **Theorem 47.** *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 ℓ 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 [27] for the VC-dimension of the set system 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

$$A' = A \cap \{x \in \mathbb{R}^2 | d_F(x, P_{A'}) \leq \Delta\} \tag{3}$$

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.

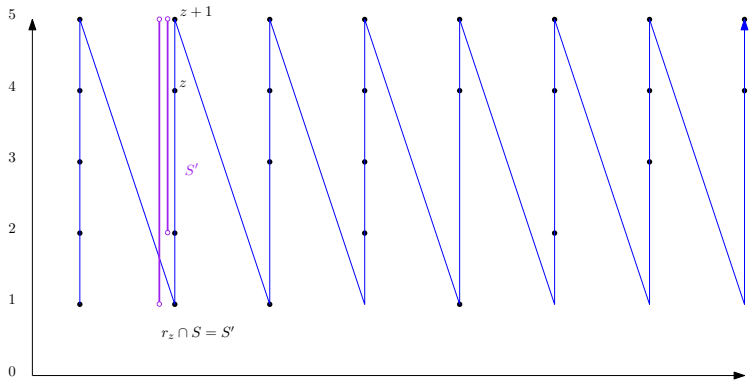


Figure 8 Schematic drawing of $P : [0,1] \to \mathbb{R}$ in the construction for the lower bound to the VC-dimension. Parameters of the construction are $\Delta = \frac{1}{3}$, $\ell = 2$ and $t = 3$. The shattered set of line segments in $\mathbb{R}$ is $S = \{\overline{1,5}\,\overline{2,5}, \overline{3,5}\}$ 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 $r_z \cap S = S'$.

In order to show 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

$$d_F(x, P') = \max_{t \in [0,1]} (x, P'(t)) \tag{4}$$

This implies that for the case $\ell = 1$ our set system $\mathcal{R}^*$ 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

$$\begin{aligned} r_z &= \left\{x \in \mathbb{R}^d \mid \exists i \le z \le j \text{ with } d_F(x, P[t_i, t_j]) \le \Delta\right\} && (5)\\ &= \bigcup_{i \le z < j} \left\{x \in \mathbb{R}^d \mid d_F(x, P[t_i, t_j]) \le \Delta\right\} && (6)\\ &= \left\{x \in \mathbb{R}^d \mid d_F(x, P[t_z, t_{z+1}]) \le \Delta\right\} && (7) \end{aligned}$$

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 $1 \le z < m$, such that $A' = A \cap r_z$ holds as required by (3). Finally, the number of breakpoints we used is $m = 2^{t+1}$ (two breakpoints for each subset of A). Therefore, we have $t \ge \log(m) - 1$. ◀

▶ **Theorem 48.** *For $d \ge 1$ and $\ell \ge 2$ 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 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 Δ. Therefore, the VC-dimension is t. The number of breakpoints m we used is upper-bounded by $(t + 2)2^t$ and therefore $t \geq \Omega(\log m)$. ◀

7 NP-hardness

We note that the problem described in Section 1.4 for $\ell = 1$ is a specific instance of a k-center problem which is NP-complete. However, we require that the input set is a connected polygonal curve. It is tempting to believe that this restriction could make the problem easier. We show in this section that the problem is still NP-hard.

Our reduction is from Planar-Monotone-3SAT, with m_{SAT} clauses and n_{SAT} variables. We show how to construct an instance B of a decision version of our problem given an instance A of Planar-Monotone-3SAT, which is NP-hard [24]. We can assume that A is given by a plane rectilinear bipartite graph between variables and clauses where variables are embedded on the x-axis, edges do not cross the x-axis, clauses are adjacent to two or three variables, and are partitioned between ***positive*** and ***negative*** whether they are embedded in the upper or lower half-plane respectively [25]. All three literals in positive clauses are positive. All three literals in negative clauses are negative. The problem asks whether there exist an assignment from the variables to $\{\texttt{true}, \texttt{false}\}$ such that every positive (negative) clause is adjacent to at least one `true` (`false`) variable.

Problem definition. We define the decision version of our problem as follows. The instance is defined by a polygonal curve P with breakpoints $0 = t_1 < t_2 < \cdots < t_m = 1$, $\Delta \in \mathbb{R}$ and $k \in \mathbb{N}$. The problem asks whether there exist a set C of k points, such that $\phi(P, C) \leq \Delta$. Note that this is equivalent to requiring

$$\max_{i \in \{1,\ldots,m-1\}} \min_{q \in C} \max_{t_i \leq t \leq t_{i+1}} \|P(t) - q\| \leq \Delta$$

A solution C is said to be in ***canonical form*** if every point in C coincides with a breakpoint, i.e., one of the points $P(t_i)$ for $i \in \{1, \ldots, m\}$.

Outline of proof. We first show how to build an instance B of our problem from A. We then show that any positive solution C (a satisfying assignment) of B can be converted in a positive solution C' in canonical form. We also show that C' exists if and only if A has a positive solution, which will conclude our proof. An example of the reduction is shown in Figure 11. The reduction uses paths formed by unit segments called wires. Figure 9 (a) shows circles whose centers represent points in a locally optimal solution. Any optimal solution would choose either the red or the blue circles' centers. A variable is represented by a cycle as shown in Figure 9 (c) formed by 2 vertical paths and 2 "zig-zag" paths connecting their endpoints. The length of such paths depend on the number of times the variable appears in clauses. Clauses are represented by a segment whose endpoint is called a clause vertex shown in Figure 9 (b) as a star. It is next to three wires connected to variable gadgets. Informally, such segment can be covered by a disk centered at a breakpoint contained in one of the wires if the wire carries a `true` signal.

Construction. We modify the embedding in A as follows. Refer to Figure 11. Replace each variable with a cycle in the hexagonal grid separated by a ***separator gadget*** shown in Figure 10 (a). Each cycle contains two vertical edges of length 5 that are all vertically aligned and $4\sqrt{3}r$ apart where r is the maximum number of incidences of the variable in either positive or negative clauses. In order to close each cycle, connect the upper (resp., lower) endpoints of the vertical edges with a "zig-zag" formed by $2r$ edges of length 4 and

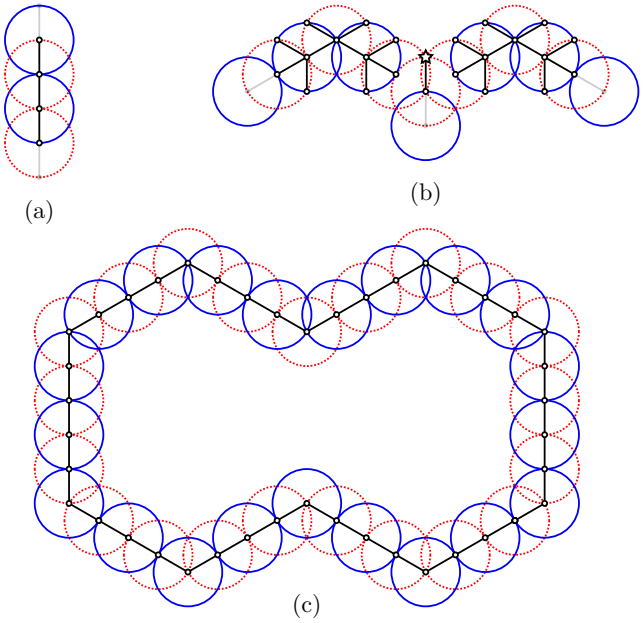
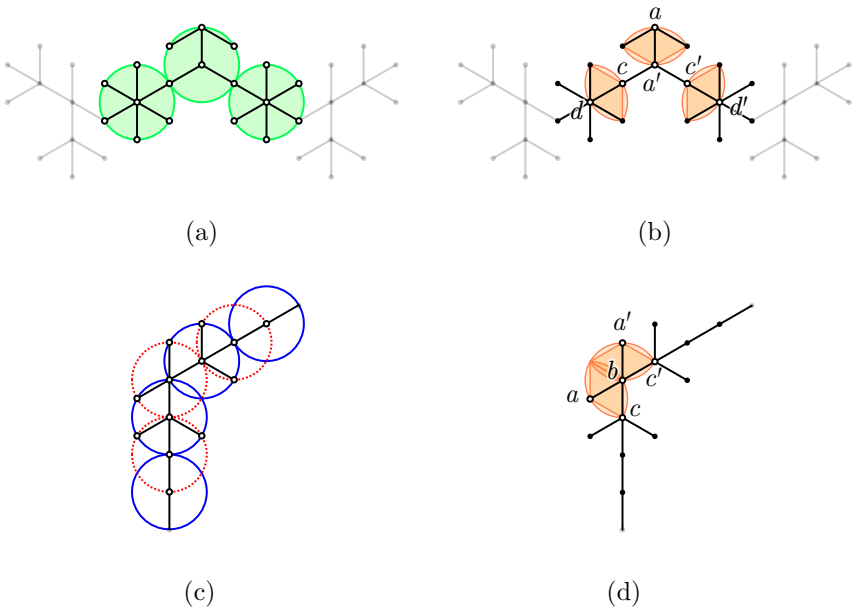


Figure 9 (a) Wire. (b) Clause vertex is shown as a star. (c) Cycle representing a variable with $r = 2$.

slopes $1/\sqrt{3}$ and $-1/\sqrt{3}$ (resp., $-1/\sqrt{3}$ and $1/\sqrt{3}$). We call every even vertex in this upper (resp., lower) "zig-zag" path a ***positive*** (resp., ***negative***) ***literal vertex***. For each clause, the embedding of A allows us to choose two or three literal vertices so that each clause can be connected to literal vertices of their incident variables in a planar way. For each clause we define three ***clause vertices*** as follows. We define positive clauses while negative clauses are defined analogously by reflections. Let p_1, p_2, and p_3 (if it exists) be the three literal vertices, ordered from left to right, to be connected by the clause, and let t be the smallest distance between them. The middle clause vertex c_2 is above p_2 by $t/\sqrt{3}+1$. Let the left clause vertex c_1 (resp., right clause vertex c_3) be $c_2 + (-\sqrt{3}/2, -1/2)$ (resp., $c_2 + (\sqrt{3}/2, -1/2)$). Connect p_2 to c_2 with a vertical edge, and p_1 to c_1 with a convex with 3 bends as in Figure 11 so that the length of the vertical edge is 3. Finally, subdivide each edge into edges of length 1 and at each bend add 6 unit edges as shown in the ***turn gadget*** in Figure 10 (c). We obtain an embedding of a graph G containing only unit edges. We partition the edges of G into two subsets E_1 and E_2 as follows. The set E_1 contains edges in separator gadgets, the 6 added edges in each turn gadget and the edge adjacent to c_2 for each clause. The set E_2 is the set of remaining edges. Define P as the path obtained by an Euler tour defined by a DFS of G. Set $\Delta = 1$ and place a breakpoint on each vertex of P. Finally, set $k = \frac{|E_2|}{2} + 3(n_{SAT} - 1)$. This finalizes the construction.

▶ **Theorem 49.** *Let $P : [0,1] \to \mathbb{R}^2$ be a polygonal curve of complexity n with breakpoints $0 \le t_1, \ldots, t_m \le 1$. It is NP-complete to decide whether there exist a set C of points in $\mathbb{R}^2$ such that $\phi(P, C) \le \Delta$ and $|C| \le k$ for given $\Delta \in \mathbb{R}$ and $k \in \mathbb{N}$.*

Proof. ($\Rightarrow$) We assume that A admits a positive solution, and constructs a positive solution C for B as follows. For each variable x_i, add all the odd (even) points in the spine of the corresponding variable gadget to C if x_i is set to `true` (`false`) in A's solution. Do the same for all wire gadgets and the portion of the clause gadgets corresponding to x_i. For each



(a) (b) (c) (d)

Figure 10 (a) and (b) show the separator gadget in black, and some edges of the adjacent variable gadgets in gray. (c) and (d) show the turn gadget. Disks indicate potential optimal solutions.

separator gadget, add the three green points shown in Figure 10 to C. This finalizes the construction of C. By construction, $|C| = k$, the variable, wire and separator gadgets are covered by disks centered at C. Because every positive (negative) clause in A is adjacent to a variable assigned `true` (`false`), the clause segment is covered by a disk centered at the spine of a incident wire. Then, C is a positive solution for B.

($\Leftarrow$) We assume that B admits a positive solution C, and constructs a positive solution for A. We first show we can construct a canonical solution C' from C with $|C'| \leq |C|$.

Consider the separator gadget in Figure 10 (b). The positions of the centers of unit disks that cover segment aa' form a lune defined by the intersection of the unit disks centered at a and a'. The analogous is true for segments cd and $c'd'$. Such lunes are disjoint, hence C has 3 distinct points, c_1, c_2 and c_3, to cover such segments. Note that they cannot cover segments outside of the separator gadget. We can move them to a, d and d' so that the set of segments that they cover is either the same or a superset of the previously covered segments. The following assumes that (i) every separator gadget is covered by three points in C as in Figure 10 (a).

Consider the turn gadget in Figure 10 (d). Assume that ab and $a'b$ are respectively covered by different points c_1 and c_2 in C. Then, we can move c_1 and c_2 to c and c' while covering the same segments and possibly more. Now, assume that ab and $a'b$ are covered by $c_1 \in C$. Then, c_1 is in the intersection of the two lunes shown in Figure 10 (d). The only segments it can cover are ab, $a'b$, cb and $c'b$. Then we can move it to b without decreasing its coverage. Assuming that turn gadgets are in canonical form, we can apply the moving argument at each remaining segment of P in order to obtain a canonical solution C', moving each point $c \in C$ to a breakpoint.

By (i), C' has $\frac{|E_2|}{2}$ points to cover segments with endpoints at literal vertices. Note that each point can cover at most 2 edges in E_2. Then each point must cover exactly 2 edges in E_2. It follows that, for each variable in A, C' contains points at either all positive literal vertices and none at negative vertices, or all negative literal vertices and none at positive

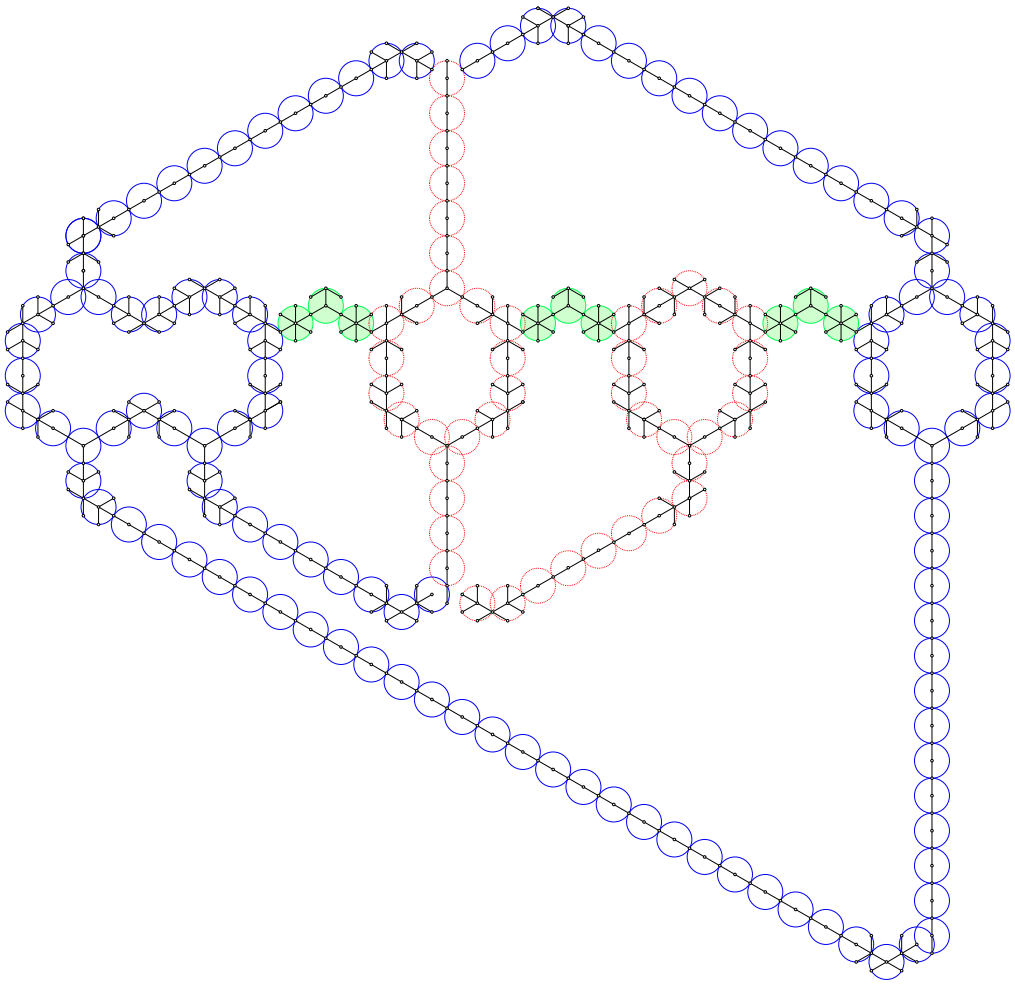


Figure 11 Example of reduction from $(x_1 \vee x_2 \vee x_4) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_4})$. The centers of the disks are an optimal solution to the instance.

vertices. Because the clause segments are covered, the clauses of A are satisfied, and we can obtain a solution for A. That concludes the proof of NP-hardness. The problem is in NP since verifying whether a given set C is a solution for our problem can be done by computing the Δ-free space diagrams for each curve in C and P, and greedily partitioning P, verifying whether it is covered. ◀

8 Future directions

We think that the definition of subtrajectory clustering that we studied in this paper (see Section 1.4) captures a fundamental problem that arises in many applications. However, one may argue that there are many other variants of subtrajectory clustering that arise from application-specific considerations, see also the discussion of related work in Section 1.1. We expect that our general approach and our problem definition can be applied to many 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 a subject for future research.
(2) Covering with gaps: One might be interested in a problem variant where not the entire curve needs to be covered, but only a certain fraction of the curve. It would be interesting to analyze our techniques in this setting.
(3) Input curves: In this paper, 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 the general approach to the discrete problem, where breakpoints are given with the input, is still applicable, if the Fréchet distance is replaced by some other distance measure that satisfies the triangle inequality.

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 optimal center curve. In any case, we think that some form of controlled regularization is necessary in the problem definition.

9 Acknowledgements

This research was initiated at the Eighth Annual Workshop on Geometry and Graphs, held at the Bellairs Research Institute in Barbados, January 31 – February 7, 2020. The authors are grateful to the organizers and to the participants of this workshop. Anne Driemel acknowledges funding from the DFG (project nr. 313421352). Erin Chambers acknowledges funding from the National Science Foundation under grants AF-1907612, AF-2106672, and DBI-1759807.

References

1 Peyman Afshani and Anne Driemel. On the complexity of range searching among curves. *CoRR*, arXiv:1707.04789v1, 2017. `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*, SIGMOD/PODS '18, page 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 Pankaj K Agarwal and Jiangwei Pan. Near-linear algorithms for geometric hitting sets and set covers. *Discrete & Computational Geometry*, 63(2):460–482, 2020. doi:10.1007/s00454-019-00099-6.

6 Pankaj K Agarwal and Cecilia M Procopiuc. Approximation algorithms for projective clustering. *Journal of Algorithms*, 46(2):115 – 139, 2003. URL: http://www.sciencedirect.com/science/article/pii/S019667740200295X, doi:https://doi.org/10.1016/S0196-6774(02)00295-X.

7 Pankaj K Agarwal and Micha Sharir. Efficient algorithms for geometric optimization. *ACM Computing Surveys (CSUR)*, 30(4):412–458, 1998. doi:10.1145/299917.299918.

8 Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry and Applications*, 5:75–91, 1995. doi:10.1142/S0218195995000064.

9 Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999. doi:10.1017/CBO9780511624216.

10 Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(6):121–164, 2012. URL: http://www.theoryofcomputing.org/articles/v008a006, doi:10.4086/toc.2012.v008a006.

11 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.

12 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 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:16, 2022. URL: https://drops.dagstuhl.de/opus/volltexte/2022/16966, doi:10.4230/LIPIcs.ESA.2022.28.

13 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*, 2017. doi:10.1145/3139958.3139964.

14 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 *Proceedings of the 4th ACM SIGSPATIAL Workshop on Location-Based Recommendations, Geosocial Networks, and Geoadvertising*, 2020. doi:10.1145/3423334.3431451.

15 Kevin Buchin, Maike Buchin, Joachim Gudmundsson, Maarten Löffler, and Jun Luo. Detecting commuting patterns by clustering subtrajectories. *International Journal of Computational Geometry & Applications*, 21(03):253–282, 2011. doi:10.1142/S0218195911003652.

16 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*, pages 2922–2938, 2019. doi:10.1137/1.9781611975482.181.

17 Maike Buchin, Anne Driemel, and Dennis Rohde. Approximating (k, l)-median clustering for polygonal curves. In *Proceedings of the Thirty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, 2021. doi:10.1137/1.9781611976465.160.

18 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.

19 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.

20 Timothy M. Chan and Qizheng He. Faster approximation algorithms for geometric set cover. In *36th International Symposium on Computational Geometry, SoCG*, pages 27:1–27:14, 2020. doi:10.4230/LIPIcs.SoCG.2020.27.

21 Vašek Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979. URL: http://www.jstor.org/stable/3689577, doi:10.1287/moor.4.3.233.

22 Kenneth L. Clarkson. Algorithms for polytope covering and approximation. In *Algorithms and Data Structures*, pages 246–252. Springer, 1993. doi:10.1007/3-540-57155-8_252.

23 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.

24 Mark de Berg and Amirali Khosravi. Optimal binary space partitions in the plane. In *International Computing and Combinatorics Conference*, pages 216–225. Springer, 2010. doi:10.1007/978-3-642-14031-0_25.

25 Erik D Demaine and Sarah Eisenstat. Flattening fixed-angle chains is strongly NP-hard. In *Workshop on Algorithms and Data Structures*, pages 314–325. Springer, 2011. doi:10.1007/978-3-642-22300-6_27.

26 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*, pages 766–785, 2016. doi:10.1137/1.9781611974331.ch55.

27 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.

28 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.

29 Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD, page 226–231. AAAI Press, 1996. URL: https://dl.acm.org/doi/10.5555/3001460.3001507, doi:10.5555/3001460.3001507.

30 H. González-Banos. A randomized art-gallery algorithm for sensor placement. In *Proceedings of the Seventeenth Annual Symposium on Computational Geometry*, SoCG, page 232–240, 2001. doi:10.1145/378583.378674.

31 Joachim Gudmundsson and Sampson Wong. Cubic upper and lower bounds for subtrajectory clustering under the continuous Fréchet distance. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 173–189, 2022. doi:10.1137/1.9781611977073.9.

32 David Haussler and Emo Welzl. Epsilon-nets and simplex range queries. *Discrete & Computational Geometry*, 2(2):127–151, 1987. doi:10.1145/10515.10522.

33 Kenneth Holmqvist, Marcus Nyström, Richard Andersson, Richard Dewhurst, Halszka Jarodzka, and Joost Van de Weijer. *Eye tracking: A comprehensive guide to methods and measures*. OUP Oxford, 2011.

34 Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: a partition-and-group framework. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 593–604, 2007. doi:10.1145/1247480.1247546.

35 Abhinandan Nath and Erin Taylor. k-Median clustering under discrete Fréchet and Hausdorff distances. In *36th International Symposium on Computational Geometry, SoCG*, volume 164, page 58, 2020. doi:10.4230/LIPIcs.SoCG.2020.58.

36 Sen Qiao, Y. Wang, and J. Li. Real-time human gesture grading based on openpose. *10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pages 1–6, 2017. doi:10.1109/CISP-BMEI.2017.8301910.

37 Roniel S. De Sousa, Azzedine Boukerche, and Antonio A. F. Loureiro. Vehicle trajectory similarity: Models, methods, and applications. *ACM Computing Surveys*, 53(5), September 2020. doi:10.1145/3406096.

38 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.

39 Sheng Wang, Zhifeng Bao, J Shane Culpepper, and Gao Cong. A survey on trajectory data management, analytics, and learning. *ACM Computing Surveys (CSUR)*, 54(2):1–36, 2021. doi:10.1145/3440207.

40 Guan Yuan, Penghui Sun, Jie Zhao, Daxing Li, and Canwei Wang. A review of moving object trajectory clustering algorithms. *Artificial Intelligence Review*, 47(1):123–144, 2017. doi:10.1007/s10462-016-9477-7.