A Strongly-Subquadratic $(3+\varepsilon)$-Approximation for the Fréchet Distance for Paths in Metric Spaces

Thijs van der Horst
Department of Information and Computing Sciences, Utrecht University, The Netherlands
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Tim Ophelders
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Abstract

The Fréchet distance is a well-studied distance measure for paths in a metric space. It is mostly studied for paths in d-dimensional Euclidean space. Here, computing the Fréchet distance between two polylines takes time roughly quadratic in the number of vertices. Assuming the strong exponential time hypothesis (SETH), it cannot be approximated to within a factor less than 3 in strongly-subquadratic time. Recently, it was shown that for any $\varepsilon > 0$, there exists a randomized algorithm that can compute a $(7+\varepsilon)$-approximation in strongly-subquadratic expected time [Cheng, Huang, and Zhang; STOC'25]. For polylines with n and m vertices in a Euclidean space of constant dimension, where $n \geq m$, their algorithm takes $O(nm^{0.99}\log(n/\varepsilon))$ time in expectation.

We present a deterministic approximation algorithm that significantly improves upon the approximation factor and running time. Specifically, our algorithm computes a $(3+\varepsilon)$-approximation in $O(nm^{2/3}\log n \cdot \log(\frac{1}{\varepsilon}\log n))$ time. Our algorithm nearly matches the conditional lower bound on the approximation factor implied by SETH. For polylines in $\mathbb{R}$, we present a 3-approximation algorithm that runs in $O(nm^{2/3}\log^{5/3} n)$ time, and exactly matches the conditional lower bound.

For our results, we introduce a general strongly-subquadratic time 3-approximate decision algorithm. This algorithm makes no assumptions on the ambient metric space, and relies only on standard assumptions on the so-called free space of the input paths. Under some mild assumptions, our decision algorithm leads to a $(3+\varepsilon)$-approximation algorithm in general metric spaces. These assumptions hold automatically for polylines in any metric space $(\mathbb{R}^d, L_p)$ with $p \geq 1$.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Fréchet distance, path similarity, approximation algorithm

Digital Object Identifier 10.4230/LIPIcs.ESA.2026.111

Related Version *Full Version*: https://arxiv.org/abs/2607.08893

Funding *Tim Ophelders*: partially supported by the Dutch Research Council (NWO) under project no. VI.Vidi.243.236.

1 Introduction

The Fréchet distance is a well-studied distance measure for paths in a metric space, used to determine the similarity of two paths. Measuring similarity is an important task in for example matching time series in data bases [14], protein alignment [13], and trajectory analysis [18]. The Fréchet distance is more discriminative than, e.g., the Hausdorff distance, as it takes into account the intrinsic ordering of the points on a path.

Alt and Godau [1] were the first to study the computability of the Fréchet distance. They showed that one can compute the Fréchet distance between two polylines in $\mathbb{R}^d$, under some L_p norm with $p \geq 1$, in $O(nm\log n)$ time, where n and m are the number of vertices of the polylines and $n \geq m$. Since their result, only marginal improvements have been made [5, 7], shaving off mere polylogarithmic factors. The lack of significant improvements can be explained by the widely-believed strong exponential time hypothesis

(SETH). Bringmann [3] proved that assuming SETH, the Fréchet distance cannot be computed in strongly-subquadratic time; i.e., $O((nm)^{1-\delta})$ time for any $\delta > 0$. Moreover, for $n = \tilde{\Theta}(m)$, Buchin, Ophelders, and Speckmann [6] showed that the Fréchet distance cannot be approximated better than within a factor 3 in strongly-subquadratic time, even for $d = 1$.

The conditional lower bound raises the question of how well the Fréchet distance can be approximated in strongly-subquadratic time. For polylines in $(\mathbb{R}^d, L_2)$, several approximation algorithms are known [4, 8, 19]. Until recently, the best approximation algorithms presented a trade-off between running time and approximation factor. In particular, for an $O(n^{2-\delta})$-time algorithm, a polynomial approximation factor of n^{δ} was needed [19].

This left the question of whether a constant factor approximation in strongly-subquadratic time is possible. Cheng, Huang, and Zhang [8] came very close to answering this question in 2025, by giving a randomized $(7+\varepsilon)$-approximation algorithm with an expected running time of $O(nm^{0.99}\log(n/\varepsilon))$, for any $\varepsilon > 0$, assuming the dimension d is constant. Very recently, Blank [2] gave the first $(3+\varepsilon)$-approximation algorithm that runs in strongly-subquadratic time when the input complexities are imbalanced; that is, when $m = n^c$ for some $c \in (0, 1)$. Specifically, they presented an algorithm with running time $O((n+m^2)\log(n/\varepsilon))$.

Algorithms for computing the Fréchet distance have also been studied in other settings. For example, Rote [17] and Conradi, Driemel, and Kolbe [10] study the setting where the input paths are not polylines, but (well-behaved) piecewise-smooth curves. Even in this generalization, we can compute the Fréchet distance in near-quadratic time. Cook and Wenk [11] consider polylines in a more restricted metric space, namely that of a simple k-gon endowed with the geodesic distance. Here, again, we can compute the Fréchet distance in roughly quadratic time, specifically $O(k + n^2\log^2 n)$. Maheshwari and Yi [15] consider polylines that lie on a convex polyhedron, and give a polynomial-time algorithm for this case.

Results. We present a new algorithm for approximating the Fréchet distance between two paths in a metric space (X, d). Our algorithm makes no assumptions on the metric space. Instead, it relies only on standard assumptions on the so-called free space of the input paths. These assumptions hold in particular for some commonly used settings, shown in Table 1.

In $(\mathbb{R}^d, L_2)$, our algorithm improves upon the algorithm by Cheng, Huang, and Zhang [8], on several fronts: it is deterministic, it has a better approximation factor, its worst-case (rather than expected) running time is strongly-subquadratic, and beats their expected running time by roughly a factor $m^{1/3-0.01}$. When $m = \Omega((n\log n)^{3/4})$, our algorithm also improves upon the algorithm by Blank [2]. In one-dimensional real space, we improve the approximation factor further to 3, which under SETH is the best approximation factor possible in strongly-subquadratic time. Our algorithm also leads to the first strongly-subquadratic constant-factor approximation for paths in a simple polygon under the geodesic distance.

Our main contribution is a 3-approximate decider, which we present in Section 5. On a high level, our approximate decider resembles that by Cheng, Huang, and Zhang. We briefly discuss some main differences in Section 7. Under some mild input assumptions outlined in

Table 1 Our results for approximating the Fréchet distance between polylines with n and $m \leq n$ vertices.

Space	Metric	Approx. factor	Running time
$\mathbb{R}^{d=O(\log n)}$	L_p for $p \geq 1$	$3+\varepsilon$	$O(nm^{2/3}\cdot\log^{4/3} n\cdot\log(\frac{1}{\varepsilon}\log n))$
$\mathbb{R}^{d=\Omega(\log n)}$	L_p for $p \geq 1$	$3+\varepsilon$	$O(nm^{2/3}\cdot(d^3+d^2\log^2 n)^{1/3}\cdot\log(\frac{1}{\varepsilon}\log n))$
$\mathbb{R}$	absolute difference	3	$O(nm^{2/3}\cdot\log^2 n)$
simple k-gon	geodesic Euclidean	$3+\varepsilon$	$O(nm^{2/3}\cdot\log^{4/3} n\cdot\log(\frac{1}{\varepsilon}\log n)+k)$

Section 3, our algorithm applies to paths in general metric spaces. We turn our 3-approximate decider into an approximation algorithm for the Fréchet distance through a simple search algorithm, at a slight increase in approximation factor. Section 6 details this step, as well as our results for the settings shown in Table 1. In addition to the results in Table 1, which are space-intensive, we present time and space tradeoffs in Section 5.4. Although our results are presented for the continuous Fréchet distance, we can obtain analogous results for the discrete Fréchet distance. Details about these results are in the full version.

2 Preliminaries

A *path* π from a point p to a point q in metric space (X, d) is a continuous map $\pi\colon [0,1] \to X$ where $\pi(0) = p$ and $\pi(1) = q$. We denote by $\pi[x, x']$ the subpath of π over the domain $[x, x']$.

Fréchet distance. To define the Fréchet distance between paths $\sigma, \tau\colon [0,1] \to X$ in a common metric space (X, d), we first introduce the concept of a matching between σ and τ. We denote by $\mathcal{D}(\sigma, \tau) = [0,1] \times [0,1]$ the product of their parameter spaces. The point $(x, y) \in \mathcal{D}(\sigma, \tau)$ corresponds to the pair of points $(\sigma(x), \tau(y))$ on σ and τ. A *matching* between σ and τ is represented by a pair of nondecreasing surjections $f, g\colon [0,1] \to [0,1]$. The pair (f, g) can be thought of as the bimonotone path $t \mapsto (f(t), g(t))$ from $(0,0)$ to $(1,1)$ in $\mathcal{D}(\sigma, \tau)$. The matching represented by (f, g) consists of the points

$$\{(f(t), g(t)) \mid t \in [0,1]\} \subseteq \mathcal{D}(\sigma, \tau)$$

on this bimonotone path. We say that the points $\sigma(f(t))$ and $\tau(g(t))$ are *matched* by (f, g). The *cost* of a matching between σ and τ is the maximum distance between matched points:

$$\max_{t \in [0,1]} d(\sigma(f(t)), \tau(g(t))).$$

A matching with cost at most δ is called a *δ-matching*. The Fréchet distance between σ and τ, denoted $d_{\mathrm{F}}(\sigma, \tau)$, is the infimum δ for which a δ-matching exists.

Free space. For $\delta \geq 0$, a point $(x, y) \in \mathcal{D}(\sigma, \tau)$ is *δ-free* if $d(\sigma(x), \tau(y)) \leq \delta$. The *$\delta$-free space* $\mathcal{F}_\delta(\sigma, \tau)$ of σ and τ is the subset of $\mathcal{D}(\sigma, \tau)$ consisting of all δ-free points. A point (x, y) can *δ-reach* a point (x', y') with $x \leq x'$ and $y \leq y'$ if there exists a bimonotone path π in $\mathcal{F}_\delta(\sigma, \tau)$ from (x, y) to (x', y'). Alternatively, we say that (x', y') is *δ-reachable* from (x, y). Note that $d_{\mathrm{F}}(\sigma[x, x'], \tau[y, y']) \leq \delta$ if and only if (x, y) can δ-reach (x', y'). In that case, we call π a *δ-matching* between $\sigma[x, x']$ and $\tau[y, y']$.

3 Input assumptions

In this work, we consider algorithms for approximating the Fréchet distance between paths σ and τ in some metric space (X, d). Because the free space between paths in general metric spaces can be quite complex, we make some mild but standard assumptions. Specifically, we require that σ and τ are subdivided so that their free space satisfies a certain property.

Subdivided paths. A path *subdivided* into n subpaths is a path π together with a sequence of *breakpoints* $(x_i)_{i=1}^{n+1}$ with $0 = x_1 < \cdots < x_{n+1} = 1$. If σ and τ are paths subdivided into n and m subpaths respectively, we partition $\mathcal{D}(\sigma, \tau)$ into a grid of $n \times m$ *cells*. If $x_1, \ldots, x_{n+1}$ are the breakpoints of σ, and $y_1, \ldots, y_{m+1}$ are the breakpoints of τ, then the cells of $\mathcal{D}(\sigma, \tau)$ are of the form $[x_i, x_{i+1}] \times [y_j, y_{j+1}]$. The *edges* of the grid are the sides of its cells.

A cell $C = [x_i, x_{i+1}] \times [y_j, y_{j+1}]$ is said to have the *δ-free-reachability* property if, for all δ-free points (x, y) and (x', y') in C, (x, y) can δ-reach (x', y') precisely if $x \leq x'$ and $y \leq y'$. See Figure 1. Such a property (or the stronger property of convexity of the δ-free space in a cell) is used in most classical decision algorithms for computing the Fréchet distance.

▶ Remark 1. If a cell has the δ-free-reachability property, the δ-free space within is ortho-convex (though possibly disconnected). On the other hand, if the δ-free space in a cell is connected and ortho-convex, the cell satisfies the δ-free-reachability property.

▶ **Property 2.** *Let σ and τ be subdivided paths. We say that $\mathcal{F}_\delta(\sigma, \tau)$ has the* free-reachability *property if all its cells have the δ-free-reachability property.*

Aside from the above property on the input paths and their free space, we require access to certain parts of the free space. To keep things general without requiring the entire free space as input, we access parts of it using the following oracle:

▶ **Definition 3.** *A* δ-free space oracle *for subdivided paths σ and τ supports the following queries. Given a cell C of $\mathcal{D}(\sigma, \tau)$ and a horizontal or vertical line ℓ, the oracle reports*
1. *the intersection of $\ell \cap C$ with the δ-free space $\mathcal{F}_\delta(\sigma, \tau)$, as well as*
2. *the intersection of $\ell \cap C$ with the boundary of the δ-free space $\partial\mathcal{F}_\delta(\sigma, \tau)$.*

We assume access to such an oracle with query time $T_O = \Omega(1)$.

▶ Remark 4. If $\mathcal{D}(\sigma, \tau)$ satisfies Property 2, then a δ-free space oracle for query (C, ℓ) reports for **1.** at most a segment on ℓ, and for **2.** at most two segments on ℓ.

4 Computing a surrogate

In this section, we present an algorithm that, given paths σ and τ, roughly speaking can efficiently compute a sequence of subcurves of σ whose concatenation forms a piecewise-continuous path within Fréchet distance δ of τ, if such a sequence exists. In that case, we call the resulting sequence a surrogate of τ in σ. Our primary purpose is to use the surrogates for our 3-approximate decision algorithm in Section 5. However, such surrogates may also be of independent interest. For example, the surrogate may be of much lower complexity than τ, in which the surrogate can be interpreted as a simplification of τ.

We now formally define the concept of a surrogate. Given an integer parameter k, we define a *(k, δ)-surrogate* of τ as a sequence $\mathcal{S} = (\sigma[x_i, x_i'])_{i=1}^k$ of subpaths of σ, for which there exists a sequence of values $0 = y_1 \leq \cdots \leq y_{k+1} = 1$ with $d_{\mathrm{F}}(\sigma[x_i, x_i'], \tau[y_i, y_{i+1}]) \leq \delta$. We refer to a sequence $(f_1, g_1), \ldots, (f_k, g_k)$, where (f_i, g_i) is a δ-matching between σ_i and $\tau[y_i, y_{i+1}]$, as a δ-matching between $\mathcal{S}$ and τ.

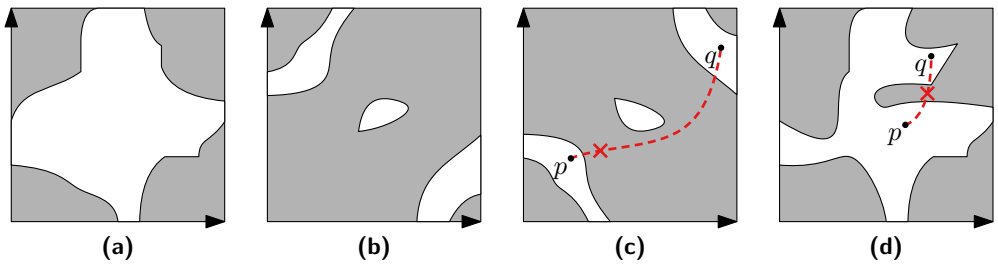

Figure 1 (a–b) Two cells with the δ-free-reachability property, with the δ-free space drawn in white. **(c–d)** Two cells that do not have the property, as witnessed by p and q.

If there does not exist a subcurve of σ within Fréchet distance δ of τ, then it is unnecessary to compute a surrogate representing the entirety of τ. Instead, we require only a surrogate of some prefix $\tau[0, y']$ for which $\mathcal{D}(\sigma, \tau) \cap ([0, 1] \times [0, y'])$ contains all δ-matchings that start on the bottom side of $\mathcal{D}(\sigma, \tau)$ and end on the top or right side. We refer to prefixes $\tau[0, y']$ with this property as *reachability-preserving*. In this section, we present an algorithm that computes a (k, δ)-surrogate of an unspecified reachability-preserving prefix $\tau[0, y']$ of τ. In particular, a suitable value y' is determined by the algorithm. In addition, our algorithm computes a useful representation of a δ-matching between the surrogate and $\tau[0, y']$, which can be queried to determine the subpaths matched to some given point on either σ or τ.

We assume that σ and τ respectively are subdivided with n and m breakpoints, so that $\mathcal{F}_\delta(\sigma, \tau)$ satisfies Property 2. Let k be a given parameter, and let $[y_1, y'_1], \ldots, [y_k, y'_k]$ be a sequence of intervals with $y_1 = 0$, $y'_k = 1$, and $y'_i = y_{i+1}$, such that each subpath $\tau[y_i, y'_i]$ has $O(m/k)$ breakpoints. To compute a (k, δ)-surrogate of a reachability-preserving prefix of τ, together with a suitable representation of a δ-matching between them, we iteratively apply the algorithm of Lemma 5 to these subpaths.

▶ **Lemma 5.** *Let σ' be a given suffix of σ with n' breakpoints. For a given j, let π be the leftmost δ-matching that starts on the bottom side of $\mathcal{D}(\sigma', \tau[y_j, y'_j])$ and ends on either the top or right side. There is an algorithm with the following property:*

- *If π does not exist, the algorithm reports this after $O(T_O \cdot n'm/k)$ time.*
- *If π does exist, and ends in the i-th column of the grid on $\mathcal{D}(\sigma', \tau[y_j, y'_j])$, the algorithm instead takes $O(T_O \cdot im/k)$ time. It then computes some representation of π that can be queried with a horizontal or vertical line ℓ to compute $\ell \cap \pi$ in $O(T_O + \log nm)$ time.*

The algorithm uses $O(n'm/k)$ space in the first case, and $O(im/k)$ in the second.

Proof. We determine a sequence of intersection points between π and the edges of the grid on $\mathcal{D}(\sigma', \tau[y_j, y'_j])$, one for each intersected edge. We use this sequence to represent π. To do so, we use a slight variation on the standard dynamic programming algorithm by Alt and Godau.

Their algorithm computes, for every edge of the grid, the subset of points that are δ-reachable from points on the bottom side of the parameter space. It assumes that a point p on the bottom or left side of a cell C can δ-reach a point q on the top or right side of C if and only if both p and q are δ-free and q lies above and to the right of p. This assumption is satisfied in our setting, since $\mathcal{F}_\delta(\sigma, \tau)$, and thus $\mathcal{F}_\delta(\sigma', \tau[y_j, y'_j])$, satisfies Property 2. We apply Alt and Godau's dynamic programming algorithm column-wise over the grid, and terminate early if an edge on the top side of the grid contains δ-reachable points.

The number of cells considered by the algorithm is $O(im/k)$ (or $O(n'm/k)$ if π does not exist). Each cell is processed in $O(1)$ time once the intersection between $\mathcal{F}_\delta(\sigma', \tau[y_j, y'_j])$ and the boundary of the cell is known, so each cell takes $O(T_O)$ time to process. This gives a total running time of $O(T_O \cdot im/k)$. We compute the intersection points between π and the edges of the grid through backtracking, taking $O(im/k)$ additional time. The algorithm uses $O(1)$ space per cell, so the total space used is $O(im/k)$.

Intersections with lines. We consider a query with a vertical line ℓ, where we seek to compute the intersection $\ell \cap \pi$. Queries with a horizontal line are handled symmetrically. Because π is a δ-matching, it is bimonotone and so the intersection is a vertical line segment. We show how to compute the bottommost intersection point; the topmost point, and thus the intersection, can be computed symmetrically.

Let $\ell = \{x\} \times (-\infty, \infty)$ and let $p = (x, y)$ be the bottommost point in $\ell \cap \pi$. We first compute a cell C of $\mathcal{D}(\sigma', \tau[y_j, y'_j])$ that contains p. If no cells contain p, we immediately return that the intersection is empty. We can compute C, or determine that it does not exist, in $O(\log(n'm/k)) = O(\log nm)$ time by performing binary search over the stored intersection points between π and the edges of the grid.

Let π' be the subpath of π inside C. This path is the top-leftmost δ-matching between its endpoints. Hence, it is the composition of a vertical segment, a part of the boundary of $\mathcal{F}_\delta(\sigma', \tau[y_j, y'_j])$, and a horizontal segment. (Any of the three parts may be empty.) Let $(\hat{x}, \hat{y})$ and $(\hat{x}', \hat{y}')$ be the starting and ending points of π'. We distinguish three cases.

If $\hat{x} = x$, we know $p = (x, \hat{y})$. If $(x, \hat{y}')$ is a point on the interior of $\mathcal{F}_\delta(\sigma', \tau[y_j, y'_j])$, then $p = (x, \hat{y}')$. Otherwise, the intersection between ℓ and π' is a subset of the boundary of $\mathcal{F}_\delta(\sigma', \tau[y_j, y'_j])$. We can determine if this is the case in $O(T_O)$ time, by querying the δ-free space oracle to determine if $(x, \hat{y}')$ is an interior point or not. Then, in the third case, p is the bottom endpoint of the top component of $\ell \cap C \cap \partial\mathcal{F}_\delta(\sigma', \tau[y_j, y'_j])$. We compute this point with another query to the δ-free space oracle. ◀

▶ **Theorem 6.** *Let τ be a subdivided path with m breakpoints, such that $\mathcal{F}_\delta(\sigma, \tau)$ satisfies Property 2. Let $k \in [1, m]$ be a given integer parameter. In $O(T_O \cdot nm/k)$ time, and using $O(nm/k)$ space, we can compute the following:*

- *A (k', δ)-surrogate of a reachability-preserving prefix $\tau[0, y']$ of τ, where $k' \leq k$. The paths in the surrogate have $O(n + k')$ breakpoints in total.*
- *A representation of a δ-matching between $\tau[0, y']$ and the surrogate. This representation can be queried with a point on σ or τ, to obtain the subpath of τ, respectively the surrogate, matched to it. Queries take $O(T_O + \log nm)$ time.*

Proof. Iteratively, with l ranging from 1 to k, we apply the algorithm of Lemma 5 to $\tau[y_l, y'_l]$ and a shrinking suffix $\sigma[x, 1]$. We initialize $x = 0$, so that for $l = 1$ we apply the algorithm to $\tau[y_1, y'_1]$ and the entirety of σ. In iteration l, the algorithm computes (a representation of) the leftmost δ-matching π_l that starts on the bottom side of $\mathcal{D}(\sigma[x, 1], \tau[y_l, y'_l])$ and ends on the top or right side of this parameter space (if such a matching exists). If such a matching does not exist, we return (representations of) $\pi_1, \ldots, \pi_{l-1}$. Otherwise, we consider the end point of the δ-matching π_l. If π_l ends at a point (x, y) with $y < y'_l$, or $l = k$, then we return (representations of) $\pi_1, \ldots, \pi_l$. Otherwise, π_l ends at a point (x'_l, y'_l), and we update $x = x'_l$, and proceed with the next iteration, which applies the algorithm of Lemma 5 to $\tau[y_{l+1}, y'_{l+1}]$ and the suffix $\sigma[x'_l, 1]$. The returned δ-matchings $\pi_1, \ldots, \pi_{k'}$ have the following properties:

- π_1 starts on the bottom side of $\mathcal{D}(\sigma, \tau)$,
- each π_l with $l > 1$ starts in a point horizontally right of the last point on π_{l-1}, and
- for any δ-matching π that starts on the bottom side of $\mathcal{D}(\sigma, \tau)$ and ends on the top or right side, every point on π lies horizontally right of a point on some π_l.

These properties imply that the paths correspond to a (k', δ)-surrogate of a reachability-preserving prefix of τ. The total time spent is $O(T_O \cdot nm/k)$, and the algorithm of Lemma 5 uses $O(nm/k)$ space.

Because the subpaths $\sigma_1, \ldots, \sigma_{k'}$ corresponding to the paths $\pi_1, \ldots, \pi_{k'}$ do not pairwise overlap, except possibly at their endpoints, they have $O(n+k')$ breakpoints in total. The paths $\pi_1, \ldots, \pi_{k'}$ can be queried for intersections with horizontal and vertical lines in $O(T_O + \log nm)$ time per query. These representations can immediately be used to compute the subcurve of σ or τ corresponding to a given point on τ or σ, respectively. ◀

5 An approximate decider

Next we outline our strongly-subquadratic 3-approximate decision algorithm. Given $\delta \geq 0$ and paths σ and τ, it reports either that $d_F(\sigma, \tau) \leq 3\delta$, or that $d_F(\sigma, \tau) > \delta$. When $d_F(\sigma, \tau) \in (\delta, 3\delta]$, it may report either answer. We assume σ and τ are subdivided paths with n and m breakpoints, respectively, so that Property 2 is satisfied for $\mathcal{F}_\delta(\sigma, \tau)$, $\mathcal{F}_{2\delta}(\sigma, \sigma)$, and $\mathcal{F}_{2\delta}(\tau, \tau)$.

Our algorithm works by determining whether $(0, 0)$ can δ-reach $(1, 1)$, in an approximate manner. Specifically, it either determines that $(1, 1)$ is 3δ-reachable, in which case we conclude that $d_F(\sigma, \tau) \leq 3\delta$, or it determines that $(1, 1)$ is not δ-reachable, in which case $d_F(\sigma, \tau) > \delta$. To do so, we use the concept of an *approximate exit set*:

▶ **Definition 7** (Entrance and exit set). *An* entrance set S *for* $\mathcal{D}(\sigma, \tau)$ *is a set of points on the bottom and left sides of* $\mathcal{D}(\sigma, \tau)$*, with at most one connected component on each edge of the grid. For* $\delta \geq 0$*, the* δ-exit set $\mathcal{R}_\delta(S)$ *of* S *is the set of all points on the top and right sides of* $\mathcal{D}(\sigma, \tau)$ *that are* δ*-reachable from points in* S*. Since* $\mathcal{D}(\sigma, \tau)$ *satisfies Property 2,* $\mathcal{R}_\delta(S)$ *has complexity* $O(n + m)$ *and consists of at most one segment per edge of the grid. For* $\alpha \geq 1$*, an* (α, δ)-exit set *of* S *is an approximation of* $\mathcal{R}_\delta(S)$*, defined as any set* A *such that* $\mathcal{R}_\delta(S) \subseteq A \subseteq \mathcal{R}_{\alpha\delta}(S)$*, and* A *consists of at most one segment on each edge of the grid.*

Our decision algorithm constructs a $(3, \delta)$-exit set for $\{(0, 0)\}$, and reports whether $(1, 1)$ is in this set. We do so using a data structure for constructing (exact) exit sets with respect to a preprocessed path and a given subpath. We present this data structure in Section 5.1. Then, in Section 5.2, we extend this data structure to handle arbitrary query paths, though in an approximate manner. Finally, in Section 5.3, we preprocess various subpaths of σ and τ into these data structures, and use them to construct a $(3, \delta)$-exit set for $\{(0, 0)\}$. See Figure 2 for an illustration of the algorithm.

5.1 A data structure for exact exit sets

In this section we develop our data structure for constructing exact exit sets with respect to a preprocessed path σ and a given subpath of σ. Let σ be a subdivided path with n breakpoints, and let $\delta \geq 0$ be a parameter for which $\mathcal{F}_\delta(\sigma, \sigma)$ satisfies Property 2. We present a data structure built on σ and δ that, given an arbitrary subpath σ' of σ and an entrance set S for $\mathcal{D}(\sigma, \sigma')$, efficiently reports the (exact) δ-exit set $\mathcal{R}_{2\delta}(S)$. This data structure is used in steps (3a) and (3b) of Figure 2.

We make use of a structure we call a *reachability map*. Let σ' be an arbitrary subpath of σ, with $m \leq n$ breakpoints. Let BL and TR be the unions of the bottom and left, respectively top and right, sides of $\mathcal{D}(\sigma, \sigma')$. Let $\mathcal{R}$ be the δ-exit set of the set of all δ-free points on BL. The *δ-reachability map* $RM_\delta(\sigma, \sigma')$ of $\mathcal{D}(\sigma, \sigma')$ consists of a pair of functions $\alpha, \beta \colon BL \to TR$, together with $\mathcal{R}$, such that a point $p \in BL$ can δ-reach a point $q \in TR$ precisely if $q \in \mathcal{R}$ and q lies between $\alpha(p)$ and $\beta(p)$ on TR.

Alt and Godau [1] present a reachability data structure that is a δ-reachability map, only with a more concrete representation. Their results imply that the functions α and β of $RM_\delta(\sigma, \sigma')$ are piecewise-linear with $O(nm)$ pieces. Thus, we can evaluate these functions for any given point on BL in $O(\log nm) = O(\log n)$ time through e.g. binary search over the pieces. Additionally, Alt and Godau's construction implies that we can construct $RM_\delta(\sigma, \sigma')$ in $O(nm \log n)$ time, given the intersection between $\mathcal{F}_\delta(\sigma, \sigma')$ and the parameter space grid. We can compute this intersection with $O(nm)$ queries to the δ-free space oracle. Alternatively, we can construct $RM_\delta(\sigma, \sigma')$ faster by merging existing reachability maps, which circumvents the need to compute the free space at the grid edges.

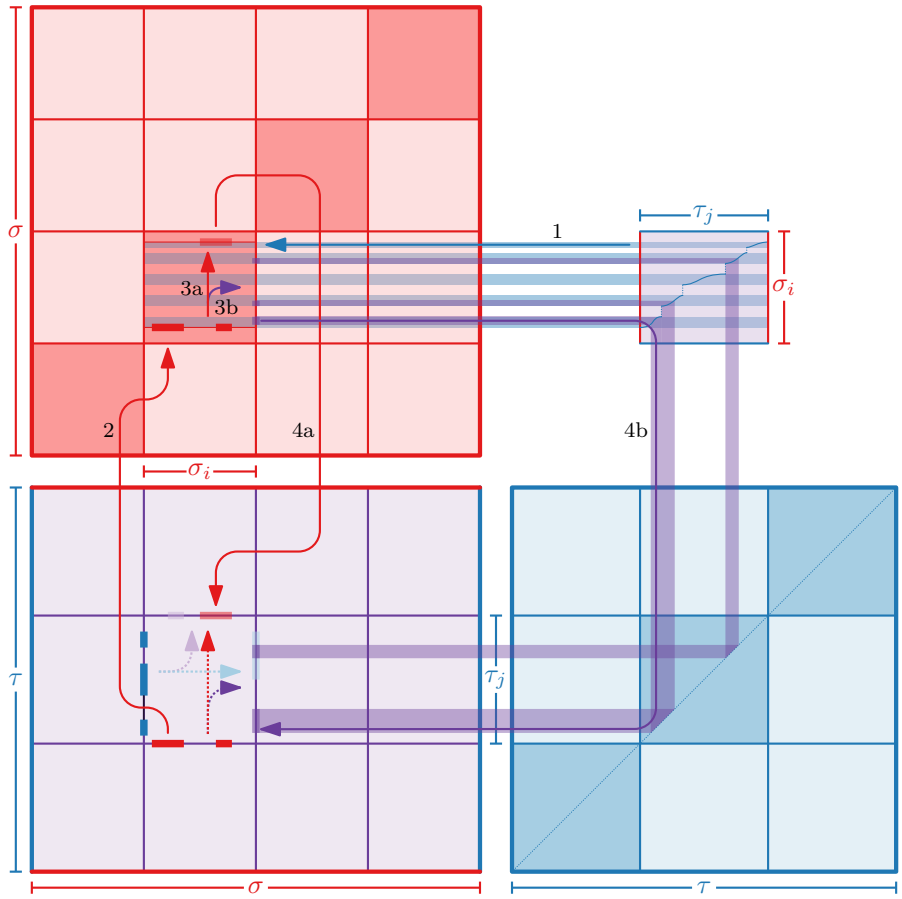

Figure 2 For each i, respectively j, we preprocess the free space $\mathcal{F}_{2\delta}(\sigma_i, \sigma_i)$ (dark red), respectively $\mathcal{F}_{2\delta}(\tau_j, \tau_j)$ (dark blue). For each i and j, we use the corresponding information to 3-approximately propagate reachability across $\mathcal{F}_\delta(\sigma_i, \tau_j)$ in two parts: from the bottom to the top and right (illustrated), and from the left to the top and right. To propagate an entrance set on the bottom, we first compute a sequence $\mathcal{T}$ (1) of subcurves of σ_i that acts as a surrogate for τ_j. Using the preprocessed information on $\mathcal{F}_{2\delta}(\sigma_i, \sigma_i)$, we propagate 2δ-reachability across $\sigma_i \times \mathcal{T}$ to obtain a 2δ-exit set on the top (3a) and right (3b) of $\sigma_i \times \mathcal{T}$. We can translate these exit set back into a $(3, \delta)$-exit set on the top (4a) and right (4b) of $\sigma_i \times \tau_j$.

▶ **Lemma 8.** *The δ-reachability map $RM_\delta(\sigma, \sigma')$ has $O(nm)$ complexity. We can construct $RM_\delta(\sigma, \sigma')$ in $O(nm \cdot (T_O + \log n))$ time, or in $O(nm)$ time when $RM_\delta(\sigma, \sigma'[0, x])$ and $RM_\delta(\sigma, \sigma'[x, 1])$ are given for some $x \in [0, 1]$.*

▶ **Lemma 9.** *Given $RM_\delta(\sigma, \sigma')$, we can compute the δ-exit set of any entrance set in $O(n \log n)$ time.*

Proof. Let S be an entrance set of $\mathcal{D}(\sigma, \sigma')$. For a point $p \in BL$, let $\mathcal{R}_p$ be the region between the points $\alpha(p)$ and $\beta(p)$. By definition of $RM_\delta(\sigma, \sigma')$, we have $\mathcal{R}_\delta(\{p\}) = \mathcal{R} \cap \mathcal{R}_p$ for every point p on BL. We compute $\mathcal{R}_\delta(S) = \bigcup_{p \in S} \mathcal{R}_\delta(\{p\})$ by first computing the union of the sets $\mathcal{R}_p$ for the points $p \in S$. This union has $O(|S|) = O(n)$ components and can be computed in $O(n \log n)$ time through $RM_\delta(\sigma, \sigma')$. Then, we compute the intersection of this union with $\mathcal{R}$ to obtain $\mathcal{R}_\delta(S)$, taking $O(n \log n)$ additional time, by sorting and scanning over the components in the sets. ◀

We wish to compute δ-exit sets with respect to $\mathcal{D}(\sigma, \sigma')$, without specifying σ' during preprocessing. We can answer such queries efficiently using multiple reachability maps, as we describe next. Consider a balanced binary tree $\mathcal{T}$ built on the breakpoints of σ. Each node μ of $\mathcal{T}$ has a subpath σ_μ of σ associated with it, namely the subpath connecting the breakpoints stored in the subtree rooted at μ. We store the δ-reachability map $RM_\delta(\sigma, \sigma_\mu)$ in node μ. The augmented tree $\mathcal{T}$ is the data structure we use to compute exit sets.

▶ **Lemma 10.** *In $O(n^2 \cdot (T_O + \log n))$ time, we can build a data structure of $O(n^2 \log n)$ size that, given an interval $[x, x'] \subseteq [0, 1]$ and an entrance set S of $\mathcal{D}(\sigma, \sigma[x, x'])$, reports the δ-exit set of S with respect to σ and $\sigma[x, x']$ in $O(n \cdot (T_O + \log^2 n))$ time.*

Proof. We first argue the complexity and construction time of the augmented tree $\mathcal{T}$. Every leaf node μ of $\mathcal{T}$ is associated with a subpath σ_μ of σ that has no breakpoints on its interior. Thus, each δ-reachability map $RM_\delta(\sigma, \sigma_\mu)$ has $O(n)$ complexity and takes $O(n \cdot (T_O + \log n))$ time to construct, by Lemma 8. This sums up to $O(n^2)$ size and $O(n^2 \cdot (T_O + \log n))$ construction time for all leaves. For an interior node μ with m nodes in its subtree, the subpath σ_μ has $O(m)$ breakpoints. By Lemma 8, we can therefore compute $RM_\delta(\sigma, \sigma_\mu)$ in $O(nm)$ time by merging the reachability maps stored in the child nodes of μ. The size of the resulting reachability map is also $O(nm)$. The total time spent merging reachability maps is $O(n^2 \log n)$, implying the total construction time of $\mathcal{T}$ is $O(n^2 \cdot (T_O + \log n))$. The total size of the reachability maps is $O(n^2 \log n)$.

To answer a query, suppose we are given an interval $[x, x'] \subseteq [0, 1]$, together with an entrance set S of $\mathcal{D}(\sigma, \sigma[y, y'])$. We seek to compute the δ-exit set of S with respect to σ and $\sigma[x, x']$. Let $x < x_i < \cdots < x_{i'} < x'$ be the breakpoints of $\sigma[x, x']$, and note that $x_i, \ldots, x_{i'}$ are also breakpoints of σ. Thus, the subpath $\sigma[x_i, x_{i'}]$ is the concatenation of $O(\log n)$ subpaths σ_μ stored in $\mathcal{T}$.

We compute the δ-exit set of S by first computing its δ-exit set with respect to σ and $\sigma[x, x_i]$ (or $\sigma[x, x']$ if x_i does not exist). By Lemmas 8 and 9, this takes $O(n \cdot (T_O + \log n))$ time by constructing $RM_\delta(\sigma, \sigma[x, x'])$ and then constructing the exit set. This set induces an entrance set S' for $\mathcal{D}(\sigma, \sigma[x_i, x_{i'}])$. Then, we compute the δ-exit set of S' with respect to $\mathcal{D}(\sigma, \sigma[x, x'])$ by querying $O(\log n)$ reachability maps in $\mathcal{T}$. This takes $O(n \log^2 n)$ time in total. Finally, we turn this exit set into an entrance set S'' for $\mathcal{D}(\sigma, \sigma[x_{i'}, x'])$ and proceed as in the first step. The result is the δ-exit set of S (with respect to σ and σ'). ◀

5.2 Handling arbitrary query paths

Next we extend our data structure for exact exit sets with respect to a preprocessed path σ and subpath of σ, to construct approximate exit sets with respect to σ and an arbitrary query path τ. These queries are exactly steps (3a) and (3b) of Figure 2. Note that we restrict the types of entrance sets we support. Specifically, given σ and δ, we construct a data structure that, given τ and some entrance set S on the bottom side of $\mathcal{D}(\sigma, \tau)$, computes a $(3, \delta)$-exit set for S. Throughout this section, let σ be a subdivided path with n breakpoints, and let $\delta \geq 0$ be a parameter for which $\mathcal{F}_{2\delta}(\sigma, \sigma)$ satisfies Property 2. Our data structure will support querying with any path τ that is subdivided so that $\mathcal{F}_\delta(\sigma, \tau)$ satisfies Property 2.

For preprocessing, we construct the data structure of Lemma 10 on σ and the value 2δ. This data structure allows us to construct exact 2δ-exit sets of any entrance set, with respect to σ and any query subpath of σ. Next we present the query algorithm.

Let τ be the given query path with m breakpoints, and let S be the given entrance set on the bottom side of $\mathcal{D}(\sigma, \tau)$. Observe that to compute a $(3, \delta)$-exit of S, it is sufficient to consider any reachability-preserving prefix of τ, instead of all of τ. That is, any prefix $\tau[0, y']$

of τ for which $\mathcal{D}(\sigma,\tau) \cap ([0,1] \times [0,y'])$ contains all δ-matchings that start on the bottom side of $\mathcal{D}(\sigma,\tau)$ and end on the top or right side. This is because the exact δ-exit set of S is contained in the region $[0,1] \times [0,y']$. Our approach is therefore to compute a surrogate of some reachability-preserving prefix of τ, in terms of subpaths of σ. Then, we can use our data structure for exact exit sets from Lemma 10.

Choose an integer parameter $k \in [1,m]$. We compute a (k,δ)-surrogate of some reachability-preserving prefix $\tau[0,y']$ of τ, using the algorithm of Theorem 6. This takes $O(T_O \cdot nm/k)$ time and uses $O(nm/k)$ space. The algorithm computes a sequence of subpaths $\sigma[x_i, x_i']$, for $i = 1,\ldots,k$, that form a (k,δ)-surrogate of $\tau[0,y']$, as well as some representation of a δ-matching between the surrogate and $\tau[0,y']$. Importantly, this representation can be queried with a point on the surrogate, to obtain the subpath of τ matched to it. Such queries take $O(T_O + \log nm)$ time.

Let Σ be the piecewise-continuous path formed by the surrogate subpaths. We parameterize Σ over $[x_1, x_1'] \cup \cdots \cup [x_k, x_k']$, such that $\Sigma(x) = \sigma(x)$ for all $x \in [x_1, x_1'] \cup \cdots \cup [x_k, x_k']$. We consider the parameter space $\mathcal{D}(\sigma,\Sigma)$ of σ and Σ. The entrance set S of $\mathcal{D}(\sigma,\tau)$, which lies on the bottom side of the parameter space, corresponds to an entrance set S' of $\mathcal{D}(\sigma,\Sigma)$, also on the bottom side. We compute the 2δ-exit set $\mathcal{R}_{2\delta}(S')$ of S' with respect to $\mathcal{D}(\sigma,\Sigma)$. We do so by querying the data structure of Lemma 10 with each surrogate subpath as query subpath. This takes a total of $O(kn \cdot (T_O + \log^2 n))$ time.

Each point $(x,y) \in \mathcal{R}_{2\delta}(S')$ corresponds to a set of points $\{x\} \times \mu(y) \in \mathcal{D}(\sigma,\tau)$, where $\mu(y)$ indexes the subpath of τ that is δ-matched to the point $\Sigma(y) = \sigma(y)$. We define $A = \bigcup_{(x,y)\in\mathcal{R}_{2\delta}(S')} (\{x\} \times \mu(y))$ and show it is a $(3,\delta)$-exit set for S with respect to σ and τ:

▶ **Lemma 11.** *The set A as defined above is a $(3,\delta)$-exit set for S.*

Proof. We assume without loss of generality that Σ has been re-parameterized over $[0,1]$ so that $\mu(y) = \{y\}$ for all $y \in [0,1]$. That is, the point $\Sigma(y)$ is δ-matched to the point $\tau(y)$, and only that point. With this new parameterization, we have $|d(\sigma(x),\Sigma(y)) - d(\sigma(x),\tau(y))| \le \delta$ for all $x,y \in [0,1]$. Thus, if a point $(x,y) \in \mathcal{D}(\sigma,\tau)$ can δ-reach a point $(x',y') \in \mathcal{D}(\sigma,\tau)$, then $(x,y) \in \mathcal{D}(\sigma,\Sigma)$ can 2δ-reach $(x',y') \in \mathcal{D}(\sigma,\Sigma)$. Conversely, if a point $(x,y) \in \mathcal{D}(\sigma,\Sigma)$ can 2δ-reach a point $(x',y') \in \mathcal{D}(\sigma,\Sigma)$, then $(x,y) \in \mathcal{D}(\sigma,\tau)$ can 3δ-reach $(x',y') \in \mathcal{D}(\sigma,\tau)$. It follows that A contains all points that are δ-reachable from points in S, and only points that are 3δ-reachable, making it a $(3,\delta)$-exit set. ◀

What remains is to compute A. Given $\mathcal{R}_{2\delta}(S')$, which consists of $O(n+k) = O(n+m)$ horizontal and vertical line segments, we compute $\mu(p)$ for each endpoint p of these line segments. We do so by querying the representation of the δ-matching between Σ and τ, taking $O(T_O + \log nm)$ time each. The image of μ over a line segment is the convex hull of the image over the endpoints, and so we compute A in $O((n+m) \cdot (T_O + \log nm))$ time.

To summarize, we require $O(T_O \cdot nm/k)$ time (and $O(nm/k)$ space) to compute a surrogate of τ, $O(kn \cdot (T_O + \log^2 n))$ time to compute $\mathcal{R}_{2\delta}(S')$, and $O((n+m) \cdot (T_O + \log nm))$ time for computing A. To optimize running time, we set $k = \sqrt{\frac{T_O \cdot m}{T_O + \log^2 n}} = \tilde{\Theta}(\sqrt{m})$.

▶ **Lemma 12.** *Suppose σ is preprocessed into the data structure of Lemma 10, with the parameter 2δ. Let τ be a subdivided path with m breakpoints, such that $\mathcal{F}_\delta(\sigma,\tau)$ satisfies Property 2. Let $k \in [1,m]$ be an integer parameter. Let S be an entrance set on the bottom side of $\mathcal{D}(\sigma,\tau)$. We can compute a $(3,\delta)$-exit set for S in $O(T_O \cdot nm/k + kn \cdot (T_O + \log^2 n) + (n+m) \cdot (T_O + \log nm))$ time, using $O(nm/k)$ space.*

▶ **Corollary 13.** *In the same context as Lemma 12, setting $k = \tilde{\Theta}(\sqrt{m})$, we can compute a $(3,\delta)$-exit set for S in $\tilde{O}(T_O \cdot n\sqrt{m})$ time and $\tilde{O}(n\sqrt{m})$ space.*

5.3 Constructing an approximate exit set

In this section, we use our data structure for approximate exit sets to efficiently compute approximate exit sets without the need for preprocessing. Specifically, we present an algorithm that, given paths σ and τ, a parameter $\delta \geq 0$, and some entrance set S of $\mathcal{D}(\sigma, \tau)$, computes a $(3, \delta)$-exit set for S. In particular, this algorithm is a 3-approximate decision algorithm.

▶ **Theorem 14.** *Let σ and τ be subdivided paths with n and $m \leq n$ breakpoints, respectively. Let $\delta \geq 0$ be a parameter for which $\mathcal{F}_\delta(\sigma, \tau)$, $\mathcal{F}_{2\delta}(\sigma, \sigma)$, and $\mathcal{F}_{2\delta}(\tau, \tau)$ all satisfy Property 2. Let $k_\sigma \in [1, n]$ and $k_\tau \in [1, m]$ be integer parameters. Given an entrance set S of $\mathcal{D}(\sigma, \tau)$, we can compute a $(3, \delta)$-exit set for S in*

$$O\Big((n^2/k_\sigma + m^2/k_\tau)(T_O + \log n) + (n\sqrt{mk_\tau} + m\sqrt{nk_\sigma})(T_O + \sqrt{T_O}\log n)\Big)$$

time and $O((n^2/k_\sigma + m^2/k_\tau)\log n)$ space.

Proof. Let $\hat{n} = n/k_\sigma$ and $\hat{m} = m/k_\tau$. We partition σ (respectively τ) into subpaths $\sigma_1, \ldots, \sigma_{k_\sigma}$ (respectively $\tau_1, \ldots, \tau_{k_\tau}$) with $O(\hat{n})$ (respectively $O(\hat{m})$) breakpoints each. We preprocess each of these subpaths into the data structure of Lemma 10, taking $O(\hat{n}^2 \cdot (T_O + \log \hat{n}))$ time for each σ_i, and $O(\hat{m}^2 \cdot (T_O + \log \hat{m}))$ time for each τ_j. The space used is $O(\hat{n}^2 \log \hat{n})$ for each σ_i, and $O(\hat{m}^2 \log \hat{m})$ for each τ_j. Thus, the total preprocessing time is $O((n^2/k_\sigma + m^2/k_\tau) \cdot (T_O + \log n))$, and the space used is $O((n^2/k_\sigma + m^2/k_\tau)\log n)$.

With the data structures built, we compute a $(3, \delta)$-exit set for the given entrance set S. The partitions of σ and τ induce a partition of $\mathcal{D}(\sigma, \tau)$ into axis-aligned rectangular regions $\mathcal{R}_{i,j}$ that correspond to σ_i and τ_j. See Figure 2 (bottom left). We process these regions column-wise, from bottom to top. For each region $\mathcal{R}_{i,j}$, we compute a $(3, \delta)$-exit set $A_{i,j}$ for S with respect to the prefixes of σ and τ up to and including σ_i and τ_j. Given such exit sets $A_{i-1,j}$ and $A_{i,j-1}$ (where $A_{0,j} = \emptyset$ and $A_{i,0} = \emptyset$), we compute $A_{i,j}$ as follows.

The set $S \cup A_{i-1,j}$ induces an entrance set S_ℓ on the left side of $\mathcal{R}_{i,j}$, and $S \cup A_{i,j-1}$ induces an entrance set S_b on the bottom side. Thus, we can query the data structures built on σ_i and τ_j with S_b and S_ℓ, respectively, to obtain $(3, \delta)$-exit sets for S_b and S_ℓ, whose union is $A_{i,j}$. The queries allow specifying a parameter k in $[1, n]$ or $[1, m]$, respectively, to influence the running time and space usage of the query algorithm. For querying the data structure built on σ_i, we set $k = \sqrt{\frac{T_O \cdot m}{T_O + \log^2 n}}$, and for querying the data structure built on τ_j, we set $k = \sqrt{\frac{T_O \cdot n}{T_O + \log^2 m}}$. This results in the queries running in

$$O\Big((\hat{n} + \hat{m})(T_O + \log \hat{n}\hat{m}) + \hat{n}\sqrt{\hat{m}}(T_O + \sqrt{T_O}\log \hat{n})\Big) \text{ and}$$
$$O\Big((\hat{n} + \hat{m})(T_O + \log \hat{n}\hat{m}) + \hat{m}\sqrt{\hat{n}}(T_O + \sqrt{T_O}\log \hat{m})\Big)$$

time, respectively, using $O(\hat{n}\sqrt{\hat{m}} + \hat{m}\sqrt{\hat{n}})$ space. Note that the space used by each individual query is dominated by the total space used by the space used by the larger of the queried data structures, and thus can be disregarded. We obtain $A_{i,j}$ by taking the union of the computed $(3, \delta)$-exit sets in $O(\hat{n} + \hat{m})$ extra time.

After preprocessing the subpaths σ_i and τ_j into the data structures, our algorithm makes $k_\sigma \cdot k_\tau$ queries to these data structures. These queries in total take time

$$O((nk_\tau + mk_\sigma)(T_O + \log nm) + (n\sqrt{mk_\tau} + m\sqrt{nk_\sigma})(T_O + \sqrt{T_O}\log n)).$$

Since $m \leq n$, we have $\log nm = O(\log n)$. Also, since $k_\sigma \leq \sqrt{nk_\sigma}$ and $k_\tau \leq \sqrt{mk_\tau}$, we have $(nk_\tau + mk_\sigma)(T_O + \log nm) = O((n\sqrt{mk_\tau} + m\sqrt{nk_\sigma})(T_O + \sqrt{T_O}\log n))$. Including the time spent precomputing the data structures, we arrive at the claimed running time. ◀

▶ **Corollary 15.** *In the same context as Theorem 14, setting* $k_\sigma \approx \frac{n}{m^{2/3}} \cdot \frac{(T_O+\log n)^{2/3}}{(T_O^2+T_O\log^2 n)^{1/3}}$ *and* $k_\tau \approx \frac{m}{n^{2/3}} \cdot \frac{(T_O+\log n)^{2/3}}{(T_O^2+T_O\log^2 n)^{1/3}}$*, we can decide whether* $d_{\mathrm{F}}(\sigma,\tau) \leq 3\delta$ *or* $d_{\mathrm{F}}(\sigma,\tau) > \delta$ *in time* $O(nm^{2/3}\cdot(T_O^3+T_O^2\log^2 n+T_O\log^3 n)^{1/3})$ *and space* $O(nm^{2/3}\cdot(T_O^3+T_O^2\log^2 n+T_O\log^3 n)^{1/3})$.

5.4 Space-efficiency

Although we aimed to optimize time complexity in Corollary 15, the corresponding space complexity is likely impractical. The space usage of our approximate decider is dominated by the precomputed data structures for $\mathcal{F}_{2\delta}(\sigma_i,\sigma_i)$ for $1 \leq i \leq k_\sigma$ and $\mathcal{F}_{2\delta}(\tau_j,\tau_j)$ for $1 \leq j \leq k_\tau$. In total, the data structures for σ use $\tilde{\Theta}(n^2/k_\sigma)$ space, and those for τ use $\tilde{\Theta}(m^2/k_\tau)$ space.

Our algorithm propagates reachability across $\sigma_i \times \tau_j$ for each pair (i,j). For the correctness of the algorithm, the order in which these pairs are processed is largely irrelevant. The only constraint is that the pair (i,j) should not be handled before $(i-1,j)$ and $(i,j-1)$.

In particular, we can propagate reachability using two nested for-loops, the outer loop iterating over j, and the inner loop over i. This way, for each j, the data structure for $\mathcal{F}_{2\delta}(\tau_j,\tau_j)$ is only necessary during a single iteration of the outer loop, and we can compute it at the start of that iteration, and discard it at the end of the iteration. Doing so, the total running time remains $\tilde{O}(T_O \cdot (n^2/k_\sigma + m^2/k_\tau + n\sqrt{mk_\tau} + m\sqrt{nk_\sigma}))$, but the total space usage reduces to $\tilde{O}(n^2/k_\sigma + (m/k_\tau)^2)$. For any $\varepsilon \in [0,\frac{1}{3}]$, selecting $k_\sigma = \Theta(n/m^{2\varepsilon})$ and $k_\tau = \Theta(m^{1-2\varepsilon})$ yields a 3-approximate decider using $\tilde{O}(T_O nm^{1-\varepsilon})$ time and $\tilde{O}(nm^{2\varepsilon})$ space.

We can moreover recompute the data structure for σ_i in each iteration of the inner loop. Doing so results in $\tilde{O}(T_O\cdot(\frac{m^2}{k_\tau}+\frac{n}{k_\sigma}\frac{m}{k_\tau}(\frac{n}{k_\sigma})^2+n\sqrt{mk_\tau}+m\sqrt{nk_\sigma}))$ time and $\tilde{O}(n+(\frac{n}{k_\sigma})^2+(\frac{m}{k_\tau})^2)$ space. For any $\varepsilon \in [0,\frac{2}{9}]$, setting $k_\sigma = \Theta(n/m^{2\varepsilon})$ and $k_\tau = \Theta(m^{1-2\varepsilon})$ yields a 3-approximate decider using $\tilde{O}(T_O nm^{1-\varepsilon})$ time and $\tilde{O}(n)$ space. Plugging in $\varepsilon = \frac{2}{9}$ gives $k_\sigma = \Theta(n/m^{\frac{4}{9}})$ and $k_\tau = \Theta(m^{\frac{5}{9}})$, resulting in $\tilde{O}(T_O nm^{\frac{7}{9}})$ time and $\tilde{O}(n)$ space.

6 Approximating the Fréchet distance

We now discuss how to approximate the Fréchet distance between two paths σ and τ in some metric space (X,d). In Section 6.1, we use our 3-approximate decision algorithm in a search procedure. In Section 6.2, we discuss the complexities of our algorithm in specific settings.

6.1 Approximating using the approximate decider

When the ambient space is Euclidean and σ and τ are polylines, there exists a black-box algorithm that takes any approximate decision algorithm and computes an approximation of the Fréchet distance, at only a slight increase in approximation factor and running time [9]. A more general procedure that does not rely on a specific metric space is not yet known however. In this section, we present an alternative approach that does not require any structure on the underlying metric space, and needs only one additional oracle for its implementation.

Our algorithm works by first computing a good enough over-estimate of $d_{\mathrm{F}}(\sigma,\tau)$. Then we refine this over-estimate into a $(3+\varepsilon)$-approximation of $d_{\mathrm{F}}(\sigma,\tau)$. To compute a suitable over-estimate δ^+ of $d_{\mathrm{F}}(\sigma,\tau)$, we assume some additional structure on σ and τ. Namely, that the subpaths in their subdivisions are essentially shortest paths. Specifically, we require them to be *r-shortest paths*, where an r-shortest path from a point p to a point q is one with length at most $d(p,q)+r$. By allowing some slack in the length of the paths, rather than requiring them to be shortest paths, our algorithm is more generally applicable. In particular, it makes our algorithm applicable to *length spaces*, where a shortest path between points might not exist, but for any $r>0$, an r-shortest path always exists.

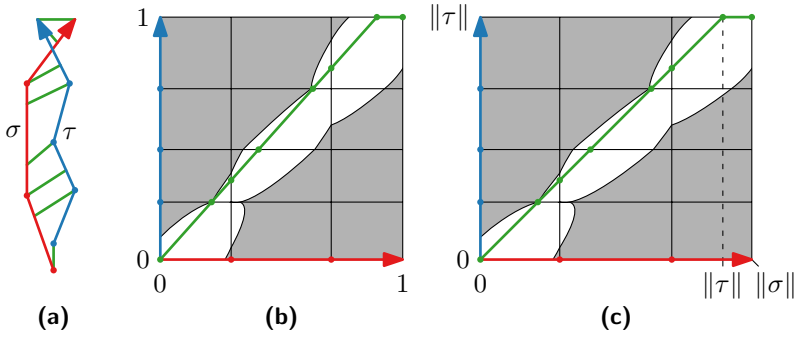

Figure 3 (a) The arclength matching between two polylines σ and τ. Green segments indicate where vertices match to. **(b)** The arclength matching in $\mathcal{D}(\sigma, \tau)$. **(c)** If we parameterize $\mathcal{D}(\sigma, \tau)$ by arclength, the arclength matching has slope 1 until it hits the top or right boundary.

We show that if σ and τ are subdivided into n and m subpaths that are r-shortest paths, the following *arclength matching* between σ and τ has a cost that is only a linear factor higher than the Fréchet distance. We denote by $\|\pi\|$ the arclength of a path π. Because the Fréchet distance is invariant under reparameterization of the input curves, we may without loss of generality assume that for any length ℓ, there is at most one x with $\|\sigma[0, x]\| = \ell$ and at most one y with $\|\tau[0, y]\| = \ell$. Let $x(\ell)$ denote the unique value x such that $\|\sigma[0, x]\| = \ell$, or $x(\ell) = 1$ if $\ell > \|\sigma\|$. Similarly, let $y(\ell)$ denote the unique value y such that $\|\tau[0, y]\| = \ell$, or $y(\ell) = 1$ if $\ell > \|\tau\|$. The arclength matching between σ and τ (see Figure 3) matches the points $x(\ell)$ and $y(\ell)$ for all $\ell \in [0, \max(\|\sigma\|, \|\tau\|)]$. In Lemma 17, we show that this matching has cost at most $2 \cdot \max\{n, m\} \cdot (d_F(\sigma, \tau) + r)$. We use the following auxiliary lemma.

▶ **Lemma 16.** *For any r-shortest path σ and arbitrary path τ, $\|\sigma\| \leq \|\tau\| + 2\, d_F(\sigma, \tau) + r$.*

Proof. By definition of the Fréchet distance, $d(\sigma(0), \tau(0)) \leq d_F(\sigma, \tau)$ and $d(\sigma(1), \tau(1)) \leq d_F(\sigma, \tau)$. By triangle inequality, and the fact that σ is an r-shortest path, we obtain that

$$\begin{aligned}\|\sigma\| &\leq d(\sigma(0), \sigma(1)) + r \\ &\leq d(\sigma(0), \tau(0)) + \|\tau\| + d(\tau(1), \sigma(1)) + r \\ &\leq \|\tau\| + 2\, d_F(\sigma, \tau) + r.\end{aligned}$$

◀

▶ **Lemma 17.** *Let σ and τ be subdivided paths with n and m breakpoints, respectively, such that each subpath between consecutive breakpoints is an r-shortest path. The arclength matching between σ and τ has cost at most $2 \max\{n, m\} \cdot (d_F(\sigma, \tau) + r)$.*

Proof. Let (f, g) be a matching between σ and τ that attains the Fréchet distance. We first prove that for every $t \in [0, 1]$ the arclengths of $\sigma[0, f(t)]$ and $\tau[0, g(t)]$ differ by at most $2 \max\{n - 1, m - 1\} \cdot (d_F(\sigma, \tau) + \alpha)$.

Suppose $\sigma[0, f(t)]$ is longer in arclength than $\tau[0, g(t)]$. Let $0 = x_1 < \cdots < x_{k+1} = f(t)$ be the breakpoints of $\sigma[0, f(t)]$. Denote by τ_i the subcurve of τ matched to $\sigma[x_i, x_{i+1}]$ by (f, g). As $\sigma[x_i, x_{i+1}]$ is an r-shortest path, we obtain from Lemma 16 that $\|\tau_i\| \geq \|\sigma[x_i, x_{i+1}]\| - 2\, d_F(\sigma[x_i, x_{i+1}], \tau_i) - \alpha$. Naturally, $d_F(\sigma[x_i, x_{i+1}], \tau_i) \leq d_F(\sigma, \tau)$, so $\|\tau_i\| \geq \|\sigma[x_i, x_{i+1}]\| - 2\, d_F(\sigma, \tau) - \alpha$. Thus, we have the following bound:

$$\|\tau[0, g(t)]\| = \sum_{i=1}^{k} \|\tau_i\| \geq \|\sigma[0, f(t)]\| - 2k \cdot (d_F(\sigma, \tau) + r).$$

The fact that $k \leq n-1$ and $\sigma[0, f(t)]$ was longer in arclength than $\tau[0, g(t)]$ implies that

$$\|\sigma[0, f(t)]\| - 2(n-1) \cdot (d_{\mathrm{F}}(\sigma, \tau) + r) \leq \|\tau[0, g(t)]\| \leq \|\sigma[0, f(t)]\|.$$

If $\tau[0, g(t)]$ is longer in arclength than $\sigma[0, f(t)]$, it follows from a symmetric argument that

$$\|\tau[0, g(t)]\| - 2(m-1) \cdot (d_{\mathrm{F}}(\sigma, \tau) + r) \leq \|\sigma[0, f(t)]\| \leq \|\tau[0, g(t)]\|.$$

Thus, $\sigma[0, f(t)]$ and $\tau[0, g(t)]$ differ in arclength by at most $2\max\{n-1, m-1\} \cdot (d_{\mathrm{F}}(\sigma, \tau) + r)$, for all $t \in [0, 1]$.

Fix $\ell \in [0, \max(\|\sigma\|, \|\tau\|)]$ and let $t \in [0, 1]$ be such that $f(t) = x(\ell)$. By definition of the arclength matching and the above proof, the arclength of τ between $\tau(g(t))$ and $\tau(y(\ell))$ is at most $2\max\{n-1, m-1\} \cdot (d_{\mathrm{F}}(\sigma, \tau) + r)$. Hence $d(\tau(g(t)), \tau(y(\ell))) \leq 2\max\{n-1, m-1\} \cdot (d_{\mathrm{F}}(\sigma, \tau) + r)$, implying

$$\begin{aligned} d(\sigma(x(\ell)), \tau(y(\ell))) &= d(\sigma(f(t)), \tau(y(\ell))) \\ &\leq d(\sigma(f(t)), \tau(g(t))) + \big|\|\tau[0, g(t)] - \|\tau[0, y(\ell)]\|\big| \\ &\leq d_{\mathrm{F}}(\sigma, \tau) + 2\max\{n-1, m-1\} \cdot (d_{\mathrm{F}}(\sigma, \tau) + r). \\ &\leq 2\max\{n, m\} \cdot (d_{\mathrm{F}}(\sigma, \tau) + r). \end{aligned}$$

◀

With the bound of Lemma 17 on the cost of the arclength matching between σ and τ, we seek to compute this cost. To do so, we require two other oracles; one to query for the distance between points, and one to query for information regarding arclengths.

▶ **Definition 18.** *Let (X, d) be a metric space. A* distance oracle *for (X, d) supports queries of the following form. Given points $p, q \in X$, the oracle reports $d(p, q)$. We assume access to such an oracle with query time $T_D = \Omega(1)$.*

▶ **Definition 19.** *Let σ be a subdivided path and let $x_1, \ldots, x_n$ index its breakpoints. An* arclength oracle *for σ supports the following two queries.* **1.** *Given a value x_i, the oracle reports the arclength of $\sigma[0, x_i]$.* **2.** *Given two consecutive values x_i and x_{i+1}, and a length ℓ, the oracle reports the point $\sigma(x)$ on $\sigma[x_i, x_{i+1}]$ for which $\sigma[0, x]$ has arclength ℓ (if it exists). We assume access to such an oracle with query time $T_A = \Omega(1)$.*

▶ **Lemma 20.** *The cost of the arclength matching between σ and τ can be computed in time $O((T_D + T_A) \cdot (n + m))$.*

Proof. Let (f, g) be the arclength matching between σ and τ. First we make the observation that the cost of (f, g) is attained at a breakpoint of σ or τ. Indeed, let $\sigma[x, x']$ and $\tau[y, y']$ be maximal subpaths matched to each other by (f, g), that contain no breakpoints on their interior. Let $\delta = \max\{d(\sigma(x), \tau(y)), d(\sigma(x'), \tau(y'))\}$. The points (x, y) and (x', y') in $\mathcal{D}(\sigma, \tau)$ are δ-free and lie on the boundary of some cell C. Since we assumed $\mathcal{F}_\delta(\sigma, \tau)$ satisfies Property 2, we immediately obtain that $d_{\mathrm{F}}(\sigma[x, x'], \tau[y, y']) \leq \delta$.

Because the cost of the arclength matching (f, g) is attained at a breakpoint of σ or τ, we can compute its cost as follows. First, we determine for each breakpoint on the paths, the point on the other path that is matched to it. We can do so by scanning the two paths synchronously, so that for any breakpoint on σ, respectively τ, we know the two consecutive breakpoints on τ, respectively σ, containing the point to match to. Then, a query to the arclength oracle yields the result. Thus, we can determine the points that match to the breakpoints of σ and τ in $O(T_A \cdot (n + m))$ time in total. Given these points, we compute the cost of the arclength matching by computing the distance between the breakpoints and the points they are matched to, taking $O(T_D \cdot (n + m))$ time. The maximum of these distances is the cost of the arclength matching. ◀

Let δ^+ be the computed cost of the arclength matching between σ and τ. We proceed to approximate $d_{\mathrm{F}}(\sigma,\tau)$. Let $\delta_i = \delta^+/2^i$ for integers $i \geq 1$. We find a value δ_i where our decision algorithm reports that $\delta_i < d_{\mathrm{F}}(\sigma,\tau)$, but reports that $3\delta_{i-1} \geq d_{\mathrm{F}}(\sigma,\tau)$. Such a value can be found in $O(\log i)$ applications of the decision algorithm, through exponential search. We then perform binary search over $O(1/\varepsilon)$ values in the interval $[\delta_i, 3\delta_{i-1}]$, using our decision algorithm, to refine the approximation into a $(3+\varepsilon)$-approximation of $d_{\mathrm{F}}(\sigma,\tau)$.

We have $\delta_i = \delta^+/2^i \leq 2\max\{n,m\}\cdot(d_{\mathrm{F}}(\sigma,\tau)+r)/2^i$ by Lemma 17. We also have $\delta_i = \Theta(d_{\mathrm{F}}(\sigma,\tau))$ by the above. Hence, $i = O(\log(n + r/\, d_{\mathrm{F}}(\sigma,\tau)))$. Together with the final binary search, our algorithm performs $O(\log(i/\varepsilon))$ calls to our approximate decision algorithm. We summarize in our main theorem:

▶ **Theorem 21.** *Let σ and τ be subdivided paths with n and $m \leq n$ breakpoints, respectively. Suppose they have the following properties:*

- *Every subpath between consecutive breakpoints is r-shortest for some $r \geq 0$, and*
- *For every $\delta \geq 0$, all of $\mathcal{F}_\delta(\sigma,\tau)$, $\mathcal{F}_\delta(\sigma,\sigma)$, and $\mathcal{F}_\delta(\tau,\tau)$ satisfy Property 2.*

For any $\varepsilon > 0$, we can compute a $(3+\varepsilon)$-approximation of $d_{\mathrm{F}}(\sigma,\tau)$ in time

$$O\left((nm^{2/3}\cdot(T_O^3 + T_O^2\log^2 n + T_O\log^3 n)^{1/3}\cdot\log\left(\frac{1}{\varepsilon}\log\left(n+\frac{r}{d_{\mathrm{F}}(\sigma,\tau)}\right)\right) + n\cdot(T_D+T_A)\right)$$

and space $O(nm^{2/3}\cdot(T_O^3 + T_O^2\log^2 n + T_O\log^3 n)^{1/3})$.

6.2 Approximation results for specific metric spaces

Our approximation algorithm is very general, assuming nothing about the ambient metric space, and only some natural assumptions on the input paths. In this section, we list some metric spaces and input assumptions that have already been considered for computing the Fréchet distance, and state what our approximation algorithm implies for those. Throughout this section, we consider polylines σ and τ with n and $m \leq n$ vertices, respectively.

First, let σ and τ be paths in $\mathbb{R}^d$, with as underlying metric the L_p norm for some $p \geq 1$. The intersection between $\mathcal{F}_\delta(\sigma,\tau)$ and a cell of the parameter space is convex for all $\delta \geq 0$ [1], so Property 2 is satisfied. We can trivially build a δ-free space oracle and distance oracle with query times $T_O = O(d)$ and $T_D = O(d)$. As a polyline is a piecewise-linear function, we can preprocess it in linear time to build an arclength oracle with constant query time. Every edge is a 0-shortest path. We obtain:

▶ **Corollary 22.** *For $p \geq 1$, let σ and τ be polylines in $(\mathbb{R}^d, L_p)$, respectively with n and $m \leq n$ vertices. For any $\varepsilon > 0$, we can compute a $(3+\varepsilon)$-approximation of $d_{\mathrm{F}}(\sigma,\tau)$, in $O\big(nm^{2/3}\cdot(d^3 + d^2\log^2 n + d\log^3 n)^{1/3}\cdot\log\left(\frac{1}{\varepsilon}\log n\right)\big)$ time.*

Next, suppose that σ and τ lie in a simple k-gon $P \subseteq \mathbb{R}^2$ endowed with the geodesic L_2 metric. The intersection between $\mathcal{F}_\delta(\sigma,\tau)$ and a cell of the free space is not necessarily convex, but it is connected and ortho-convex [11], so Property 2 is satisfied. In this setting, we can in $O(k)$ time build a δ-free space oracle with query time $T_O = O(\log k)$ [11]. A distance oracle with the same asymptotic complexity exists [12]. For the arclength oracle, the same oracle can be used as for $(\mathbb{R}^d, L_2)$. Every edge is still a 0-shortest path. We obtain:

▶ **Corollary 23.** *Let P be a simple k-gon endowed with the geodesic Euclidean metric. Let σ and τ be polylines in P, respectively with n and $m \leq n$ vertices. For any $\varepsilon > 0$, we can compute a $(3+\varepsilon)$-approximation of $d_{\mathrm{F}}(\sigma,\tau)$ in $O\Big(nm^{2/3}\cdot\log^{4/3} n\cdot\log\left(\frac{1}{\varepsilon}\log n\right) + k\Big)$ time.*

Lastly, we consider the setting where σ and τ lie on the real line $\mathbb{R}$, where the underlying metric is the standard norm. We can improve upon Corollary 22 in this case, to obtain a 3-approximation in roughly the same time bound. In this setting, the Fréchet distance between σ and τ is either the distance between a vertex of σ and a vertex of τ, or half the distance between two vertices of σ or two vertices of τ [1]. We can represent the set of all vertex-vertex distances as the Cartesian sum $X \oplus Y = \{x + y \mid x \in X, y \in Y\}$ of two sets X and Y of $O(n)$ and $O(m)$ values, respectively. Selection in $X \oplus Y$ can be done in $O(n+m)$ time [16], so we can binary search over these values with $O(n+m)$ overhead per step. The smallest value δ for which our decision algorithm reports $d_{\mathrm{F}}(\sigma,\tau) \leq 3\delta$ is then a 3-approximation, as $\delta \leq d_{\mathrm{F}}(\sigma,\tau)$. Thus, we obtain:

▶ **Corollary 24.** *Let σ and τ be polylines in $(\mathbb{R}, |\cdot|)$, respectively with n and $m \leq n$ vertices. We can compute a 3-approximation of $d_{\mathrm{F}}(\sigma,\tau)$ in $O(nm^{2/3}\log^2 n)$ time.*

7 Discussion

Our approximate decider resembles that by Cheng, Huang, and Zhang [8]. They also subdivide σ and τ into smaller subcurves σ_i and τ_j, and build data structures for each. Crucially, our data structures are significantly more time- and space-efficient to build and query. Ultimately, this means that we do not have to subdivide σ and τ as much, resulting in fewer pairs (i,j) over which to propagate reachability. To propagate reachability for a pair (i,j), the algorithm of [8] computes something akin to our surrogates, but theirs are less accurate and rely on randomization, resulting in an expected rather than worst-case strongly subquadratic running time. Finally, the data structures in [8] must compensate for the less accurate surrogates, resulting in a worse approximation factor compared to ours.

Although the approximation factor of our approximate decider is optimal under SETH, it is unclear whether faster constant-factor approximate deciders exist. Let ξ_α be the infimum value ξ for which an α-approximate decider with running time $O(n^\xi)$ exists. Then, assuming the strong exponential time hypothesis, we have $\xi_\alpha = 2$ for all $\alpha < 3$. Our results imply that $\xi_\alpha \leq 5/3$ for all $\alpha \geq 3$. We wish to still gain a better understanding of ξ_α as a function of α.

It is also unclear whether the time and space tradeoffs of Section 5.4 can be improved.

References

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

2 Lotte Blank. Fréchet distance in the imbalanced case. In *Proc. 42nd International Symposium on Computational Geometry (SoCG)*, volume 367 of *LIPIcs*, pages 17:1–17:17, 2026. doi:10.4230/LIPIcs.SOCG.2026.17.

3 Karl Bringmann. Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless SETH fails. In *Proc. 55th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 661–670, 2014. doi:10.1109/FOCS.2014.76.

4 Karl Bringmann and Wolfgang Mulzer. Approximability of the discrete Fréchet distance. *Journal of Computational Geometry*, 7(2):46–76, 2016. doi:10.20382/jocg.v7i2a4.

5 Kevin Buchin, Maike Buchin, Wouter Meulemans, and Wolfgang Mulzer. Four soviets walk the dog: Improved bounds for computing the Fréchet distance. *Discrete & Computational Geometry*, 58(1):180–216, 2017. doi:10.1007/s00454-017-9878-7.

6 Kevin Buchin, Tim Ophelders, and Bettina Speckmann. SETH says: Weak Fréchet distance is faster, but only if it is continuous and in one dimension. In *Proc. 2019 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2887–2901, 2019. doi:10.1137/1.9781611975482.179.

7 Siu-Wing Cheng and Haoqiang Huang. Fréchet distance in subquadratic time. In *Proc. 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 5100–5113, 2025. doi:10.1137/1.9781611978322.173.

8 Siu-Wing Cheng, Haoqiang Huang, and Shuo Zhang. Constant approximation of Fréchet distance in strongly subquadratic time. In *Proc. 57th Annual ACM Symposium on Theory of Computing (STOC)*, pages 2329–2340, 2025. doi:10.1145/3717823.3718157.

9 Connor Colombe and Kyle Fox. Approximating the (continuous) Fréchet distance. In *Proc. 37th International Symposium on Computational Geometry (SoCG)*, pages 26:1–26:14, 2021. doi:10.4230/LIPIcs.SoCG.2021.26.

10 Jacobus Conradi, Anne Driemel, and Benedikt Kolbe. Revisiting the Fréchet distance between piecewise smooth curves. *Computational Geometry*, 129:102194, 2025. doi:10.1016/J.COMGEO.2025.102194.

11 Atlas F. Cook and Carola Wenk. Geodesic Fréchet distance inside a simple polygon. *ACM Transactions on Algorithms*, 7(1):9:1–9:19, 2010. doi:10.1145/1868237.1868247.

12 Leonidas J. Guibas and John Hershberger. Optimal shortest path queries in a simple polygon. *Journal of Computer and System Sciences*, 39(2):126–152, 1989. doi:10.1016/0022-0000(89)90041-X.

13 Minghui Jiang, Ying Xu, and Binhai Zhu. Protein structure-structure alignment with discrete Fréchet distance. *Journal of Bioinformatics and Computational Biology*, 6(1):51–64, 2008. doi:10.1142/S0219720008003278.

14 Xiao-Ying Liu and Chuan-Lun Ren. Fast subsequence matching under time warping in time-series databases. In *Proc. 12th International Conference on Machine Learning and Cybernetics (ICML)*, pages 1584–1590, 2013. doi:10.1109/ICMLC.2013.6890855.

15 Anil Maheshwari and Jiehua Yi. On computing Fréchet distance of two paths on a convex polyhedron. In *(Informal) Proc. 21st European Workshop on Computational Geometry (EuroCG)*, pages 41–44, 2005. URL: http://www.win.tue.nl/EWCG2005/Proceedings/11.pdf.

16 Andranik Mirzaian and Eshrat Arjomandi. Selection in X+Y and matrices with sorted rows and columns. *Information Processing Letters*, 20(1):13–17, 1985. doi:10.1016/0020-0190(85)90123-1.

17 Günter Rote. Computing the Fréchet distance between piecewise smooth curves. *Computational Geometry*, 37(3):162–174, 2007. doi:10.1016/J.COMGEO.2005.01.004.

18 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.

19 Thijs van der Horst, Marc J. van Kreveld, Tim Ophelders, and Bettina Speckmann. Faster Fréchet distance approximation through truncated smoothing. *Journal of Computational Geometry*, 16(2):265–298, 2024. doi:10.20382/JOCG.V16I2A9.