On Computing the k-Shortcut Fréchet Distance

Jacobus Conradi
Department of Computer Science, Universität Bonn, Germany

Anne Driemel
Hausdorff Center for Mathematics, Universität Bonn, Germany

Abstract

The Fréchet distance is a popular measure of dissimilarity for polygonal curves. It is defined as a min-max formulation that considers all direction-preserving continuous bijections of the two curves. Because of its susceptibility to noise, Driemel and Har-Peled introduced the shortcut Fréchet distance in 2012, where one is allowed to take shortcuts along one of the curves, similar to the edit distance for sequences. We analyse the parameterized version of this problem, where the number of shortcuts is bounded by a parameter k. The corresponding decision problem can be stated as follows: Given two polygonal curves T and B of at most n vertices, a parameter k and a distance threshold δ, is it possible to introduce k shortcuts along B such that the Fréchet distance of the resulting curve and the curve T is at most δ? We study this problem for polygonal curves in the plane. We provide a complexity analysis for this problem with the following results: (i) assuming the exponential-time-hypothesis (ETH), there exists no algorithm with running time bounded by $n^{o(k)}$; (ii) there exists a decision algorithm with running time in $\mathcal{O}(kn^{2k+2}\log n)$. In contrast, we also show that efficient approximate decider algorithms are possible, even when k is large. We present a $(3+\varepsilon)$-approximate decider algorithm with running time in $\mathcal{O}(kn^2\log^2 n)$ for fixed ε. In addition, we can show that, if k is a constant and the two curves are c-packed for some constant c, then the approximate decider algorithm runs in near-linear time.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Fréchet distance, Partial similarity, Conditional lower bounds, Approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2022.46

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2202.11534` [19]

Funding This work has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – AA 1111/2-2 (FOR 2535 Anticipating Human Behavior).

1 Introduction

With the prevalence of geographical data collection and usage, the need to process and compare polygonal curves stemming from this data arises. A popular versatile distance measure for polygonal curves is the Fréchet distance [27]. The distance measure is very similar to the well-known Hausdorff distance for geometric sets, except that it takes the ordering of points along the curves into account by minimizing over all possible direction-preserving continuous bijections between the two curves. Intuitively, the distance measure can be defined as follows. Imagine two agents independently traversing the two curves with varying speeds. Let δ be an upper bound on the (Euclidean) distance of the two agents that holds at any point in time during the traversal. The Fréchet distance corresponds to the minimum value of δ that can be attained over all possible traversals.

In practice, the distance measure may be distorted by outliers and measurement errors. As a remedy, partial similarity and distance measures have been introduced which are thought to be more robust. Buchin, Buchin and Wang define a *partial Fréchet distance* [14]

which maximizes the portions of the two curves matched to one-another within some given distance threshold. Driemel and Har-Peled suggested the *shortcut Fréchet distance* [21] in the spirit of the well-known edit distance for strings: a set of non-overlapping subcurves can be replaced by straight edges connecting the endpoints (so-called shortcuts) to minimize the Fréchet distance of the resulting curves. Akitaya, Buchin, Ryvkin and Urhausen [3] introduced a variant of the Fréchet distance, where a certain number of "jumps" (backwards and forwards) are allowed during the traversal of the two curves. We note that it has been acknowledged in the literature that partial dissimilarity measures generally do not satisfy metric properties [12, 25, 28].

It is conceivable that computing a partial dissimilarity based on the Fréchet distance should be more difficult than the standard Fréchet distance because of the structure of the optimization problems involved. We briefly discuss what is known for these problems. The continuous Fréchet distance can be computed in $\mathcal{O}(n^2\text{polylog}(n))$ time for two polygonal curves of n vertices [5, 13]. For the discrete Fréchet distance, slightly subquadratic time is possible, as it can be computed in $\mathcal{O}(\frac{n^2\log\log(n)}{\log(n)})$ time [2]. However, assuming the Strong-Exponential-Time-Hypothesis (SETH), there is no algorithm, in either the discrete or the continuous setting, for any $\varepsilon > 0$, with running time in $\mathcal{O}(n^{2-\varepsilon})$ [9, 11, 15]. Following these works, we know that the complexity of computing the standard Fréchet distance is roughly quadratic in the size of the input, both in the discrete and continuous setting, and this holds for any dimension $d \geq 1$. Strongly subquadratic approximation algorithms are possible for restricted classes of curves [6, 7, 10, 22] and for large approximation factors [17, 18].

Compared to the standard Fréchet distance, the overall picture of the computational complexity of the partial variants is much more heterogeneous. De Carufel et al. [20] showed that the problem of computing the partial Fréchet distance is not solvable by radicals over $\mathbb{Q}$ and that the degree of the polynomial equations involved is unbounded in general. On the other hand, some variants of the partial Fréchet distance can be computed exactly in polynomial time [14]. Computing the shortcut Fréchet distance was shown to be NP-hard [16] when shortcuts are allowed anywhere along the curve. On the other hand, the *discrete Fréchet distance with shortcuts* was shown to be computable in strictly subquadratic time by Avraham et al. [8], which is even faster than computing the standard variant without shortcuts. The variant defined by Akitaya et al. [3] turns out to be NP-hard, but allows for fixed-parameter tractable algorithms.

Our contribution. In this paper, we study the computational complexity of a parameterized version of the shortcut Fréchet distance, where the maximum number of shortcuts that may be introduced on the curve is restricted by a parameter k. We show that assuming the Exponential-Time-Hypothesis (ETH), no fixed-parameter tractable running time is possible with k being the parameter. For polygonal curves in the plane, we present an exponential-time exact algorithm and we show that near-linear time approximation algorithms are possible using certain realistic input assumptions on the two curves.

Previous work. Driemel and Har-Peled [21] introduced the shortcut Fréchet distance and described a near-linear time $(3+\varepsilon)$-approximation algorithm for the class of c-packed curves. However, they only allowed shortcuts that start and end at *vertices* of the base curve. Buchin, Driemel and Speckmann [16] showed that, if shortcuts are allowed *anywhere along the curve*, then the problem of computing the shortcut Fréchet distance exactly is NP-hard via reduction from SUBSET-SUM. They also describe a 3-approximation algorithm for the decision problem with running time in $\mathcal{O}(n^3 \log n)$ for the case that shortcuts may start and

end in the middle of edges. Prior to our work, there has been no study of exact algorithms for the shortcut Fréchet distance, with non-restricted shortcuts. Our analysis of the exact problem therefore closes an important gap in the literature. Obtaining the exact algorithm was surprisingly simple, once the relevant techniques were combined in the right way.

1.1 Basic definitions

▶ **Definition 1** (curve). *A curve T is a continuous map from $[0,1]$ to $\mathbb{R}^d$, where $T(t)$ denotes the point on the curve parameterized by $t \in [0,1]$. For $0 \leq s < t \leq 1$ we denote the subcurve of T from $T(s)$ to $T(t)$ by $T[s,t]$. A polygonal curve of complexity n is given by a sequence of n points in $\mathbb{R}^d$. The curve is then defined as the piecewise linear interpolation between consecutive points.*

▶ **Definition 2** (Fréchet distance). *Given two curves T and B in $\mathbb{R}^d$, their Fréchet distance is defined as*

$$d_{\mathcal{F}}(T,B) = \inf_{f,g:[0,1]\to[0,1]} \max_{t\in[0,1]} \|T(f(t)) - B(g(t))\|,$$

where f and g are monotone, continuous, non-decreasing and surjective. We call a pair of such functions (f,g) a traversal. Any such traversal has the cost $\max_{\alpha\in[0,1]} \|T(f(\alpha)) - B(g(\alpha))\|$ associated to it.

In our definition of the Fréchet distance given above, we follow Alt and Godau [4]. Strictly speaking, this definition does not use bijections as for the sake of convenience the strict monotonicity of f and g is relaxed.

▶ **Definition 3** (k-shortcut curve). *We call a line segment between two arbitrary points $B(s)$ and $B(t)$ of a curve B a shortcut on B, where $s < t$ and denote it by $\overline{B}[s,t]$. A k-shortcut curve of B is the result of replacing k subcurves $B[s_i,t_i]$ of B for $1 \leq i \leq k$ by shortcuts $\overline{B}[s_i,t_i]$ connecting their start and endpoint, with $t_i \leq s_{i+1}$ for $1 \leq i \leq k-1$.*

▶ **Definition 4** (k-shortcut Fréchet Distance). *Given two polygonal curves T and B, their k-shortcut Fréchet distance $d_{\mathcal{S}}^k(T,B)$ is defined as the minimum Fréchet distance between T and any k'-shortcut curve of B for some $0 \leq k' \leq k$. In this context, we call B the base curve (where we take shortcuts) and T the target curve (which we want to minimize the Fréchet distance to).*

1.2 Overview of this paper

In Section 3 we present an exact algorithm for deciding if the k-shortcut Fréchet distance is smaller than a given threshold δ. The algorithm can also be used for the non-parameterized variant by setting $k = n$. Our first main result is the following theorem.

▶ **Theorem 5.** *Let T and B be two polygonal curves in the plane with overall complexity n, together with a value $\delta > 0$. There exists an algorithm with running time in $\mathcal{O}\big(kn^{2k+2}\log n\big)$ and space in $\mathcal{O}\big(kn^{2k+2}\big)$ that decides whether $d_{\mathcal{S}}^k(T,B) \leq \delta$.*

Our algorithm for Theorem 5 iterates over the free space diagram by Alt and Godau [5] in k rounds. Within the free space diagram, a direction-preserving continuous bijection between two curves corresponds to a monotone path starting at $(0,0)$ and ending at $(1,1)$. In each round, we compute the set of points in the parametric space of the two curves that are reachable by using one additional shortcut. For computing the set of eligible shortcuts

spanning a fixed set of edges, we make use of the so-called line-stabbing wedge introduced by Guibas et al. [23]. Line-stabbing wedges were also used in the approximation algorithm by Buchin et al [16]. In our case, since we perform exact computations, the reachable space may be fragmented into a number of components, and this number may grow exponentially with the number of rounds.

In Section 5 we give some evidence that this high complexity due to fragmentation is not an artifact of our algorithm, but may be inherent in the problem itself. For this, we assume that the exponential time hypothesis (ETH) holds. The ETH states that 3-SAT in n variables cannot be solved in $2^{o(n)}$ time [24]. Our second main result is the following conditional lower bound.

▶ **Theorem 6.** *Unless ETH fails, there is no algorithm for the k-shortcut Fréchet distance decision problem in $\mathbb{R}^d$ for $d \geq 2$, with running time $n^{o(k)}$.*

Our conditional lower bound of Theorem 6 is obtained via reduction from a variant of the k-SUM problem, which is called k-Table-SUM. In particular, we construct a $(4k+2)$-shortcut Fréchet distance decision instance for a given k-Table-SUM instance. Our construction is based on the NP-hardness reduction by Buchin, Driemel and Speckmann [16]. Their reduction was from SUBSET-SUM and could not be directly applied to obtain our result. The construction implicitly encodes partial solutions for the SUBSET-SUM instance as reachable intervals on the edges of one of the curves. In this way, each shortcut taken by the optimal solution implements a choice for an element to be included in the sum. The previous reduction implemented this in the form of a binary choice, thereby leading to a number of shortcuts that is linear in n. In our case, the number of shortcuts taken should only depend on k and not n. Therefore, we give a new construction for a choice gadget, that allows to choose an element from a set to be included in a partial solution while using only a constant number of shortcuts for this choice. We remark that a tighter bound can be obtained when assuming the k-SUM hypothesis [1].

In light of the above results, it is interesting to consider approximation algorithms and realistic input assumptions for this problem. In Section 4 we show that there is an efficient approximation algorithm for this problem. If we can assume that the input curves are well-behaved, we even obtain a near-linear time algorithm for constant k. To formalize this, we consider the class of c-packed curves, see also [22].

▶ **Definition 7** (c-packed curves)**.** *For $c > 0$, a curve X is called c-packed if the total length of X inside any ball is bounded by c times the radius of the ball.*

The following is our third main result. Since any polygonal curve of complexity n is c-packed for some $c \leq 2n$, the theorem also implies a running time of $\mathcal{O}\big(kn^2\varepsilon^{-5}(\log^2(n\varepsilon^{-1})\big)$ for polygonal curves in the plane – without any input assumptions.

▶ **Theorem 8.** *Let T and B be two c-packed polygonal curves in the plane with overall complexity n, together with values $0 < \varepsilon \leq 1$ and $\delta > 0$. There exists an algorithm with running time in $\mathcal{O}\big(kcn\varepsilon^{-5}\log^2(n\varepsilon^{-1})\big)$ and space in $\mathcal{O}\big(kcn\varepsilon^{-4}\log^2(\varepsilon^{-1})\big)$ which outputs one of the following: (i) $d_{\mathcal{S}}^k(T,B) \leq (3+\varepsilon)\delta$ or (ii) $d_{\mathcal{S}}^k(T,B) > \delta$. In any case, the output is correct.*

The main ideas that go into the proof of Theorem 8 can be sketched as follows. The first observation is that a highly fragmented reachable space that leads to the high running time of the exact algorithm of Theorem 5 can be approximated by limiting the number of shortcuts that the algorithm may take. To show that the algorithm still takes the right

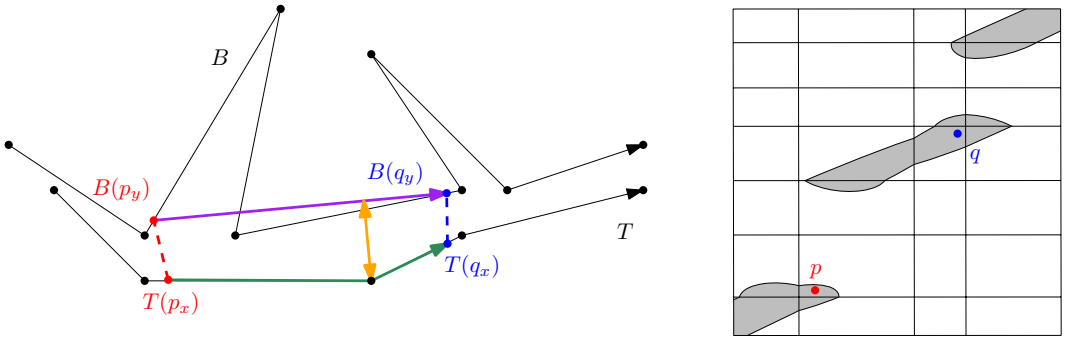


Figure 1 Free space diagram for a base curve B and a target curve T, with the δ-free space in gray. Marked are some grid lines with their corresponding vertex on each curve. The figure also shows a feasible proper tunnel $\tau(p, q)$. The shortcut $\overline{B}[p_y, q_y]$ is shown in purple, and the subcurve $T[p_x, p_y]$ in green. The price of $\tau(p, q)$ is the Fréchet distance of the shortcut and the subcurve.

decisions (within the approximation bounds), we make use of a property of shortcut prices that was first observed by Driemel and Har-Peled [21]. Namely, the price of a shortcut is approximately monotone and it suffices in each round to take the 'shortest' feasible shortcut among all shortcuts that are available. Now, the main challenge as compared to the algorithm in [21] is that this shortcut may still start in the middle of an edge. Thus, we would need to invoke the line-stabbing wedges, but this would be too expensive. Instead, we use a data structure by Driemel and Har-Peled [21] that allows to query the Fréchet distance of a line segment to a subcurve. We combine this with a scheme to simulate an approximation of the output of the line-stabbing wedge with queries to this data structure. In particular, we can approximate the line-stabbing wedge with a convex hull of a set of grid points. However, this is still not enough, as the free-space may have quadratic complexity. To obtain a near-linear running time for small c, we make use of the property of c-packed curves as observed by Driemel, Har-Peled and Wenk [22], that the complexity of the free space diagram of two c-packed curves is only linear in $c \cdot n$ when the curves are appropriately simplified.

2 Preliminaries

Our algorithm uses the free space diagram which was introduced by Alt and Godau [5] for computing the standard Fréchet distance.

▶ **Definition 9** (Free space diagram). *Let T and B be two polygonal curves in $\mathbb{R}^d$. The free space diagram of T and B is the joint parametric space $[0, 1]^2$ together with a not necessarily uniform grid, where each vertical line corresponds to a vertex of T and each horizontal line to a vertex of B (refer to Figure 1). We call the cell of the parametric space corresponding to the ith edge of the target curve and the jth edge of the base curve $C_{i,j}$. The δ-free space of T and B is defined as*

$$\mathcal{D}_\delta(T, B) = \{(x, y) \in [0, 1]^2 \mid \|T(x) - B(y)\| \leq \delta\}$$

This is the set of points in the parametric space whose corresponding points on B and T are at a distance at most δ. Denote by $\mathcal{D}_\delta^{(i,j)}(T, B) = \mathcal{D}_\delta(T, B) \cap C_{i,j}$ the δ-free space inside the cell $C_{i,j}$.

In the following T and B will often be fixed, thus we will simply write $\mathcal{D}_\delta$. It is known that $\mathcal{D}_\delta^{(i,j)}$ is convex and has constant complexity. More precisely, it is an ellipse intersected with the cell $C_{i,j}$. Furthermore the Fréchet distance between two curves is less than or equal

to δ if and only if there exists a monotone path (in x and y) in the free space that starts in the lower left corner $(0,0)$ and ends in the upper right corner $(1,1)$ cf. [5]. In the case of the k-shortcut Fréchet distance we need to also consider shortcuts when traversing the parametric space. When considering any k-shortcut curve B' of B and any traversal (f,g) of B' and T with associated cost δ, then (f,g) induces traversals (f',g') with associated cost at most δ on every shortcut $\overline{B}[s,t]$ and some corresponding subcurve $T[u,v]$ of T. To capture this, we use the notion of *tunnels* which was introduced in [21] and is defined as follows.

▶ **Definition 10** (Tunnel). *A tunnel $\tau(p,q)$ is a pair of points $p=(x_p,y_p)$ and $q=(x_q,y_q)$ in the parametric space of B and T, with $x_p \le x_q$ and $y_p \le y_q$. $\tau(p,q)$ is called feasible if p and q are in $\mathcal{D}_\delta$. We say that a tunnel is proper, if the endpoints of the shortcut do not lie on the same edge of B. We say a tunnel has a price* $\mathrm{prc}(\tau(p,q)) = d_{\mathcal{F}}(T[x_p,x_q],\overline{B}[y_p,y_q])$, *refer to Figure 1.*

▶ **Observation 11.** *Given line segments $\overline{ab}$ and $\overline{cd}$ in $\mathbb{R}^d$, then for the Fréchet distance we have $d_{\mathcal{F}}(\overline{ab},\overline{cd}) = \max(||a-c||,||b-d||)$.*

▶ **Definition 12** (Reachable space). *We define the (δ,s)-reachable free space of T and B*

$$\mathcal{R}_{\delta,s}(T,B) = \{(x_p,y_p) \in [0,1]^2 \mid d^s_{\mathcal{S}}(T[0,x_p],B[0,y_p]) \le \delta\}$$

and again $\mathcal{R}^{(i,j)}_{\delta,s}(T,B) = \mathcal{R}_{\delta,s}(T,B) \cap C_{i,j}$. We call the intersection $\mathcal{R}^{(i,j)}_{\delta,s}(T,B) \cap C_{a,b}$ for any $(a,b) \in \{(i-1,j),(i,j-1),(i+1,j),(i,j+1)\}$ a reachability interval of the cell $C_{i,j}$. In particular for $(a,b) \in \{(i-1,j),(i,j-1)\}$ we call them incoming reachability intervals and for $(a,b) \in \{(i+1,j),(i,j+1)\}$ we call them outgoing reachability intervals.

Note that the reachability intervals for every cell $C_{i,j}$ and s are contained in $\partial C_{i,j}$, and each reachability interval is described by a (possibly empty) single interval, since any two points in the reachability interval can be connected via a monotone path that stays inside the δ-free space. We will simply write $\mathcal{R}_{\delta,s}$ and $\mathcal{R}^{(i,j)}_{\delta,s}$ whenever T and B are fixed. The k-shortcut Fréchet distance of T and B is at most δ if and only if $(1,1) \in \mathcal{R}_{\delta,k}$.

We want to reduce the problem of deciding the k-shortcut Fréchet distance to the problem of deciding on the existence of a monotone path in the free space diagram with k tunnels starting in $(0,0)$ and ending in $(1,1)$. Note that any tunnel $\tau(p,q)$ with $p=(x_p,y_p)$ and $q=(x_q,y_q)$, that is not proper, induces a traversal of $B[y_p,y_q] = \overline{B}[y_p,y_q]$ and $T[x_p,x_q]$. Thus we can omit the tunnel and replace it with a monotone path from p to q in $\mathcal{D}_\delta$. Therefore, in the following, we only consider monotone paths with proper tunnels.

▶ **Definition 13** (Monotone path with tunnels). *A monotone path with k proper tunnels in the δ-free space of two curves consists of $k+1$ monotone (in x and y) paths in the δ-free space from s_i to t_i for $1 \le i \le k+1$, with $s_1=(0,0)$, such that t_i lies to the left and below s_{i+1}, for $1 \le i \le k$. The k proper tunnels have the form $\tau(t_i,s_{i+1})$ for $1 \le i \le k$.*

▶ **Observation 14.** *Let T and B be two polygonal curves. The set $\mathcal{R}_{\delta,s}(T,B)$ is exactly the set of points $p \in \mathcal{D}_\delta(T,B)$ such that there exists a monotone path ending in p with at most s proper tunnels, each of price at most δ. (By definition, these paths have to start at $(0,0)$).*

To decide whether a cell is reachable by a tunnel, we need to check if there exists a shortcut edge that stabs through an ordered set of disks centered at a subset of the vertices of the other curve. To formalize this, we use the notion of ordered stabbers and line-stabbing wedges as defined by Guibas et al. [23].

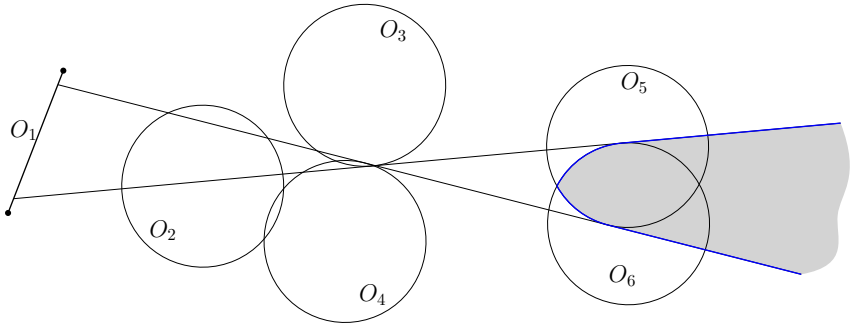


Figure 2 Example for the line-stabbing wedge for a line segment O_1 and disks $O_2, \ldots, O_6$. The line-stabbing wedge is shown in gray, with its boundary in blue.

▶ **Definition 15** (Line-stabbing wedge). *Given a sequence of n convex objects $O_1, \ldots, O_n$, an ordered stabber of this sequence is a line segment $l(x) = (1-x)s + xt$ from s to t, such that points $0 \leq x_1 \leq x_2 \leq \ldots \leq x_n \leq 1$ exist with $p_i = l(x_i) \in O_i$. We call p_i the realising points of l. We say that l stabs through $O_1, \ldots, O_n$. We call the set of points t that are endpoints of ordered stabbers of $O_1, \ldots, O_n$ the line-stabbing wedge of this sequence.*

In their paper, Guibas et al. give an algorithm to compute the line-stabbing wedge for a sequence of n unit disks as well as convex polygons of constant size, with running time $\mathcal{O}(n \log n)$. This line-stabbing wedge is described by $\mathcal{O}(n)$ circular arcs, and two tangents that go to infinity (see Figure 2). These instances can be scaled, such that we can use any sequence of disks $\{\mathrm{b}_\delta(p_1), \ldots, \mathrm{b}_\delta(p_n)\}$, where all disks have the same radius ($\mathrm{b}_\delta(p)$ denotes the closed disk of radius δ centered around p).

▶ **Observation 16.** *Let T, B and δ be given. Denote by v_k the vertices of T. For any feasible tunnel $\tau(p, q)$ with $p = (x_p, y_p) \in C_{a,b}$ and $q = (x_q, y_q) \in C_{i,j}$, it holds that $\overline{B}[y_p, y_q]$ stabs through the ordered set $\{\mathrm{b}_\delta(v_{a+1}), \ldots, \mathrm{b}_\delta(v_i)\}$, if and only if the price of $\tau(p, q)$ is at most δ.*

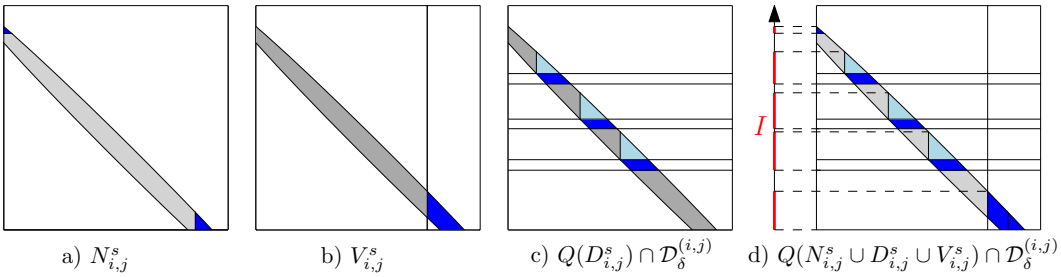
3 Exact decider algorithm

In this section we describe an exact decider algorithm for the k-shortcut Fréchet distance for two polygonal curves. The algorithm can also be used to solve the decision problem of the (unparameterized) shortcut Fréchet distance by setting $k = n$. We first describe the algorithm and then analyse its correctness and running time in Section 3.2. The full correctness proof can be found in [19].

3.1 The Algorithm

We are given a parameter k, a value δ and the two polygonal curves T and B in the plane. Our algorithm iterates over the δ-free space diagram of T and B in k rounds. In each round, based on the computation of the previous round, we compute the set of points that are reachable by using one additional shortcut. The goal is to compute the (δ, s)-reachable space $\mathcal{R}_{\delta,s}(T, B)$ in round s. In each round, we handle the cells of the free space diagram in a row-by-row order, and within each row from left to right. For every cell $C_{i,j}$ we consider three possible ways that a monotone path with proper tunnels can enter.

1. The monotone path could enter the cell $C_{i,j}$ from the neighboring cell $C_{i-1,j}$ to the left or from the neighboring cell $C_{i,j-1}$ below. This does not directly involve a tunnel.



a) $N^s_{i,j}$ b) $V^s_{i,j}$ c) $Q(D^s_{i,j}) \cap \mathcal{D}^{(i,j)}_\delta$ d) $Q(N^s_{i,j} \cup D^s_{i,j} \cup V^s_{i,j}) \cap \mathcal{D}^{(i,j)}_\delta$

Figure 3 Example of the composition of the reachable space within a single free-space cell. The lightblue sets in *c*) (resp. *d*)) contain all the points reachable via monotone paths inside the cell reachable from $D^s_{i,j}$ (resp. $N^s_{i,j} \cup D^s_{i,j} \cup V^s_{i,j}$) computed via $Q(\cdot) \cap \mathcal{D}^{(i,j)}_\delta$. The fragmentation of the reachable space in this cell $P^s_{i,j} = Q(N^s_{i,j} \cup D^s_{i,j} \cup V^s_{i,j}) \cap \mathcal{D}^{(i,j)}_\delta$ results in a large family of intervals I on the edge of B.

2. The monotone path could reach $C_{i,j}$ with a proper tunnel. We distinguish between vertical and diagonal tunnels (compare [16, 21] for a similar distinction).
   - **(i)** The tunnel may start in any cell $C_{a,b}$ with $a < i$ and $b < j$. We call this a diagonal tunnel.
   - **(ii)** The tunnel may start in any cell $C_{i,b}$ for $b < j$. We call this a vertical tunnel.

Using this distinction, we will describe how to compute the set of points reachable by a monotone path with s proper tunnels, for each cell of the diagram. We denote the set computed by the algorithm for cell $C_{i,j}$ in round s with $P^s_{i,j}$. The (δ, s)-reachable space is then obtained by taking the union of these sets over all rounds $\mathcal{R}^{(i,j)}_{\delta,s} = \bigcup_{0 \le s' \le s} P^{s'}_{i,j}$.

After k rounds, the algorithm tests whether the point $(1, 1)$ is contained in our computed set of reachable points. If this is the case, then the algorithm returns "$d^k_{\mathcal{S}}(T, B) \le \delta$", otherwise the algorithm returns "$d^k_{\mathcal{S}}(T, B) > \delta$".

Propagating reachability within a cell. To simplify the description of the algorithm, we use the following set function which receives a set $P \subseteq C_{i,j}$ for some cell $C_{i,j}$ and which extends P to all points above and to the right of it.

$$Q(P) = \{(x, y) \in [0, 1]^2 \mid \exists (a, b) \in P \text{ such that } a \le x \text{ and } b \le y\}$$

We will usually intersect this set with $\mathcal{D}^{(i,j)}_\delta$ to obtain all points that are reachable from a point of P by a monotone path that stays inside the δ-free space of this cell. Figure 3 c) shows an example of the resulting set. Note that the boundary of the resulting set can be described by pieces of the boundary of $\mathcal{D}^{(i,j)}_\delta$, pieces of the boundary of P, and horizontal and vertical line segments.

Step 1: Neighbouring cells. Since we traverse the cells of the diagram in a lexicographical order, we have already computed the (possibly empty) sets $P^s_{i-1,j}$ and $P^s_{i,j-1}$, by the time we handle cell $C_{i,j}$ in round s. Therefore, we can compute the incoming reachability intervals by intersecting $P^s_{i-1,j}$ and $P^s_{i,j-1}$ with $C_{i,j}$. Now we apply the function Q to these sets and denote the result with $N^s_{i,j}$ (refer to Figure 3 a):

$$N^s_{i,j} = \big(Q(P^s_{i-1,j} \cap C_{i,j}) \cup Q(P^s_{i,j-1} \cap C_{i,j})\big) \cap \mathcal{D}^{(i,j)}_\delta$$

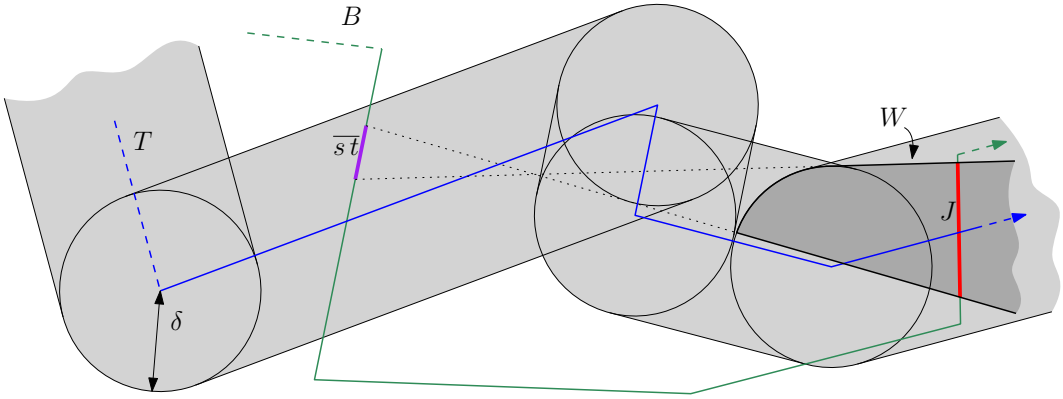


Figure 4 Example of the set J (in red) computed by the DIAGONALTUNNEL procedure.

Step 2 (i): Diagonal tunnels. We invoke the following procedure for every $a < i$ and $b < j$ with $P^{s-1}_{a,b}$. We denote the union of resulting sets of points in $\mathcal{D}^{(i,j)}_\delta$ computed in this step with $D^s_{i,j}$.

The procedure is given a set of points $P^{s-1}_{a,b}$ in the δ-free space $\mathcal{D}^{(a,b)}_\delta$ and computes all points in $\mathcal{D}^{(i,j)}_\delta$ that are endpoints of tunnels starting in $P^{s-1}_{a,b}$ with price at most δ. The procedure first projects $P^{s-1}_{a,b}$ onto the edge e_b of the base curve. The resulting set consists of disjoint line segments $I = \{\overline{s_1\,t_1}, \ldots\}$ along e_b (refer to Figure 3 d)). The procedure then computes the line-stabbing wedge W through $\overline{s_1\,t_1}$ and disks $\mathrm{b}_\delta(v_{a+1}), \ldots, \mathrm{b}_\delta(v_i)$ centered at vertices of T. W is then intersected with the edge e_j, resulting in a set J on e_j corresponding to a horizontal slab in $C_{i,j}$ (compare Figure 3 c) and Figure 4). This resulting set is then intersected with $\mathcal{D}^{(i,j)}_\delta$ to obtain all endpoints of feasible shortcuts with price at most δ starting in $\overline{s_1\,t_1}$. The procedure performs the above steps for every line segment $\overline{s\,t} \in I$ and returns the union of these sets. The resulting set may look as illustrated in Figure 3 c).

Step 2 (ii): Vertical tunnels. Let p denote a point in $\bigcup_{l \le j-1} P^{s-1}_{i,l}$ with minimal x-coordinate, i.e., a leftmost point in this set. A feasible vertical tunnel always has price at most δ, as the Fréchet distance of two line segments is bounded by the maximal euclidean distance of the start points or end points. Therefore, we simply take all points in the δ-free space to the right of p in the cell $C_{i,j}$. To do this, we compute the intersection of a halfplane that lies to the right of the vertical line at p with the δ-free space in $C_{i,j}$. We denote this set with $V^s_{i,j}$. Refer to Figure 3 b) for an example.

Putting things together. Finally, we compute the set $P^s_{i,j}$ by taking the union of the computed sets and extending this set by using the function Q defined above:

$$P^s_{i,j} = Q(N^s_{i,j} \cup D^s_{i,j} \cup V^s_{i,j}) \cap \mathcal{D}^{(i,j)}_\delta$$

It remains to specify the initialization: We define $P^0_{1,0} = P^0_{0,1} = \{(0,0)\}$, if $(0,0) \in \mathcal{D}_\delta$, and otherwise $P^0_{1,0} = P^0_{0,1} = \emptyset$. In addition we define $P^{-1}_{i,j} = \emptyset$ for all i, j.

3.2 Analysis

We argue that the structure of $P^s_{i,j}$ as computed by the algorithm is indeed as claimed. Namely for all i, j and s it holds that $\mathcal{R}^{(i,j)}_{\delta,s} = \bigcup_{0 \le s' \le s} P^{s'}_{i,j}$. The main argument of the proof consists of considering any monotone path with s proper tunnels ending in some cell.

Correctness then follows via induction over the lexicographical ordering of the cells in each round, and by induction over all rounds. The full proof of correctness can be found in [19]. Here, we provide an analysis of the running time of the algorithm.

▶ **Lemma 17.** *Let T and B be two polygonal curves in the plane with overall complexity n, together with a distance threshold $\delta > 0$. The algorithm described in Section 3.1 has running time in $\mathcal{O}(kn^{2k+2}\log n)$ and uses $\mathcal{O}(kn^{2k+2})$ space.*

Proof. Note that the sets $N^s_{i,j}$, $D^s_{i,j}$ and $V^s_{i,j}$ computed by the algorithm are described as intersections of $\mathcal{D}^{(i,j)}_\delta$ with halfplanes and horizontal slabs, and unions of these. For a fixed $P^s_{i,j}$ we define $n_{i,j,s}$ as the total number of such operations from which $P^s_{i,j}$ was obtained. As such, $\mathcal{O}(n_{i,j,s})$ bounds the complexity of this set.

The complexity of $N^s_{i,j}$ and $V^s_{i,j}$ is constant. The complexity of $D^s_{i,j}$ is bounded by the sum of the complexities of all cells to the lower left:

$$n_{i,j,s} \in \mathcal{O}\left(\sum_{0\le a<i}\sum_{0\le b<j} n_{a,b,s-1}\right).$$

As $i, j \le n$, and $s \le k$, and $n_{a,b,0} \in \mathcal{O}(1)$ for all a and b, it holds that $n_{i,j,s} \in \mathcal{O}(n^{2k})$.

Computing $D^s_{i,j}$ takes $\mathcal{O}(\sum_{a<i}\sum_{b<j} n_{a,b,s-1}\log n + n^2\log n) = \mathcal{O}(n^{2k}\log n)$ time. This follows from the fact, that we compute $\mathcal{O}(n)$ line-stabbing wedges, and for every cell $C_{a,b}$ with $a < i$ and $b < j$ we handle $n_{a,b,s-1}$ line segments based on $P^{s-1}_{a,b}$. Computing $N^s_{i,j}$ takes $\mathcal{O}(n_{i-1,j,s} + n_{i,j-1,s}) = \mathcal{O}(n^{2k})$ time, as we need to compute the reachability intervals from neighbouring cells. Computing $V^s_{i,j}$ takes $\mathcal{O}(\sum_{b<j} n_{i,b,s-1}) = \mathcal{O}(n^{2k-1})$ time, as we need to compute the leftmost point $l^{s-1}_{i,j-1}$. The space required to store $P^s_{i,j}$ as required by latter iterations and cells is in $\mathcal{O}(n^{2k})$. Computing $Q(N^s_{i,j} \cup V^s_{i,j} \cup D^s_{i,j})$ takes linear time in the complexity of $N^s_{i,j} \cup V^s_{i,j} \cup D^s_{i,j}$, i.e. $\mathcal{O}(n^{2k})$. As we do this for every cell in every round, the running time overall is $\mathcal{O}(kn^{2k+2}\log n)$, and the space is bounded by $\mathcal{O}(kn^{2k+2})$. ◀

Lemma 17 together with the claimed correctness then imply Theorem 5. The algorithm can also be used for the (unparameterized) shortcut Fréchet distance by choosing $k = n$, since there can be at most n proper tunnels. We obtain the following corollary.

▶ **Corollary 18.** *Let T and B be polygonal curves in the plane with overall complexity n and let $\delta > 0$. There exists an algorithm with running time in $\mathcal{O}\big(n^{2n+3}\log n\big)$ and space in $\mathcal{O}\big(n^{2n+3}\big)$ that decides whether the shortcut Fréchet distance of T and B is at most δ.*

4 Approximate decision algorithm

In this section we describe a $(3+\varepsilon)$-approximation algorithm for the decision problem of the k-shortcut Fréchet distance of two polygonal curves in the plane. The algorithm has running time near-linear in n for c-packed curves. For general curves the running time is still polynomial in n and linear in k.

4.1 The modified algorithm

We describe how to modify the algorithm of Section 3 to circumvent the exponential complexity of the reachable space and obtain a polynomial-time approximation algorithm.

Let two polygonal curves T and B be given, together with a distance threshhold δ and approximation parameter ε. As before, the algorithm iterates over the cells of the free-space diagram and computes sets $N^s_{i,j}$, $V^s_{i,j}$, and $D^s_{i,j}$ for each cell $C_{i,j}$. The main difference is

now that, instead of computing the exact set of points that can be reached by a diagonal tunnel, we want to use an approximation for this set. For this, we define an approximate diagonal tunnel procedure, see further below. This procedure is called with the rightmost point $r^{s-1}_{i-1,j-1}$ in $\bigcup_{a<i;b<j} P^{s-1}_{a,b}$, ε and $\delta' = 3\delta$. Crucially, the set resulting from one call to the procedure has constant complexity and is sufficient to approximate the set $D^s_{i,j}$. We then compute $P^s_{i,j} = Q(N^s_{i,j} \cup D^s_{i,j} \cup V^s_{i,j}) \cap \mathcal{D}^{(i,j)}_\delta$, similarly to Section 3. From this we compute (i) the leftmost point $l^s_{i,j}$ in $\bigcup_{b\leq j} P^s_{i,b}$ based on $P^s_{i,j}$ and $l^s_{i,j-1}$, (ii) the rightmost point $r^s_{i,j}$ in $\bigcup_{a\leq i;b\leq j} P^s_{a,b}$ based on $P^s_{i,j}$, $r^s_{i-1,j}$ and $r^s_{i,j-1}$, and (iii) the outgoing reachability intervals of $P^s_{i,j}$. We store these variables to be used in the next round. Finally, after k rounds, we check if $(1,1)$ is contained in the computed set of reachable points.

Our approximate diagonal tunnel procedure makes use of a data structure by Driemel and Har-Peled, which is summarized in the following lemma. This data structure needs to be built once on T in the beginning and is then available throughout the algorithm.

▶ **Lemma 19** (distance oracle [21]). *Given a polygonal curve Z with n vertices in $\mathbb{R}^d$ and $\varepsilon > 0$, one can build a data structure $\mathcal{F}_\varepsilon$ in $\mathcal{O}\big(\chi^2 n \log^2 n\big)$ time, that uses $\mathcal{O}\big(n\chi^2\big)$ space such that given a query segment $\overline{p\,q}$ and any two points u and v on the curve, one can $(1+\varepsilon)$-approximate $d_{\mathcal{F}}(\overline{p\,q}, Z[u,v])$ in $\mathcal{O}\big(\varepsilon^{-2}\log n \log\log n\big)$ time, where $\chi = \varepsilon^{-d}\log\big(\varepsilon^{-1}\big)$.*

▶ **Definition 20** (Grid). *We define the scaled integer grid $\mathbb{G}_\delta = \big\{(\delta x, \delta y) \mid (x,y) \in \mathbb{Z}^2\big\}$.*

Approximate diagonal tunnel procedure. The procedure is described by Algorithm 1. The procedure is provided with parameters ε, δ, some $r' = (r_T, r_B)$ in cell $C_{a,b}$ and the edge e_j that is associated with a cell $C_{i,j}$. We want to compute a set of stabbers starting at $r = B(r_B)$ that contains every stabber through the disks $\mathrm{b}_\delta(v_{a+1}), \ldots, \mathrm{b}_\delta(v_i)$, and is contained in the set of all stabbers through disks of radius $(1+\varepsilon)^2\delta$ centered at the same vertices. We approximate this set of stabbers as follows.

We iterate over all grid points t of $\mathbb{G}_{\frac{\delta\varepsilon}{\sqrt{2}}}$ inside the disk $\mathrm{b}_{(1+\varepsilon)\delta}(v_i)$, and make queries to $\mathcal{F}_\varepsilon$ to determine if the Fréchet distance of the query segment $\overline{r\,t}$ to the subcurve of T from $T(r_T)$ to v_i is sufficiently small. We mark t if the approximate distance returned by the data structure is at most $(1+\varepsilon)^2\delta$. We then compute the convex hull H of all marked grid points, and the two tangents t_1 and t_2 of H through $B(r_B)$. The true set of endpoints of stabbers is approximated by the set H' of points that lie inside and 'behind' the convex hull H, from the perspective of r. Figure 5 illustrates this. We then intersect H' with the edge e_j resulting in a single horizontal slab in $C_{i,j}$, which is then intersected with $\mathcal{D}^{(i,j)}_\delta$ and returned.

4.2 Correctness of the approximate diagonal tunnel procedure

We denote with $\{\mathrm{b}_\delta(v_i)\}_i$ a sequence of disks $\{\mathrm{b}_\delta(v_1), \ldots, \mathrm{b}_\delta(v_m)\}$ for some m.

▶ **Lemma 21.** *Let $a, b_1, b_2 \in \mathbb{R}^d$ together with a sequence of vertices $v_1, \ldots, v_n$ be given. If $\overline{a\,b_1}$ stabs through disks $\{\mathrm{b}_\delta(v_i)\}_i$, and $\overline{a\,b_2}$ stabs through $\{\mathrm{b}_\delta(v_i)\}_i$, then for any $t \in [0,1]$ the line segment $\overline{a\,b(t)}$ stabs through $\{\mathrm{b}_\delta(v_i)\}_i$, where $b(t) = (1-t)b_1 + tb_2$.*

Proof. Refer to Figure 6. We can interpret the setting as a triangle with sides $(b_1 - a)$, $(b_2 - a)$, $(b_1 - b_2)$, where the first two sides correspond to the original stabbers and the last side to $b(t)$. Note that any line segment $\overline{a\,b(t)}$ lies completely within this triangle with $(b_1 - a)$ on the one and $(b_2 - a)$ on the other side. Hence, for every i and realising points p_i of $\overline{a\,b_1}$ and q_i of $\overline{a\,b_2}$, p_i lies on the one and q_i on the other side of $\overline{a\,b(t)}$. Since $\mathrm{b}_\delta(v_i)$ is convex and p_i and q_i are inside this disk, the intersection of $\overline{p_i\,q_i}$ and $\overline{a\,b(t)}$ is inside the disk

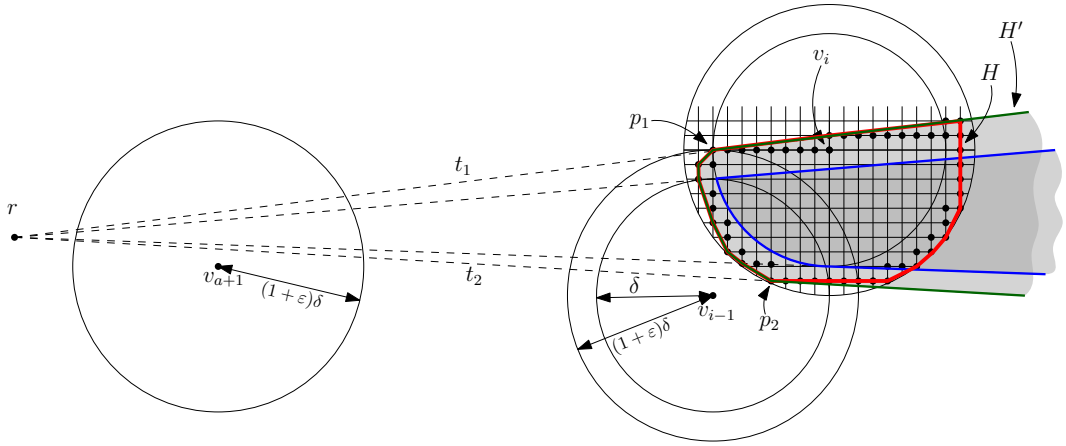


Figure 5 Illustration to the approximate diagonal tunnel procedure. The true line-stabbing wedge for disks with radius δ is shown in blue. The convex hull of eligible grid points is shown in red. The approximate line stabbing wedge is shown in green.

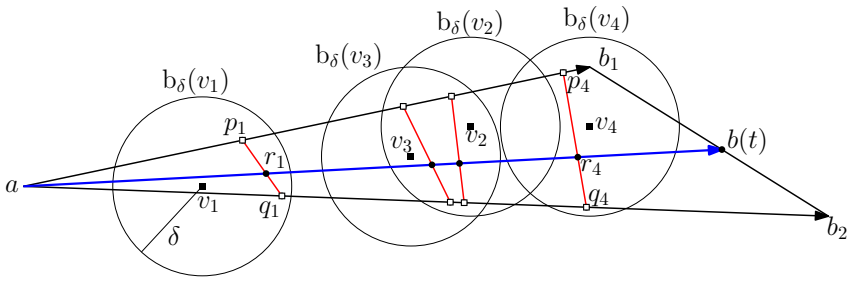


Figure 6 Linear interpolation between two δ-stabbers starting in a. Illustrations to the proof of Lemma 21. In blue $\overline{a\,b(t)}$, and in red $\overline{p_i\,q_i}$ is illustrated. Their intersections form the realising points r_i of $\overline{a\,b(t)}$.

as well. Call this intersection point r_i. The set $\{r_i\}_i$ are realising points for $\overline{a\,b(t)}$. This follows directly from the fact that $\{p_i\}_i$ and $\{q_i\}_i$ are ordered along their respective line segments, and thus $\overline{p_i\,q_i}$ never crosses another $\overline{p_j\,q_j}$. Thus for $i < j$, r_i appears before r_j along $\overline{a\,b(t)}$, implying the claim. ◀

▶ **Lemma 22.** *Let $a_1, a_2, b_1, b_2 \in \mathbb{R}^2$ together with a sequence of vertices $v_1, \ldots, v_n$ be given. If $\overline{a_1\,b_1}$ stabs through $\{\mathrm{b}_\delta(v_i)\}_i$, and $\|a_1 - b_1\| \leq \delta'$ and $\|a_2 - b_2\| \leq \delta'$, then $\overline{a_1\,b_1}$ stabs through $\{\mathrm{b}_{\delta+\delta'}(v_i)\}_i$.*

Proof. By Observation 11, $d_{\mathcal{F}}(\overline{a_1\,b_1}, \overline{a_2\,b_2}) \leq \delta'$, via the reparametrizaion (f, g) with $f(t) = (1-t)a_1 + ta_2$ and similarly $g(t) = (1-t)b_1 + tb_2$. As $p = \overline{a_1\,b_1}$ stabs through $\{\mathrm{b}_\delta(v_i)\}_i$, there exist realising points p_i along p, with p_i lying in the δ-disk centered at v_i. Then

$$\|g(f^{-1}(v_i)) - v_i\| \leq \|g(f^{-1}(v_i)) - p_i\| + \|p_i - v_i\| \leq \delta' + \delta.$$

Additionally, $q_i = g(f^{-1}(v_i))$ are ordered along $q = \overline{a_2\,b_2}$, proving the claim. ◀

▶ **Lemma 23.** *Given $r' = (r_T, r_B) \in C_{a,b}$, and edge e of B corresponding to the cell $C_{i,j}$, ε and δ like in the approximate diagonal tunnel procedure. Denote by S_δ the set of endpoints of all δ-stabbers (that is, stabbers through $\mathrm{b}_\delta(v_m)$ for $a+1 \leq m \leq i$) on the edge e_j starting at $r = B(r_B)$ and let C' be the point set computed by the algorithm. Then*

$$S_\delta \subseteq C' \subseteq S_{(1+\varepsilon)^2\delta}.$$

Algorithm 1 Approximate Diagonal Tunnel.

procedure apxDiagonalTunnel((r_T, r_B), (i, j), ε, δ)
  Let $r = B(r_B)$
  //r is the starting point of the shortcut
  for $t \in \left(\mathbb{G}_{\frac{\delta\varepsilon}{\sqrt{2}}} \cap \mathrm{b}_{3(1+\varepsilon)\delta}(v_i)\right)$ **do**
    Query $\mathcal{F}_\varepsilon$ for the distance $d_{\mathcal{F}_\varepsilon}(\overline{r\,t}, T[r_T, v_i])$ and store the answer in δ'
    if $\delta' \leq (1+\varepsilon)^2\delta$ **then**
      Mark t as eligible
      //t is an eligible endpoint of a shortcut
  Compute the convex hull H of eligible points
  if $r \in H$ **then**
    Return $C = \mathcal{D}_\delta^{(i,j)}$
  else
    Let U be the cone with apex r formed by tangents t_1 and t_2 from r to H
    Let $p_i \in H$ be a supporting point of the tangent t_i for $i \in \{1, 2\}$
    Let L be the subchain of ∂H with endpoints p_1 and p_2 which is facing r
    Let $H' \subset U$ be the set bounded by L and the rays supported by t_1 and t_2 facing away from r
    Let C' be the intersection of H' with e_j
    Return $C = (e_i \times C') \cap \mathcal{D}_\delta^{(i,j)}$

Proof. Let $y \in C'$. Then $q = B(y) \in H'$ where H' is set of points computed by the algorithm. Denote the intersection of $\overline{r\,q}$ and the boundary of H' by h. h is then a linear combination of at most two grid points whose stabbers from r have been marked as eligible i.e. who are $(1+\varepsilon)^2\delta$-stabber. Hence, Lemma 21 implies that $\overline{r\,q}$ is also a $(1+\varepsilon)^2\delta$-stabber, implying $C' \subseteq S_{(1+\varepsilon)^2\delta}$. Now let $q \in e$ be an arbitrary point such that $\overline{r\,q}$ is a δ-stabber. Let t be the last realising point of $\overline{r\,q}$. The line segment $\overline{r\,t}$ is a δ-stabber and t lies in $\mathrm{b}_\delta(v_i)$. We claim that t lies in H. Consider the set $G = \mathbb{G}_{\frac{\delta\varepsilon}{\sqrt{2}}} \cap \mathrm{b}_{\varepsilon\delta}(t)$. By the scale of the grid, t lies within the convex hull of G. Moreover $G \subset \mathrm{b}_{(1+\varepsilon)\delta}(v_i)$. Lemma 22 implies that $\overline{r\,t'}$ is a $((1+\varepsilon)\delta)$-stabber for any $t' \in G$. This in turn implies that for the first point s' of $\overline{r\,t'}$ inside $\mathrm{b}_{\delta(1+\varepsilon)}(v_a)$, $\overline{s'\,t'}$ is a $((1+\varepsilon)\delta)$-stabber, hence, t' would have been marked as an eligible endpoint of a $((1+\varepsilon)^2\delta)$-stabber. Since H is the convex hull of eligible points, it follows that $t \in \mathrm{conv}(G) \subset H$. Therefore $q \in H'$ and thus $q \in C'$. ◀

4.3 Result

We argue that the structure of $P_{i,j}^s$ as approximated by the algorithm is indeed as claimed. Namely for all i, j and s it holds that $\mathcal{R}_{\delta,s}^{(i,j)} \subset \bigcup_{0 \leq s' \leq s} P_{i,j}^{s'} \subset \mathcal{R}_{3(1+\varepsilon)^2\delta,s}^{(i,j)}$. The full proof can be found in [19]. We again consider any monotone path with s proper tunnels ending in some cell and show the set inclusion by induction. Indeed, it suffices to consider the tunnel starting in the rightmost reachable point in the lower left quadrant of the cell, if we call the approximate diagonal tunnel procedure with a distance threshold $\delta' = 3\delta$. This is implied by a lemma by Driemel and Har-Peled concerning the structure of prices of tunnels. The lemma states that if a feasible tunnel $\tau(r, q)$ costs more than 3δ then any feasible tunnel $\tau(p, q)$ with $x_p \leq x_r$ costs more than δ.

▶ **Lemma 24** (monotonicity of tunnels [21]). *Given a value $\delta > 0$ and two curves T_1 and T_2 such that T_2 is a subcurve of T_1, and given two line segments $\bar{B}_1$ and $\bar{B}_2$ such that $d_{\mathcal{F}}(T_1, \bar{B}_1) \leq \delta$ and the start (resp. end) point of T_2 is within distance δ to the start (resp. end) point of $\bar{B}_2$, then $d_{\mathcal{F}}(T_2, \bar{B}_2) \leq 3\delta$.*

The proof of the following theorem is very similar to the proof of Theorem 5. We invoke the algorithm of Section 4.1 with approximation parameter $\varepsilon' = \varepsilon/9$ and distance threshold δ, as $3(1+\varepsilon')^2 \leq (3+\varepsilon)\delta$. The main difference is the approximation of the set of points reachable by a diagonal tunnel.

▶ **Theorem 25.** *Let T and B be two polygonal curves in the plane with overall complexity n, together with values $0 < \varepsilon \leq 1$ and $\delta > 0$. There exists an algorithm with running time in $\mathcal{O}\big(kn^2\varepsilon^{-5}\log^2\big(n\varepsilon^{-1}\big)\big)$ and space in $\mathcal{O}\big(kn^2\varepsilon^{-4}\log^2\big(\varepsilon^{-1}\big)\big)$ which outputs one of the following: (i) $d_{\mathcal{S}}^k(T,B) \leq (3+\varepsilon)\delta$ or (ii) $d_{\mathcal{S}}^k(T,B) > \delta$. In any case, the output is correct.*

The running time of this algorithm can be improved even more for the special class of c-packed curves. For this, we use a known approximation scheme (see [22, 21, 7]) that uses μ-simplifications, which are a kind of approximation of the input curves. Crucially, the free-space diagram of two μ-simplifications of c-packed curves consists of only $\mathcal{O}(cn\varepsilon^{-1})$ many non-empty cells, if μ is chosen in the right way. These non-empty cells can be found using standard techniques in an output-sensitive manner.

Concretely, our algorithm first computes μ-simplifications of T and B, with $\mu = \varepsilon''\delta''$ where $\varepsilon'' = \varepsilon/20$ and $\delta'' = (1-2\varepsilon'')$, and then invokes the algorithm of Section 4.1 on the simplifications with distance threshold δ'' and approximation parameter ε''. From this, we obtain the following theorem. The detailed analysis can be found in [19].

▶ **Theorem 8.** *Let T and B be two c-packed polygonal curves in the plane with overall complexity n, together with values $0 < \varepsilon \leq 1$ and $\delta > 0$. There exists an algorithm with running time in $\mathcal{O}\big(kcn\varepsilon^{-5}\log^2\big(n\varepsilon^{-1}\big)\big)$ and space in $\mathcal{O}\big(kcn\varepsilon^{-4}\log^2\big(\varepsilon^{-1}\big)\big)$ which outputs one of the following: (i) $d_{\mathcal{S}}^k(T,B) \leq (3+\varepsilon)\delta$ or (ii) $d_{\mathcal{S}}^k(T,B) > \delta$. In any case, the output is correct.*

5 Hardness

We prove that the decision problem cannot be solved in $n^{o(k)}$ time, unless ETH fails. The full description can be found in [19]. We describe how to modify the NP-hardness reduction by Buchin, Driemel and Speckmann from [16] to suit our needs.

5.1 General idea

Our reduction from ETH works via the following intermediate problem. This link is facilitated by Pătraşcu and Williams [26].

▶ **Definition 26** (k-Table-SUM). *We are given k lists $S_1, \ldots, S_k$ of n non-negative integers $\{s_{i,1}, \ldots, s_{i,n}\}$ and a non-negative integer σ. We want to decide whether there are indices $\iota_1, \ldots, \iota_k$ such that $\sum_{i=1}^k s_{i,\iota_i} = \sigma$. We call $\sigma_j = \sum_{i=1}^j s_{i,\iota_i}$ the jth partial sum.*

Based on a k-Table-SUM instance we describe how to construct a $(4k+2)$-shortcut Fréchet distance instance consisting of the target curve T and the base curve B with the property, that they have a distance of 1 if and only if the underlying instance has a solution.

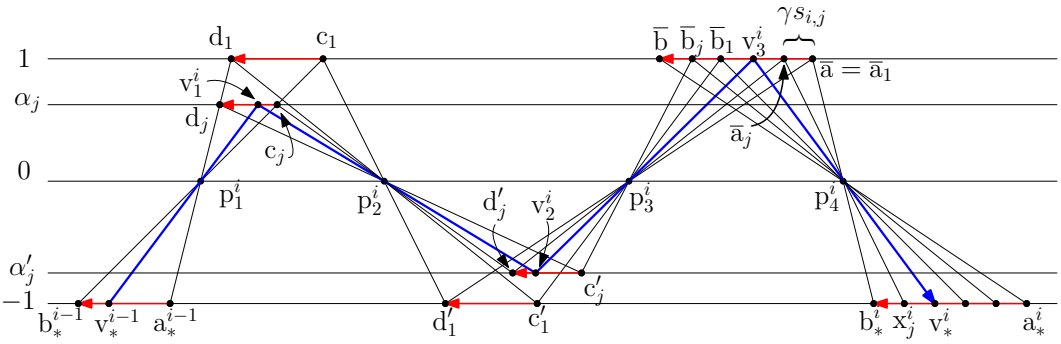


Figure 7 Schematic view of the path of a shortcut curve (in blue) through the gadget g_i in the case, where $s_{i,j}$ is selected from the ith list. Most top indices i are omitted. Furthermore, for presentation the mirror edges have horizontal overlap, while in the construction they do not.

The target curve T will lie on a horizontal line mostly going to the right. The only exceptions being so called *twists*. Twists force shortcuts traversing it to go through precisely one point, called its *focal point*. These twists are constructed by going a distance of 2 to the left, before continuing rightwards. Refer to Figure 8, points $p_1, \cdots, p_4$ and Figure 7, points $p_1^i, \cdots, p_4^i$.

The set of points in $\mathbb{R}^2$ which have a distance of at most 1 to the target curve we will call the *hippodrome*. The base curve will consist of several horizontal edges going to the left close to the boundary of the hippodrome. All other edges of the base curve will (essentially) lie outside the hippodrome. Exceptions have to be considered carefully. Any shortcut curve of B that has Fréchet distance of at most 1 to T we will call *feasible*. It is easy to see that any feasible shortcut curve must lie completely in the hippodrome. Since any edge of the base curve inside the hippodrome lies close to the boundary of it and is oriented in the opposite direction of the base curve, no feasible shortcut curve contains a large subcurve of the base curve. We will not place any edges of the base curve too close to twists, so that a shortcut must be taken to traverse these.

Intuitively we can think of the horizontal edges of the base curve as mirrors that disperse incoming light in all directions and focal points as a wall with a hole, like in a pinhole camera. A shortcut curve can be thought of as the path of a photon that tries to traverse this instance. It bounces from mirror to mirror, always passing through a focal point. A feasible shortcut curve exists if and only if it is possible to send a photon from the beginning of the base curve to the very end.

The instance as a whole can be segmented into gadgets, with a special gadget at the start and end, used to initialize and verify the solution (c.f. [16]). In between these, k gadgets will be placed, encoding one of the lists each from the k-Table-SUM instance.

5.2 Construction of the gadgets

In this section, we describe three gadgets: the encoding gadget g_i, the initialization gadget g_0 and the terminal gadget g_{k+1}. The base curve and the target curve pieces of these gadgets are then connected in the order of i. The encoding gadget is the heart-piece of our construction and constitutes the main difference to the NP-hardness reduction presented in [16].

Encoding gadget. The overall structure of a gadget g_i for some $1 \leq i \leq k$ is depicted in Figure 8. This gadget will encode the ith table $S_i = \{s_{i,1}, \ldots, s_{i,n}\}$ of the k-Table-SUM instance. As for the parameters, λ^i is the length of the entry edge, determined by the

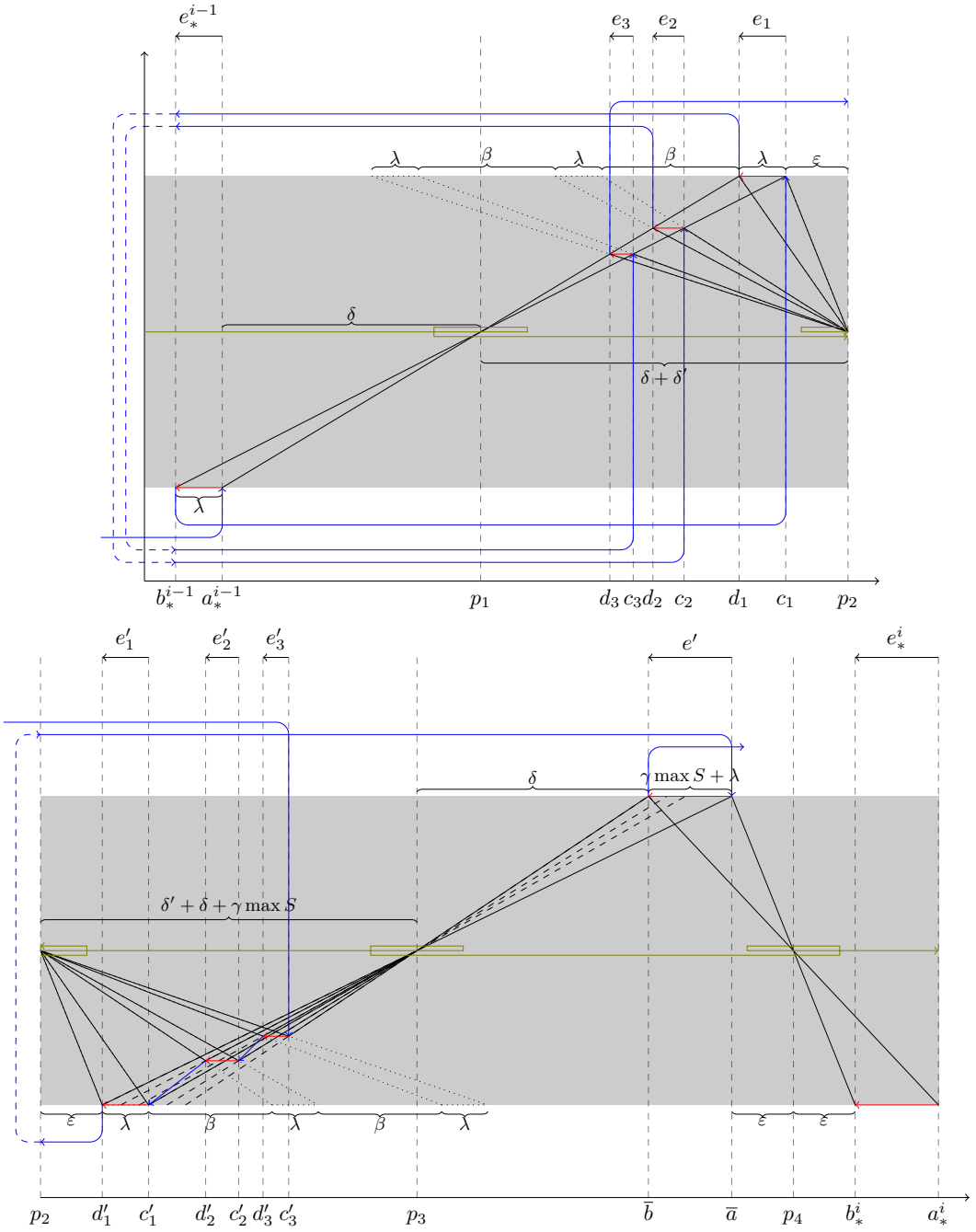


Figure 8 Detailed view of the construction of the encoding gadget. Mirror edges are red, connector edges blue and the target curve is green. Projection cones are black.

previous gadget g_{i-1}, and β is a global spacing parameter. The parameters δ^i and δ'^i are auxiliary parameters, with $\delta'^i = \lambda^i + \varepsilon$ and $\delta^i = \max(2\varepsilon + 1, (n-1)(\lambda^i + \beta) - \delta'^i)$ where ε is a globally fixed spacing parameter for twists. Excluding the entry edge of the base curve, B consists of $2n+2$ mirror edges and $\mathcal{O}(n)$ connector edges. For $1 \leq j \leq n$ the first n mirror edges e^i_j are defined by c^i_j and d^i_j, and the second n mirror edges e'^i_j are defined by c'^i_j and d'^i_j. The last two mirror edges are defined by $\bar{\mathrm{a}}^i$ and $\bar{\mathrm{b}}^i$, and a^i_* and b^i_*. The target curve T has four twists centred at $\mathrm{p}^i_1, \ldots, \mathrm{p}^i_4$. Since the index i will not change other than for the entry and exit edge, we will omit these indices in the construction of this gadget.

Intuition. The intuition behind the construction is as follows: We place the first projection point p_1 at a distance from the entry edge, such that n 'well-spaced' scaled copies of the entry edge fit 'behind' the projection point. These edges offer the choice, each edge corresponding to a single element in the table S_i. After this choice has been presented, we place another n copies of the entry edge behind the second projection point p_2. The two sets of copies correspond one-to-one, and any line starting at one copy going through p_2 only ever hits a single copy of the second set. The second set of copies is placed in such a way, such that passing through the third projection point p_3, any feasible shortcut curve receives an additional offset of $\gamma s_{i,j}$ on the edge e' corresponding to the choice of edge e_j. Lastly we define the entry edge to the next gadget.

How a solution is encoded. A shortcut curve traversing this gadget will look as follows. A shortcut curve reaches some point on the entry edge e^{i-1}_*. From here it takes a shortcut to some e^i_j, where the number j corresponds to a choice of edge to end on. The next shortcuts are forced to land on e'^i_j, then e'^i and finally e^i_*. The offset between the endpoint v^i_* of the last shortcut and b^i_* will be precisely the offset between v^{i-1}_* and b^i_* plus $\gamma s_{i,j}$ (refer to Figure 7), thus the choice of which edge to land on encodes picking an item. After k choices the offset will be approximately encoding the corresponding partial sum in the offset from b^k_*.

Initialization gadget. For the construction refer to Figure 9. Both the target curve T and the base curve B will start at x-coordinate 0 placing the start point for the base curve at $(0, 1)$, and the start point for the target curve at $(0, 0)$. The target curve will go rightwards, up to the first twist centred at $(\varepsilon + \gamma, 0)$ and continue rightwards after that. The base curve will immediately leave the hippodrome to the left and connect to the first mirror edge from $\mathrm{a}^0_* = (3\gamma + 2\varepsilon, -1)$ to $\mathrm{b}^0_* = (\gamma + 2\varepsilon, -1)$.

Terminal gadget. The terminal gadget g_{k+1} is the dual to the initialization gadget (refer to Figure 9). The entry edge from $(b^k_* + \lambda^k, -1)$ to $(b^k_*, -1)$ is defined by the previous gadget. The target curve T has a single twist at $(b^k_* + \lambda + \varepsilon, 0)$ and ends at $(b^k_* + 2\lambda + 2\varepsilon - \gamma(\sigma+1), 0)$. The base curve B connects the entry edge to $(b^k_* + 2\lambda + 2\varepsilon - \gamma(\sigma+1), 1)$ from outside the hippodrome. The final vertex $B(1)$ of the base curve is placed such that a shortcut from the entry edge e^k_* has to start precisely at x-coordinate $b^k_* + \gamma(\sigma+1)$ to hit the vertex.

5.3 Result

In the full version [19] of this paper we show that the curves can be constructed in $\mathcal{O}(kn)$ time. Furthermore, if we choose $\varepsilon = \frac{5}{2}$, $\beta = \max(32 + 4\lambda^k, \lambda^{k+1}) + 1$ and $\gamma = 16k + 6$, then the maximum numerical value of the coordinates constructed is in $\mathcal{O}(k^2 n \sum_{i=0}^{k} \max S_i)$. We then analyse the structure of a feasible solution under this choice of parameters and show that it properly encodes the partial sums, such that we can retain the solution to the k-Table-SUM instance. From this analysis, we obtain the following theorem.

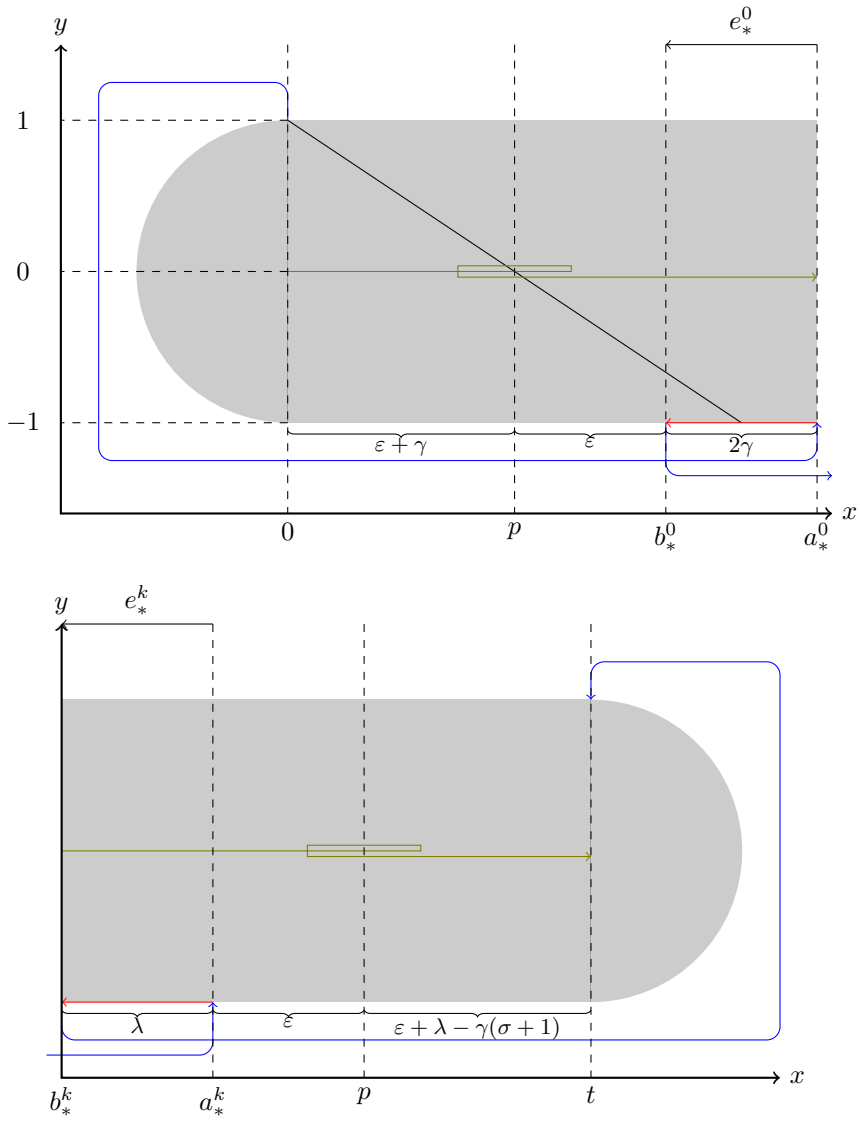


Figure 9 Construction of the Initialization and Terminal gadget. The first forced shortcut in the Initialization gadget is drawn in black. Mirror edges are red, connector edges are blue, and the target curve is green.

▶ **Theorem 6.** *Unless ETH fails, there is no algorithm for the k-shortcut Fréchet distance decision problem in $\mathbb{R}^d$ for $d \geq 2$, with running time $n^{o(k)}$.*

References

1 Amir Abboud and Kevin Lewi. Exact Weight Subgraphs and the k-Sum Conjecture. In *Proceedings of the 40th International Conference on Automata, Languages, and Programming – Volume Part I*, pages 1–12. Springer-Verlag, 2013. doi:10.1007/978-3-642-39206-1_1.

2 Pankaj K. Agarwal, Rinat Ben Avraham, Haim Kaplan, and Micha Sharir. Computing the Discrete Fréchet Distance in Subquadratic Time. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 156–167. Society for Industrial and Applied Mathematics, 2013. doi:10.1137/1.9781611973105.12.

3 Hugo Alves Akitaya, Maike Buchin, Leonie Ryvkin, and Jérôme Urhausen. The k-Fréchet Distance: How to Walk Your Dog While Teleporting. In *30th International Symposium on Algorithms and Computation*, volume 149, pages 50:1–50:15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019. doi:10.4230/LIPIcs.ISAAC.2019.50.

4 Helmut Alt, Bernd Behrends, and Johannes Blömer. Approximate matching of polygonal shapes. *Annals of Mathematics and Artificial Intelligence*, 13:251–265, 1995. doi:10.1007/BF01530830.

5 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.

6 Helmut Alt, Christian Knauer, and Carola Wenk. Comparison of Distance Measures for Planar Curves. *Algorithmica*, 38:45–58, 2003. doi:10.1007/s00453-003-1042-5.

7 Boris Aronov, Sariel Har-Peled, Christian Knauer, Yusu Wang, and Carola Wenk. Fréchet Distance for Curves, Revisited. In *Proceedings of the 14th Conference on Annual European Symposium - Volume 14*, pages 52–63. Springer-Verlag, 2006. doi:10.1007/11841036_8.

8 Rinat Ben Avraham, Omrit Filtser, Haim Kaplan, Matthew J. Katz, and Micha Sharir. The Discrete and Semicontinuous Fréchet Distance with Shortcuts via Approximate Distance Counting and Selection. *ACM Transactions on Algorithms*, 11(4):29:1–29:29, 2015. doi:10.1145/2700222.

9 Karl Bringmann. Why Walking the Dog Takes Time: Frechet Distance Has No Strongly Subquadratic Algorithms Unless SETH Fails. In *55th IEEE Annual Symposium on Foundations of Computer Science*, pages 661–670, 2014. doi:10.1109/FOCS.2014.76.

10 Karl Bringmann and Marvin Künnemann. Improved Approximation for Fréchet Distance on c-Packed Curves Matching Conditional Lower Bounds. *International Journal of Computational Geometry & Applications*, 27(1-2):85–120, 2017. doi:10.1142/S0218195917600056.

11 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.

12 Alexander M Bronstein, Michael M Bronstein, Alfred M Bruckstein, and Ron Kimmel. Partial Similarity of Objects, or How to Compare a Centaur to a Horse. *International Journal of Computer Vision*, 84(2):163, 2009. doi:10.1007/s11263-008-0147-3.

13 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.

14 Kevin Buchin, Maike Buchin, and Yusu Wang. Exact algorithms for partial curve matching via the Fréchet distance. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 645–654. Society for Industrial and Applied Mathematics, 2009. doi:10.1137/1.9781611973068.71.

15 Kevin Buchin, Tim Ophelders, and Bettina Speckmann. SETH Says: Weak Fréchet Distance is Faster, but only if it is Continuous and in One Dimension. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2887–2901, 2019. doi:10.1137/1.9781611975482.179.

16 Maike Buchin, Anne Driemel, and Bettina Speckmann. Computing the Fréchet distance with shortcuts is NP-hard. In *Proceedings of the Thirtieth Annual Symposium on Computational Geometry*, pages 367–376. Association for Computing Machinery, 2014. doi:10.1145/2582112.2582144.

17 Timothy M. Chan and Zahed Rahmati. An improved approximation algorithm for the discrete Fréchet distance. *Information Processing Letters*, 138:72–74, 2018. doi:10.1016/j.ipl.2018.06.011.

18 Connor Colombe and Kyle Fox. Approximating the (Continuous) Fréchet Distance. In *37th International Symposium on Computational Geometry*, volume 189, pages 26:1–26:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.SoCG.2021.26.

19 Jacobus Conradi and Anne Driemel. On Computing the k-Shortcut Fréchet Distance, 2022. doi:10.48550/ARXIV.2202.11534.

20 Jean-Lou De Carufel, Amin Gheibi, Anil Maheshwari, Jörg-Rüdiger Sack, and Christian Scheffer. Similarity of polygonal curves in the presence of outliers. *Computational Geometry*, 47(5):625–641, 2014. doi:10.1016/j.comgeo.2014.01.002.

21 Anne Driemel and Sariel Har-Peled. Jaywalking Your Dog: Computing the Fréchet Distance with Shortcuts. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 318–337. Society for Industrial and Applied Mathematics, 2012. doi:10.1137/1.9781611973099.30.

22 Anne Driemel, Sariel Har-Peled, and Carola Wenk. Approximating the Fréchet Distance for Realistic Curves in near Linear Time. In *Proceedings of the Twenty-Sixth Annual Symposium on Computational Geometry*, pages 365–374. Association for Computing Machinery, 2010. doi:10.1145/1810959.1811019.

23 Leonidas Guibas, John Hershberger, Joseph Mitchell, and Jack Snoeyink. Approximating Polygons and Subdivisions with Minimum-Link Paths. In *International Journal of Computational Geometry & Applications*, volume 3, pages 151–162, 1994. doi:10.1007/3-540-54945-5_59.

24 Russell Impagliazzo and Ramamohan Paturi. The complexity of k-SAT. In *Proceedings of the Fourteenth Annual IEEE Conference on Computational Complexity*, pages 237–240. IEEE Computer Society, 1999. doi:10.1109/CCC.1999.766282.

25 David Jacobs, Daphna Weinshall, and Yoram Gdalyahu. Classification with Nonmetric Distances: Image Retrieval and Class Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):583–600, 2000. doi:10.1109/34.862197.

26 Mihai Pătraşcu and Ryan Williams. On the Possibility of Faster SAT Algorithms. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1065–1075. Society for Industrial and Applied Mathematics, 2010. doi:10.1137/1.9781611973075.86.

27 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.

28 R.C. Veltkamp. Shape matching: similarity measures and algorithms. In *Proceedings International Conference on Shape Modeling and Applications*, pages 188–197, 2001. doi:10.1109/SMA.2001.923389.