Computing the Fréchet distance with shortcuts is NP-hard

Computing the Fréchet distance with shortcuts is NP-hard

Maike Buchin
Faculty of Mathematics
Ruhr-University Bochum
Maike.Buchin@rub.de

Anne Driemel*
Dep. Mathematics and Computer Science
TU Eindhoven
a.driemel@tue.nl

Bettina Speckmann†
Dep. Mathematics and Computer Science
TU Eindhoven
b.speckmann@tue.nl

ABSTRACT

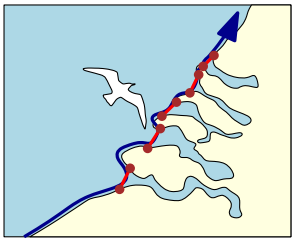
We study the *shortcut Fréchet distance*, a natural variant of the Fréchet distance, that allows us to take shortcuts from and to any point along one of the curves. The classic Fréchet distance is a bottle-neck distance measure and hence quite sensitive to outliers. The shortcut Fréchet distance allows us to cut across outliers and hence produces more meaningful results when dealing with real world data. Driemel and Har-Peled recently described approximation algorithms for the restricted case where shortcuts have to start and end at input vertices. We show that, in the general case, the problem of computing the shortcut Fréchet distance is NP-hard. This is the first hardness result for a variant of the Fréchet distance between two polygonal curves in the plane. We also present two algorithms for the decision problem: a 3-approximation algorithm for the general case and an exact algorithm for the vertex-restricted case. Both algorithms run in $O(n^3 \log n)$ time.

1. INTRODUCTION

Measuring the similarity of two curves is an important problem which occurs in many applications. A popular distance measure, that takes into account the continuity of the curves, is the Fréchet distance. Imagine walking forwards along both of the two curves whose similarity is to be measured. At any point in time, the positions on the two curves have to stay within distance ε. The minimal ε for which such a traversal is possible is the Fréchet distance. It has been used for simplification [1, 5], clustering [8], and map-matching [2, 16]. The Fréchet distance also has applications in matching biological sequences [19], analysing tracking data [6, 7], and matching coastline data [18].

*Anne Driemel was supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 612.065.823.
†Bettina Speckmann was supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 639.023.208.

SoCG'14, June 8–11, 2014, Kyoto, Japan.
 ACM 978-1-4503-2594-3/14/06 ...$15.00.



Despite its versatility, the Fréchet distance has one serious drawback: it is a bottleneck distance. Hence it is quite sensitive to outliers, which are frequent in real world data sets. To remedy this Driemel and Har-Peled [15] introduced a variant of the Fréchet distance, namely the *shortcut Fréchet distance*, that allows shortcuts from and to any point along one of the curves. The shortcut Fréchet distance is then defined as the minimal Fréchet distance over all possible such shortcut curves.

The shortcut Fréchet distance automatically cuts across outliers and allows us to ignore data specific "detours" in one of the curves. Hence it produces more meaningful results when dealing with real world data than the classic Fréchet distance. Consider the following two examples. Birds are known to use coastlines for navigation, e.g., the Atlantic flyway for migration. However, when the coastline takes a "detour", like a harbor or the mouth of a river, the bird ignores this detour, and instead follows a "shortcut" across. See the example of a seagull in the figure, navigating along the coastline of Zeeland. Using the shortcut Fréchet distance, we can detect this similarity. Now imagine a hiker following a pilgrims route. The hiker will occasionally detour from the route, for breaks along the way. In the former example, shortcuts are allowed on the coastline, in the latter on the hiker's path.

Related work. The standard Fréchet distance can be computed in time roughly quadratic in the complexity of the input curves [3, 9]. Driemel and Har-Peled introduce the notion of the shortcut Fréchet distance and describe approximation algorithms in the restricted case where shortcuts have to start and end at input vertices [15]. In particular, they give a $(3+\varepsilon)$-approximation algorithm for the vertex-restricted shortcut Fréchet distance that runs in $O(n \log^3 n)$ time under certain input assumptions. Specifically, they assume c-packedness, that is, the length of the input curves in any ball is at most c times the radius of the ball, where c is a constant. Their algorithm also yields a polynomial-time exact algorithm to compute the vertex-restricted shortcut Fréchet distance that runs in $O(n^5 \log n)$ time and uses $O(n^4)$ space without using any input assumptions [14].

A discrete version of the shortcut Fréchet distance has been studied by Avraham *et al.* [4]. Their definition is based on the discrete Fréchet distance, which is a distance measure for sequences of points, rather than polygonal curves. Among other

things, they show that a linear-time decision algorithm is possible. Considering that the best-known algorithms for deciding the discrete Fréchet distance are essentially quadratic, the fact that introducing shortcuts allows for considerably faster algorithms in the discrete case stands in sharp contrast to our results.

The shortcut Fréchet distance can be interpreted as a partial distance measure, that is, it maps parts of one curve to another curve. In contrast to other partial distance measures, it is parameter-free. A different notion of a partial Fréchet distance was studied by Buchin et al. [10] and more recently by De Carufel *et al.* [13]. They propose the total length of the longest subcurves which lie within a given Fréchet distance to each other as similarity measure. The parts omitted are completely ignored, while in our definition these parts are substituted by shortcuts, which have to be matched under the Fréchet distance.

Results and Organization. We study the complexity of computing the shortcut Fréchet distance. Specifically, we show that in the general case, where shortcuts can be taken at any point along a curve, the problem of computing the shortcut Fréchet distance exactly is NP-hard. This is the first hardness result for a variant of the Fréchet distance between two polygonal curves.

Below we give an exact definition of the problem we study. In Section 2 we describe a reduction from SUBSET-SUM to the decision version of the shortcut Fréchet distance and in Section 3 we prove its correctness. The NP-hardness stems from an exponential number of combinatorially different shortcut curves which translate into an exponential number of components in the free space diagram. We use this in our reduction together with a mechanism that controls the sequence of free space components that may be visited.

In Section 4 we give two algorithms for the decision version of the problem, which both run in $O(n^3 \log n)$ time. These algorithms traverse the free space (as usual for Fréchet distance algorithms), and make use of a line stabbing algorithm of Guibas et al. [17] to test whether shortcuts are admissible. The first algorithm uses a lemma of Driemel and Har-Peled [15] to approximate the reachable free space and so prevents it from being fragmented. This yields a 3-approximation algorithm for the decision version of the general shortcut Fréchet distance. The second algorithm concerns the vertex-restricted case. Here, the free space naturally does not fragment, however, the line-stabbing algorithm helps us to improve upon the running time of the exact decision algorithm described by Driemel [14].

Definitions. A *curve* T is a continuous mapping from $[0,1]$ to $\mathbb{R}^2$, where $T(t)$ denotes the point on the curve parameterized by $t \in [0,1]$. Given two curves T and B in $\mathbb{R}^2$, the *Fréchet distance* between them is

$$\mathsf{d}_{\mathcal{F}}(T,B) = \inf_{f:[0,1]\to[0,1]} \max_{\alpha\in[0,1]} \|T(f(\alpha)) - B(\alpha)\|,$$

where f is an orientation-preserving homeomorphism. We call the line segment between two arbitrary points $B(y)$ and $B(y')$ on B a *shortcut* on B. Replacing a number of subcurves of B by the shortcuts connecting their endpoints results in a shortcut curve of B. Thus, a *shortcut curve* is an order-preserving concatenation of non-overlapping subcurves of B that has straight line segments connecting the endpoints of the subcurves.

Our input are two polygonal curves: the *target curve* T and *base curve* B. The *shortcut Fréchet distance* $\mathsf{d}_{\mathsf{S}}(T,B)$ is now defined as the minimal Fréchet distance between the target curve T and any shortcut curve of the base curve B.

2. NP-HARDNESS REDUCTION

We prove that deciding if the shortcut Fréchet distance between two given curves is smaller or equal a given value is weakly NP-hard by a reduction from SUBSET-SUM. We first discuss the main ideas and challenges in Section 2.1, then we formally describe the reduction in Section 2.2. The correctness is proven in Section 3.

2.1 General idea

The SUBSET-SUM instance is given as a set of input values and a designated sum value. A solution to the instance is a subset of these values that has the specified sum. We describe how to construct a *target curve* T and a *base curve* B, such that there exists a shortcut curve of B which is in Fréchet distance 1 to T if and only if there exists a solution to the SUBSET-SUM instance. We call such a shortcut curve *feasible* if it lies within Fréchet distance 1. The construction of the curves is split into gadgets, each of them encoding one of the input values.

The idea of the reduction is as follows. We construct the target curve to lie on a horizontal line going mostly rightwards. The base curve has several horizontal edges which lie at distance exactly 1 to the target curve and which go leftwards. We call these edges "mirrors" for reasons that will become clear later. All other edges of the base curve lie at distance greater than 1 to the target curve. If a shortcut curve is feasible, then any of its shortcuts must start where the previous shortcut ended. This way, the shortcut curve "jumps" rightwards along the mirrors of the base curve and visits each edge in exactly one point. We restrict the solution space of feasible shortcut curves further by letting the target curve go leftwards for a distance 2 and then rightwards again (see Figure 1). We call this a "twist". We place the mirrors far enough from any twist, such that any shortcut curve which is feasible has to go through the center of each such twist, since it has to traverse the twist region using a shortcut and this shortcut has to have Fréchet distance at most 1 to the twisting portion.

The reader may picture the shortcut curve as a lightbeam, which is reflected in all directions when it hits a mirror. In this analogy, the target curve is a wall which has a hole at the center of each twist, thus, these are the only points where light can go through. Only if we can send light from the first vertex of the base curve to the last vertex of the base curve, there exists a feasible shortcut curve. This curve describes the path of one photon from the first to the last vertex of the base curve.

Using the basic mechanism of twists and mirrors, we can transport information rightwards along the shortcut curve as follows. Assume we have a shortcut curve that encodes the empty set. It describes the path of a photon emitted from the first vertex of the base curve. Assume that another photon, which took a different path, is reflected at distance x along the same mirror. If both photons also travel through the next twist center, they will hit the next mirror at the same distance x from each other.

We can offer a choice to the photon by placing two mirror edges e^i and $\overline{\mathsf{e}}^i$ in its line of direction (see Figure 1). The choice of the edge changes the position at which the photon will hit

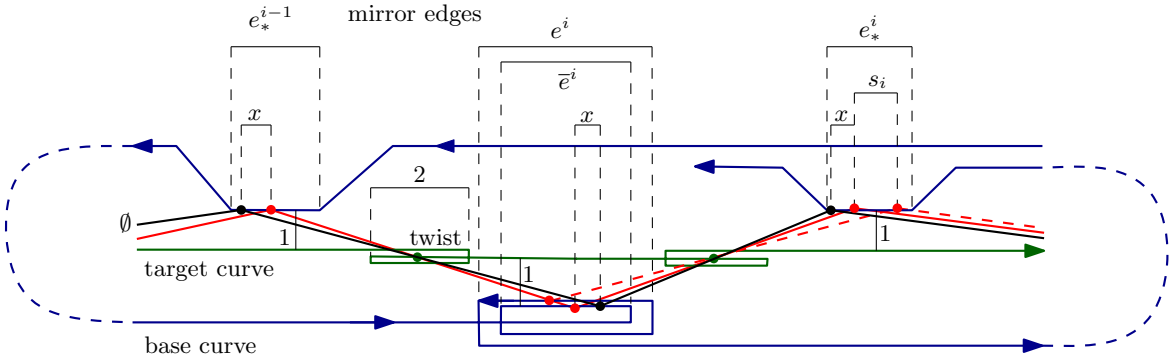


Figure 1: Simplistic version of the gadget encoding one of the input values s_i. A shortcut curve entering from a position on the edge e_*^{i-1} has the choice to visit either e^i or $\overline{\mathsf{e}}^i$. Its distance on e_*^i to the shortcut curve encoding the empty set (x in the figure) is altered by s_i. The target curve is distorted to show its topology.

the next mirror edge. In particular, by placing the mirror edges carefully, we can encode one of the input values s_i in this horizontal shift. By visiting a number of such gadgets, which have been threaded together, a shortcut curve accumulates the sum of the selected subset in its distance to the curve that encodes the empty set. We construct the terminal gadget such that only those photons that selected a subset which sums to the designated value can see the last vertex of the base curve through the last hole in the wall.

There are two aspects of this construction we need to be careful with. First, if we want to offer a choice of mirrors to the same photon, we cannot place both of them at distance exactly 1 to the target curve. We will place one of them at distance $\alpha = 1/2$. In this case, it may happen that a shortcut curve visits the edge in more than one point by moving leftward on the edge before leaving again in rightward direction. Therefore, the visiting position which encodes the current partial sum will be only approximated. We will scale the problem instance to prevent influence of this approximation error on the solution. Secondly, if two mirror edges overlap horizontally, such as e^i and $\overline{\mathsf{e}}^i$ in Figure 1, a photon could visit both of them. We will use more than two twists per gadget to realize the correct spacing (see Figure 2).

2.2 Reduction

We describe how to construct the curves T and B from an instance of SUBSET-SUM and how to extract a solution.

Input. We are given n positive integers $\{s_1, s_2, \ldots, s_n\}$ and a positive integer σ. The problem is to decide whether there exists an index set I, such that $\sum_{i\in I} s_i = \sigma$. For any index set I, we call $\sigma_i = \sum_{1\le j\le i, j\in I} s_j$ the ith *partial sum* of the corresponding subset.

Global layout. We describe global properties of the construction and introduce basic terminology. Our construction consists of $n+2$ gadgets: an initialization gadget g_0, a terminal gadget g_{n+1}, and split gadgets g_i for each value s_i, for $i = 1, \ldots, n$. A *gadget* g_i consists of curves T_i and B_i. We concatenate these in the order of i to form T and B. The construction of the gadgets $g_1, \ldots, g_n$ is incremental. Given the endpoints of the last mirror edge of gadget g_{i-1} and the value s_i, we construct gadget g_i. We denote with H_y the horizontal line at y. The target curve will be contained in H_0. We call the locus of points that are within distance 1 to the target curve the *hippodrome*.

The base curve B_i will have leftward horizontal edges on H_{-1}, H_1 and H_α, where $\alpha = \frac{1}{2}$ is a global parameter of the construction. We call these edges *mirror edges*. The remaining edges of B_i, which are used to connect the mirror edges to each other, are called *connector edges*. The mirror edges which will be located on H_1 and H_{-1} can be connected using curves that lie outside the hippodrome. Since those connector edges cannot be visited by any feasible shortcut curve, their exact placement is irrelevant. The edges connecting to mirror edges located on H_α and which intersect the hippodrome are placed carefully such that no feasible shortcut curve can visit them.

The edges of the target curve T_i lie on H_0 running in positive x-direction, except for occasional twists, which we define as follows. A *twist* centered at a point $\mathsf{p} = (p, 0)$ is a subcurve defined by the vertices $(p-1, 0)$,$(p+1, 0)$,$(p-1, 0)$,$(p+1, 0)$ which we connect by straight line segments in this order. We call p the *projection center* of the twist. Let $\zeta = 5$ be a global parameter of the construction, we call the open rectangle of width ζ and height 3 and centered at p a *buffer zone* of the twist. In our construction, the base curve stays outside the buffer zones.

Since all relevant points of the construction lie on a small set of horizontal lines, we can slightly abuse notation by denoting the x-coordinate of a point and the point itself with the same variable, albeit using a different font.

Global variables. The construction uses four global variables. The parameter $\alpha = \frac{1}{2}$ is the y-coordinate of a horizontal mirror edge which does not lie on H_1 or H_{-1} . The parameter $\beta = 5$ controls the minimal horizontal distance between mirror edges that lie between two consecutive buffer zones. The parameter $\gamma > 0$ acts as a scaling parameter to ensure (i) that the projections stay inside the designated mirror edges and (ii) that the projections of two different partial sums are kept disjoint despite the approximation error. How to choose the exact value of γ will follow from Lemma 4. The fourth parameter $\zeta = 5$ defines the width of a buffer zone. It controls the minimum horizontal distance that a point on a mirror edge has to a projection center.

Initialization gadget (g_0)**.** We let both curves start on the vertical line at $a_0^0 = 0$ by placing the first vertex of T_0 at $(a_0^0, 0)$ and the first vertex of the B_0 at $(a_0^0, 1)$. The base curve B_0 then continues to the left on H_1 while the target curve T_0 continues to the right on H_0. See Figure 2 (top

Step 1:	$\mathsf{p}_1 = \overline{\mathsf{a}_*^{i-1}(\phi_i, -\alpha)} \cap H_0$	with $\phi_i = a_*^{i-1} + \beta + \lambda_i$
		$\lambda_i = a_*^{i-1} - b_*^{i-1}$
	$\mathsf{a}_1 = \overline{\mathsf{b}_*^{i-1}\mathsf{p}_1} \cap H_1$	$\mathsf{b}_1 = \overline{\mathsf{a}_*^{i-1}\mathsf{p}_1} \cap H_1$
	$\mathsf{c}_1 = \overline{\mathsf{b}_*^{i-1}\mathsf{p}_1} \cap H_\alpha$	$\mathsf{d}_1 = \overline{\mathsf{a}_*^{i-1}\mathsf{p}_1} \cap H_\alpha$
Step 2:	$\mathsf{p}_2 = (a_1 + \zeta/2, 0)$	
	$\mathsf{a}_2 = \overline{\mathsf{b}_1\mathsf{p}_2} \cap H_{-1}$	$\mathsf{b}_2 = \overline{\mathsf{a}_1\mathsf{p}_2} \cap H_{-1}$
	$\mathsf{c}_2 = \overline{\mathsf{d}_1\mathsf{p}_2} \cap H_{-1}$	$\mathsf{d}_2 = \overline{\mathsf{c}_1\mathsf{p}_2} \cap H_{-1}$
Step 3:	$\mathsf{p}_3 = (c_2 + \zeta/2, 0)$	
	$\mathsf{a}_3 = \overline{\mathsf{b}_2\mathsf{p}_3} \cap H_\alpha$	$\mathsf{b}_3 = \overline{\mathsf{a}_2\mathsf{p}_3} \cap H_\alpha$
	$\mathsf{c}_3 = \overline{\mathsf{d}_2\mathsf{p}_3} \cap H_1$	$\mathsf{d}_3 = \overline{\mathsf{c}_2\mathsf{p}_3} \cap H_1$
Step 4:	$\mathsf{p}_4 = \overline{(c_3 - \gamma s_i, 1)\mathsf{a}_3} \cap H_0$	
	$\mathsf{a}_4 = \overline{\mathsf{b}_3\mathsf{p}_4} \cap H_{-1}$	$\mathsf{b}_4 = \overline{\mathsf{a}_3\mathsf{p}_4} \cap H_{-1}$
	$\mathsf{c}_4 = \overline{\mathsf{d}_3\mathsf{p}_4} \cap H_{-1}$	$\mathsf{d}_4 = \overline{\mathsf{c}_3\mathsf{p}_4} \cap H_{-1}$
	$\mathsf{a}_* = (\max(a_4, b_4, c_4, d_4), -1)$	$\mathsf{b}_* = (\min(a_4, b_4, c_4, d_4), -1)$

Table 1: Construction of split gadgets g_i for $1 \le i \le n$. We omitted the top index i of the variables. Thus, b_* stands for b_*^i, etc. H_1, H_{-1} and H_α denote the horizontal lines at $1, -1$ and α respectively. We used $\overline{\mathsf{ab}}$ to denote the line through the points a and b.

left) for an illustration. The curve T_0 has one twist centered at $p_1^0 = a_0^0 + \gamma + \zeta/2$. The curve B_0 has one mirror edge $\mathsf{e}_*^0 = \mathsf{a}_*^0\mathsf{b}_*^0$, which we define by setting $b_*^0 = p_1^0 + \zeta/2$ and $a_*^0 = p_1^0 + 2\gamma + \zeta/2$.

Split gadgets $(g_1, \ldots, g_n)$**.** The overall structure is depicted in Figure 2. The curve B_i for $1 \le i \le n$ has seven mirror edges. These are $\mathsf{e}_j^i = \mathsf{a}_j^i\mathsf{b}_j^i$, and $\overline{\mathsf{e}}_j^i = \mathsf{c}_j^i\mathsf{d}_j^i$, for $1 \le j \le 3$, and the edge $\mathsf{e}_*^i = \mathsf{a}_*^i\mathsf{b}_*^i$. We connect the mirror edges using additional edges to define the following order along the base curve:

$$\mathsf{e}_*^{i-1}, \mathsf{e}_1^i, \overline{\mathsf{e}}_1^i, \overline{\mathsf{e}}_2^i, \mathsf{e}_2^i, \mathsf{e}_3^i, \overline{\mathsf{e}}_3^i, \mathsf{e}_*^i.$$

This order also defines the visiting order of edges visited by a shortcut curve. The mirror edges lie on the horizontal lines H_1, H_{-1} and H_α. We use vertical connector edges which run in positive y-direction and additional connector edges which lie completely outside the hippodrome to connect the mirror edges on H_α. The curve T_i for $1 \le i \le n$ consists of four twists centered at the projection centers p_j^i for $1 \le j \le 4$ which are connected in the order of j by rightward edges on H_0.

To choose the exact coordinates of these points, we go through several rounds of fixing the position of the next projection center and then projecting the endpoints of mirror edges to obtain the endpoints of the next set of mirror edges. The construction is defined in four steps in Table 1 and illustrated in Figure 2 (bottom).

The intuition behind this choice of projection centers is the following. In every step we make sure that the base curve stays out of the buffer zones. Furthermore, in Step 1 we choose the projection center far enough to the right such that two mirror edges located between two consecutive buffer zones have horizontal distance at least β to each other. In Step 4 we align the projections of the two edges e_3^i and $\overline{\mathsf{e}}_3^i$. In this alignment, the visiting position that represents "0" on e_3^i (i.e., in its distance to a_3^i) and the visiting position that represents "s_i" on $\overline{\mathsf{e}}_3^i$ (i.e., in its distance to c_3^i) both project to the same point on e_*^i (i.e., the visiting position that represents "s_i" in its distance to b_*^i). In this way, the projections from e_3^i are horizontally shifted by s_i (scaled by γ) with respect to the projections from $\overline{\mathsf{e}}_3^i$.

Terminal gadget (g_{n+1})**.** The curve T_{n+1} has one twist centered at $p_1^{n+1} = a_*^n + \zeta/2$. Let $p_\sigma = (b_*^n + \gamma(\sigma + 1))$ and project the point $(p_\sigma, -1)$ through p_1^{n+1} onto H_1 to obtain a point $(a_\sigma, 1)$. We finish the construction by letting both the target curve T_{n+1} and the base curve B_{n+1} end on a vertical line at a_σ. The curve T_{n+1} ends on H_0 approaching from the left, while the curve B_{n+1} ends on H_1 approaching from the right. Figure 2 (top right) shows an illustration.

Encoding of a subset. Any shortcut curve of the base curve encodes a subset of the SUBSET-SUM instance. We say the value s_i is included in the encoded subset if and only if the shortcut curve visits the edge e_1^i. The ith partial sum of the encoded subset will be represented by the point where the shortcut curve visits the edge e_i^*. In particular, the distance of the visiting point to the endpoint of the edge represents this value, scaled by γ and up to a small additive error.

3. CORRECTNESS OF THE REDUCTION

Now we prove that the construction has the desired behavior. We first focus on a restricted class of shortcut curves which allow for a simplified analysis. The analysis for the general case is very similar. We call a shortcut curve *one-touch* if it visits any edge of the base curve in at most one point. Intuitively, for any feasible shortcut curve of B, there exists a one-touch shortcut curve that "approximates" it. In the following, we define a *one-touch encoding*, which is a one-touch

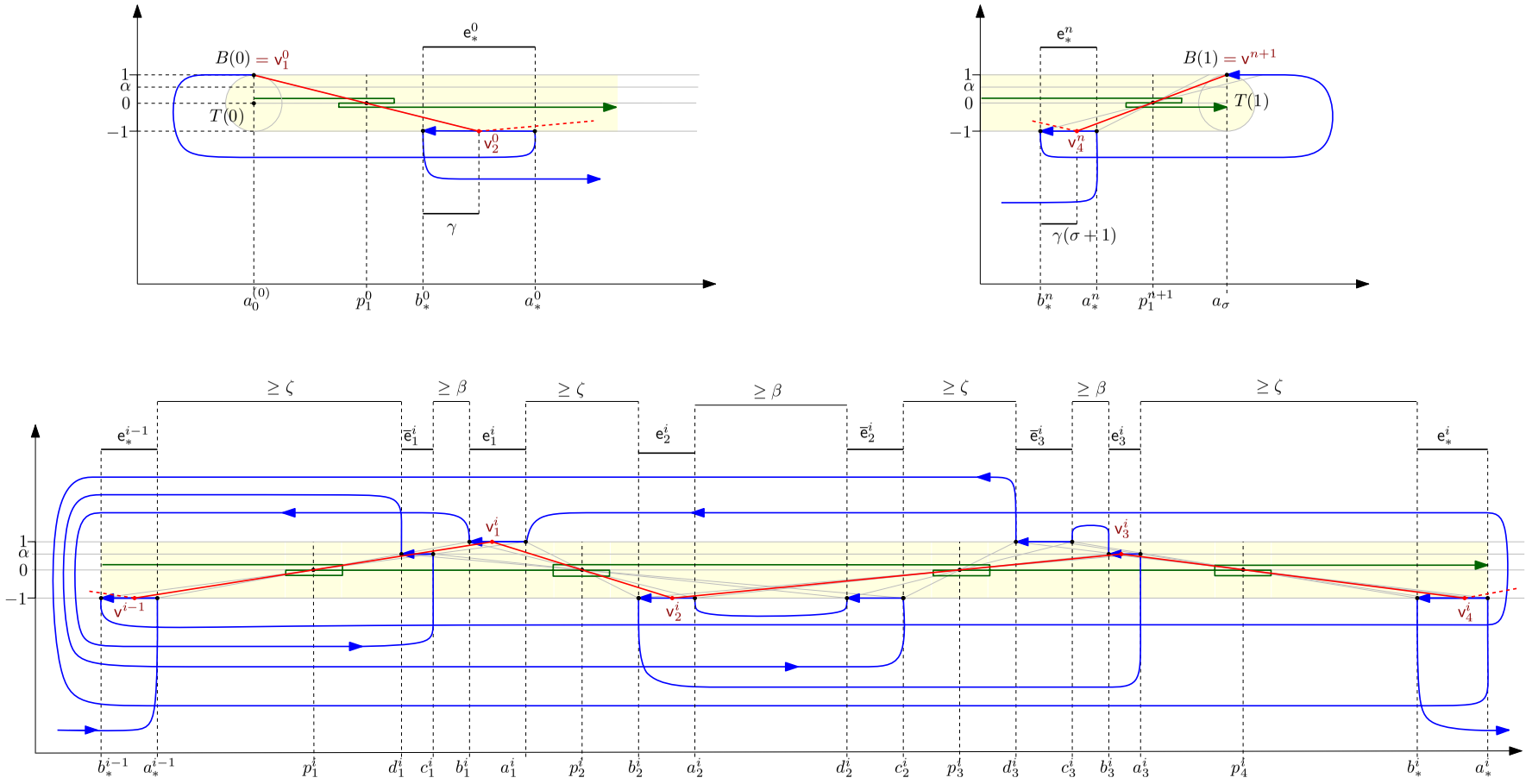
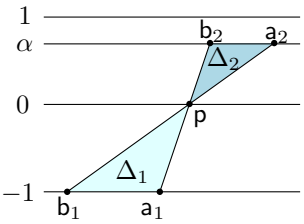


Figure 2: Top left: initialization gadget g_0; top right: terminal gadget g_{n+1}; bottom: gadget g_i. The target curve is shown in green, the base curve in blue. An example shortcut curve is shown in red. Buffer zones and the hippodrome are shown as shaded regions. For the sake of presentation, the target curve is distorted to show its topology, and the lengths of the mirror edges have been assumed smaller.

shortcut curve that is feasible if and only if the encoded subset constitutes a valid solution.

Definition 1 (One-touch encoding). *Let I be an index set of a subset $S' \subseteq S$. We construct a one-touch shortcut curve B_I of the base curve incrementally. The first two vertices on the initial gadget are defined as follows. We choose the first vertex of the base curve $B(0)$ for v_0^0, then we project it through the first projection center p_1^0 onto e_*^0 to obtain v_*^0. Now for $i > 0$, if $i \in I$, then we project v_*^{i-1} through p_1^i onto e_1^i, otherwise onto $\overline{\mathsf{e}}_1^i$ to obtain v_1^i. We continue by projecting v_j^i through p_{j+1}^i onto B_i to obtain v_{j+1}^i, for $1 \le j \le 4$. Let $\mathsf{v}_*^i = \mathsf{v}_4^i$. We continue this construction throughout all gadgets in the order of i. Finally, we choose $B(1) = (a_\sigma, 1)$ as the last vertex of our shortcut curve. Figure 2 shows an example.*

For such curves, Lemma 1 describes the correspondence of the current partial sum with the visiting position on the last edge of each gadget.



In the proof we often reason by using projections of distances between H_1,H_{-1} and H_α. The common argument is captured in the following observation.

Observation 1 (Distance Projection). *If Δ_1 is a triangle defined by two points a_1 and b_1 that lie on H_{-1} and a point p that lies on H_0, and if Δ_2 is a triangle defined by the points p, and the two points $\mathsf{b}_2 = \overline{\mathsf{a}_1\mathsf{p}} \cap H_\alpha$ and $\mathsf{a}_2 = \overline{\mathsf{b}_1\mathsf{p}} \cap H_\alpha$, which are the projections onto H_α, then it holds that $\alpha(a_1 - b_1) = a_2 - b_2$, where a_1, b_1, a_2 and b_2 are the respective x-coordinates of the points.*

Lemma 1. *Given a shortcut curve B_I which is a one-touch encoding (Definition 1), let v_*^i be the vertex of B_I on e_*^i, for any $0 \le i \le n$. It holds for the distance of v_*^i to the endpoint of this edge that $\|\mathsf{v}_*^i - \mathsf{b}_*^i\| = \gamma(\sigma_i + 1)$, where σ_i is the ith partial sum of the subset encoded by B_I.*

Proof. We prove the claim by induction on i. For $i = 0$, the claim is true by the construction of the initialization gadget, since the partial sum $\sigma_0 = 0$ and $\|\mathsf{v}_*^0 - \mathsf{b}_*^0\| = \gamma$. For $i > 0$, there are two possibilities, either $i \in I$ or $i \notin I$. Consider the case that s_i is included in the set encoded by B_I. In that case, the curve has to visit the edge e_1^i.

By a repeated application of Observation 1 we can derive

$$\left\|\mathsf{v}_*^{i-1} - \mathsf{b}_*^{i-1}\right\| = \left\|\mathsf{v}_1^i - \mathsf{a}_1^i\right\| = \cdots = \frac{\left\|\mathsf{v}_3^i - \mathsf{a}_3^i\right\|}{\alpha} = \left\|\mathsf{v}_*^i - \mathsf{b}_4^i\right\|.$$

Refer to Figure 3 for an illustration of the geometry of the path through the gadget. The shaded region shows the triangles that transport the distances. Therefore, by induction and by construction

$$\begin{aligned}\left\|\mathsf{v}_*^i - \mathsf{b}_*^i\right\| &= \left\|\mathsf{v}_*^i - \mathsf{b}_4^i\right\| + \left\|\mathsf{b}_4^i - \mathsf{d}_4^i\right\| \\ &= \gamma(\sigma_{i-1} + 1) + \gamma s_i = \gamma(\sigma_i + 1).\end{aligned}$$

Thus, the claim follows for the case that s_i is selected. For the second case, the curve has to visit the edge $\overline{\mathsf{e}}_1^i$. Again, by Observation 1 it holds that

$$\left\|\mathsf{v}_*^{i-1} - \mathsf{b}_*^{i-1}\right\| = \frac{\left\|\mathsf{v}_1^i - \mathsf{c}_1^i\right\|}{\alpha} = \left\|\mathsf{v}_2^i - \mathsf{d}_2^i\right\| = \cdots = \left\|\mathsf{v}_*^i - \mathsf{d}_4^i\right\|.$$

Thus $\left\|\mathsf{v}_*^i - \mathsf{d}_4^i\right\| = \gamma(\sigma_{i-1} + 1) = \gamma(\sigma_i + 1)$. The construction guarantees that $\mathsf{d}_4^i = \mathsf{b}_*^i$, therefore the claim is implied also in this case. □

The statement of the above lemma is generalized later in Lemma 3, where we bound the approximation error in the encoding of the partial sums. First, we need a lemma that describes the general behaviour of a feasible shortcut curve.

Lemma 2. *If $\alpha \in [1/2, 1)$, $\zeta > 4$, and $\beta \ge \zeta/2$, then a feasible shortcut curve (not necessarily one-touch) passes through every buffer zone of the target curve via its projection center and furthermore it does so from left to right.*

Proof. Any feasible shortcut curve has to start at $B(0)$ and end at $B(1)$, and all of its vertices must lie in the hippodrome or on its boundary. We constructed the base curve such that it does not enter any of the buffer zones and therefore the feasible shortcut curve has to pass through the buffer zone by using a shortcut. If we choose the width of a buffer zone $\zeta > 4$, then the only manner possible to do this while matching to the two associated vertices of the target curve in their respective order, is to go through the intersection of their unit disks that lies at the center of the buffer zone. This is the projection center associated with the buffer zone. By the order in which the mirror edges are connected to form the base curve, it must do so in positive x-direction and it must do so exactly once. □

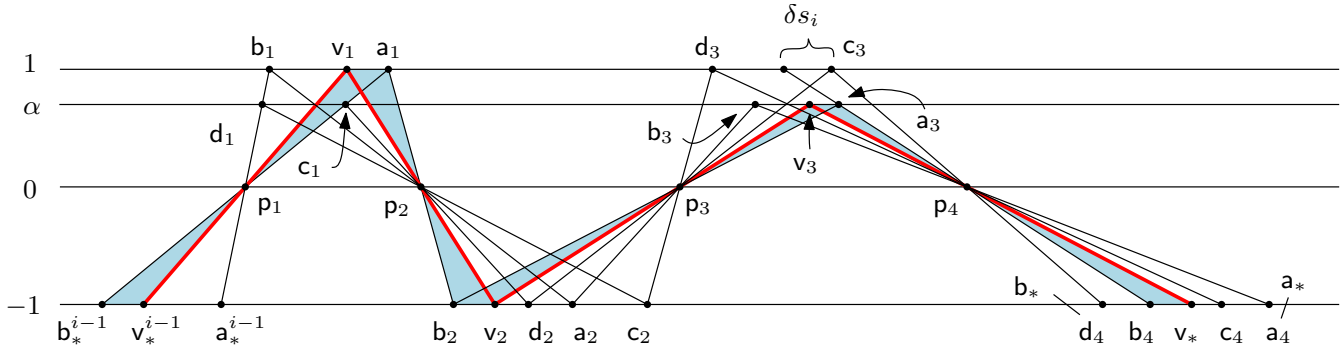


Figure 3: The path of a shortcut curve through the gadget g_i in the case where s_i is included in the selected set (Lemma 1). For presentation purposes, we allowed the mirror edges to overlap horizontally and we omitted the top index i of the variables.

The following lemma shows that *any* feasible shortcut curve (not necessarily one-touch) encodes a partial sum, up to an additive error, in its visiting position of the last edge in each gadget. The proof is in the same line as the argument for Lemma 1. We omit it here and refer to the full version [11].

LEMMA 3. *Choose $\alpha \in [1/2, 1)$, $\zeta > 4$ and $\beta > 4$. Given a feasible shortcut curve $B_\diamond$, let v_*^i be any point of $B_\diamond$ on e_*^i and let v_*^i denote its x-coordinate. For any $0 \le i \le n$ let σ_i denote the ith partial sum of the subset encoded by $B_\diamond$. If we choose $\gamma > \varepsilon_i$, then it holds that*

$$b_*^i + \gamma_i - \varepsilon_i \le v_*^i \le b_*^i + \gamma_i + \varepsilon_i$$

where $\gamma_i = \gamma(\sigma_i + 1)$ and $\varepsilon_i = \frac{8i+4}{\alpha}$.

LEMMA 4. *If we choose $\alpha \in [1/2, 1), \beta > 4$, $\zeta > 4$, and $\gamma \ge 25n$, then any feasible shortcut curve $B_\diamond$ encodes a subset of $\{s_1, \ldots, s_n\}$ that sums to σ.*

PROOF. Since $B_\diamond$ is feasible, it must be that it visits e_*^n at distance $\gamma(\sigma + 1)$ to b_*^n, since this is the only point to connect via a shortcut through the last projection center to the endpoint of B and by Lemma 2 all projection centers have to be visited. So let $v_*^n = b_*^n + \gamma(\sigma + 1)$ be the x-coordinate of this visiting point (the starting point of the last shortcut), and let σ_n be the sum of the subset encoded by $B_\diamond$. Lemma 3 implies that

$$b_*^n + \gamma(\sigma_n + 1) - \varepsilon_n \le b_*^n + \gamma(\sigma + 1) \le b_*^n + \gamma(\sigma_n + 1) + \varepsilon_n,$$

since $\varepsilon_n = \frac{8n+4}{\alpha} < 25n = \gamma$. Therefore,

$$\sigma_n - \varepsilon_n/\gamma \le \sigma \le \sigma_n + \varepsilon_n/\gamma.$$

For our choice of parameters, it holds that $\varepsilon_n/\gamma < 1$. Thus, it must be that $\sigma = \sigma_n$, since both values are integers. □

3.1 Main result

In the full version [11] we prove that the reduction is polynomial. Hence, Lemma 4 together with the fact that one can construct a feasible shortcut curve for any subset that sums to σ (Definition 1) implies the NP-hardness of the problem.

THEOREM 1. *The problem of deciding whether the shortcut Fréchet distance between two given curves is less or equal a given distance is NP-hard.*

4. ALGORITHMS

We give two $O(n^3 \log n)$ time algorithms for deciding the shortcut Fréchet distance. One is a 3-approximation algorithm for the general case, and one an exact algorithm for the vertex-restricted case. Both algorithms traverse the free space as usual, using a line stabbing algorithm by Guibas et al. [17] to test the admissability of shortcuts.

The approximation algorithm for the general case uses a crucial lemma of Driemel and Har-Peled [15] to approximate the reachable free space and prevent it from fragmenting. The exact algorithm for the vertex-restricted case uses a similar lemma for efficiently testing all possible shortcuts. In this case, the free space naturally does not fragment.

First we discuss relevant preliminaries, in particular tunnels in the free space diagram and ordered line-stabbing.

4.1 Preliminaries

Free space Diagram. Let T, B be two polygonal curves parameterized over $[0, 1]$. The standard way to compute the Fréchet distance uses the *δ-free space* of T and B, which is a subset of the joint parametric space of T and B, defined as

$$\mathcal{D}_{\le\delta}(T, B) = \left\{ (x, y) \in [0, 1]^2 \;\middle|\; \|T(x) - B(y)\| \le \delta \right\}.$$

From now on, we will simply write $\mathcal{D}_{\le\delta}$. The square $[0, 1]^2$, which represents the joint parametric space, can be broken into a (not necessarily uniform) grid called the *free space diagram*, where a vertical line corresponds to a vertex of T and a horizontal line corresponds to a vertex of B.

Every pair of segments of T and B define a *cell* in this grid. Let $C_{i,j}$ denote the cell that corresponds to the ith edge of T and the jth edge of B. The cell $C_{i,j}$ is located in the ith column and jth row of this grid. It is known that the free space, for a fixed δ, inside such a cell $C_{i,j}$ (i.e., $\mathcal{D}_{\le\delta} \cap C_{i,j}$) is convex [3]. We will denote it with $\mathcal{D}_{\le\delta}^{(i,j)}$.

Furthermore, the Fréchet distance between two given curves is less or equal to δ if and only if there exists a monotone path in the free space that starts in the lower left corner $(0, 0)$ and ends in the upper right corner $(1, 1)$ of the free space diagram [3]. For the shortcut Fréchet distance, we need to also allow shortcuts. This is captured in the concept of tunnels in free space. A shortcut segment $\overline{B}[y_\mathsf{p}, y_\mathsf{q}]$ and the subcurve $T[x_\mathsf{p}, x_\mathsf{q}]$ it is being matched to, correspond in the parametric space to a segment $\mathsf{pq} \subseteq [0, 1]^2$, called a *tunnel* and denoted by $\tau(\mathsf{p}, \mathsf{q})$, where $\mathsf{p} = (x_\mathsf{p}, y_\mathsf{p})$ and $\mathsf{q} = (x_\mathsf{q}, y_\mathsf{q})$. We require $x_\mathsf{p} \le x_\mathsf{q}$ and $y_\mathsf{p} \le y_\mathsf{q}$ for monotonicity. We call the Fréchet distance of the shortcut segment to the subcurve the *price* of this tunnel and denote it with $\mathrm{prc}_\tau(\mathsf{p}, \mathsf{q}) = \mathsf{d}_\mathcal{F}\big(T[x_\mathsf{p}, x_\mathsf{q}], \overline{B}[y_\mathsf{p}, y_\mathsf{q}]\big)$. A tunnel $\tau(\mathsf{p}, \mathsf{q})$ is *feasible* for δ if $\mathsf{p}, \mathsf{q} \in \mathcal{D}_{\le\delta}(T, B)$.

Now, we define the *reachable free space*, as follows

$$\mathcal{R}_{\le\delta}(T, B) = \left\{ (x_\mathsf{p}, y_\mathsf{p}) \in [0, 1]^2 \;\middle|\; \mathsf{d}_\mathbb{S}(T[0, x_\mathsf{p}], B[0, y_\mathsf{p}]) \le \delta \right\}.$$

From now on, we will simply write $\mathcal{R}_{\le\delta}$. This is the set of points that have an (x, y)-monotone path from $(0, 0)$ that stays inside the free space and otherwise uses tunnels. We will denote the reachable space inside a cell (i.e., $\mathcal{R}_{\le\delta} \cap C_{i,j}$) with $\mathcal{R}_{\le\delta}^{(i,j)}$.

Monotonicity of tunnel prices. In order to prevent the reachable space from being fragmented, as it is the case with the exact problem we showed to be NP-hard, we will approximate it. For this, we will use the following lemma by Driemel and Har-Peled [15].

LEMMA 5 ([15]). *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 $\overline{B}_1$ and $\overline{B}_2$, such that $\mathsf{d}_\mathcal{F}(T_1, \overline{B}_1) \le \delta$ and the start (resp. end) point of T_2 is in distance δ to the start (resp. end) point of $\overline{B}_2$, then $\mathsf{d}_\mathcal{F}(T_2, \overline{B}_2) \le 3\delta$.*

Horizontal, vertical and diagonal tunnels. We can distinguish three types of tunnels. We call a tunnel that stays within a column of the grid, a *vertical tunnel*. Likewise, a tunnel that stays within a row is called a *horizontal tunnel*. Tunnels that span across rows and columns are *diagonal tunnels*. Note that vertical tunnels that are feasible for a value of δ also have a price at most δ. Furthermore, the shortcut which corresponds to a horizontal tunnel lies within an edge of

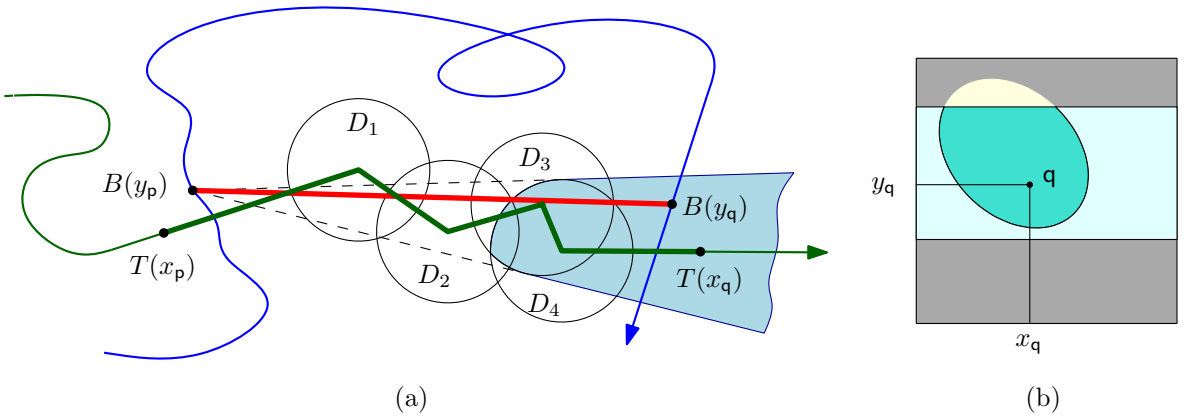


Figure 4: (a) Example of a tunnel $\tau(\mathsf{p},\mathsf{q})$ computed by the **diagonalTunnel** procedure. The shaded area shows the line-stabbing wedge. (b) The free space cell which contains the endpoint of the tunnel.

the input curve. Thus, shortcutting the curve does not have any effect in this case and we can safely ignore such horizontal tunnels.

Ordered line stabbing. Guibas et al. [17] study the problem of stabbing an ordered set of unit radius disks with a line. In particular, one of the problems studied is the following. Given a series of unit radius disks $D_1, \ldots, D_n$, does there exist a directed line ℓ, with n points p_i, which lie along ℓ in the order of i, such that $\mathsf{p}_i \in D_i$? As was already noted by Guibas et al., their techniques can be applied to decide whether there exists a line segment that lies within Fréchet distance one to a given polygonal curve X. Simply center the disks at the vertices of X in their order along the curve and the relationship follows from the fact that the Fréchet distance between two line segments is the maximum of the distances of their endpoints.

The algorithm described by Guibas et al. maintains a so called *line-stabbing wedge* that contains all points p, such that there is a line through p that visits the first i disks before visiting p. The algorithm runs in $O(n \log n)$ time. We will use this algorithm, to compute all tunnels of price at most δ starting from a particular point in the parametric space and ending in a particular cell.

4.2 Approximate decision algorithm

We describe an approximate decision algorithm for the directed continuous shortcut Fréchet distance. Given a value of δ and two polygonal curves T and B in $\mathbb{R}^2$ of total complexity $n = n_1 + n_2$, the algorithm outputs either (i) "$d_S(T,B) \leq 3\delta$", or (ii) "$d_S(T,B) > \delta$". The algorithm runs in $O(n^3 \log n)$ time and $O(n)$ space. We first discuss the challenges and then give a sketch of the algorithm.

Challenge and ideas. The standard way to solve the decision problem for the Fréchet distance and its variants is to search for monotone paths in the free space diagram. In the case of the shortcut Fréchet distance, this path can now use tunnels in the free space diagram, which correspond to shortcuts on B which are matched to subcurves of T. In the general version of the shortcut Fréchet distance, the tunnels can now start and end anywhere inside the free space cells, while in the vertex-restricted case they are constrained to the grid of the parametric space. In order to extend the algorithm by Driemel and Har-Peled [15] to this case, we need a new method to compute the space which is reachable within a free space cell.

We use the concept of line-stabbing to compute all tunnels $\tau(\mathsf{p},\mathsf{q})$ of price at most δ starting from a particular point p in the parametric space and ending in a particular cell. By intersecting the line-stabbing wedge with the edge of B that corresponds to the cell, we obtain a horizontal strip which represents the set of such tunnel endpoints q. See Figure 4 for an illustration.

The second challenge is that the reachable free space can fragment into exponentially many of such horizontal strips. However, we can exploit the monotonicity of the tunnel prices to approximate the reachable free space as done in the algorithm by Driemel and Har-Peled. In this approximation scheme, the combinatorial complexity of the reachable space is constant per cell. Thus, the overall complexity is bounded by $O(n^2)$. Our algorithm takes $O(n \log n)$ time per cell, resulting in $O(n^3 \log n)$ time overall.

Driemel and Har-Peled make certain assumptions on the input curves and achieve a near-linear running time. Instead of traversing $O(n^2)$ cells, they considered only those cells that intersect the free space on their boundary. To compute these cells, a data structure of de Berg and Streppel [12] was used. Unfortunately, this method does not immediately extend to our case, since we also need to consider cells that intersect the free space in their interior only. Therefore, our algorithm traverses the entire free space diagram yielding a much simpler, yet slower algorithm, without making assumptions on the input curves.

Algorithm sketch. We traverse the free space as usual to compute the reachable free space. In each cell, in addition to the reachable free space from neighboring cells, we also compute the free space reachable by a shortest tunnel at price 3δ. For this, we store (or find) the right-most point in the free space below and to the right of the current cell, i.e., the point that will give the shortest tunnel. We modify the line-stabbing algorithm of Guibas et al. to compute the free space reachable by a tunnel from this point (see **diagonalTunnel** below). Now, by Lemma 5, we know that any point not reachable by a shortest tunnel at price 3δ is also not reachable by a longer tunnel at price δ. Because we compute reachability by only one tunnel, the free space fragments only in a constant

number of pieces. In this way, we obtain a 3-approximation to the decision version of the problem.

The tunnel procedures. The **diagonalTunnel** procedure receives as input a cell $C_{i,i}$ and a point $\mathsf{p} \in \mathcal{D}_{\leq\delta}$, such that p lies in the lower left quadrant of the lower left corner of the cell $C_{i,i}$ and a parameter $\delta > 0$. The output will be a set of points $\mathsf{P} \subseteq C_{i,i}$, such that $\mathrm{prc}_\tau(\mathsf{p},\mathsf{q}) \leq \delta$ if and only if $\mathsf{q} \in \mathsf{P}$. We use the line-stabbing algorithm mentioned above with minor modifications. Let $x_\mathsf{p} = x_-$ be the x-coordinate of p and let x_+ be the x-coordinate of some point in the interior of $C_{i,i}$. Let $D_1, \ldots, D_k$ be the disks of radius δ centered at the vertices of T that are spanned by the subcurve $T[x_-, x_+]$. There are two cases, either $B(y_\mathsf{p})$ is contained in D_i for all $1 \leq i \leq k$, or there exists some i, such that $B(y_\mathsf{p})$ lies outside of D_i. In the first case, we return $\mathsf{P} = C_{i,i} \cap \mathcal{D}_{\leq\delta}$. In the second case we initialize the line-stabbing wedge of [17] with the tangents of $B(y_\mathsf{p})$ to the disk D_i, where i is the smallest index such that $B(y_\mathsf{p}) \notin D_i$. We then proceed with the algorithm as written by handling the disks $D_{i+1}, \ldots, D_k$. Finally, we intersect the line-stabbing wedge with the edge of B that corresponds to $C_{i,i}$. Refer to Figure 4 for an illustration. This yields a horizontal slab of points that lie in $C_{i,i}$ which we then intersect with the δ-free space and return as our set P.

The **verticalTunnel** procedure receives as input a cell $C_{i,i}$ and a point p which lies below this cell in the same column and a parameter $\delta \geq 0$. Let H_p be the closed halfplane which lies to the right of the vertical line through p. The procedure returns the intersection of H_p with the δ-free space in $C_{i,i}$. This is illustrated in Figure 5.

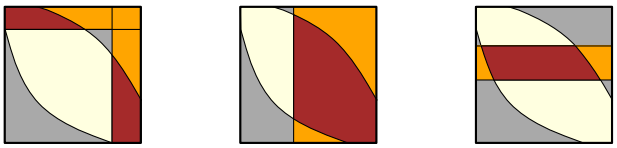


Figure 5: Examples of the approximate reachable free space in one cell: reachable by lower or left boundary (left), by a vertical tunnel (center), or a diagonal tunnel (right).

Result. The algorithm is described and analysed in detail in the full version [11]. We obtain the following theorem.

THEOREM 2. *Given two curves T and B of complexity $n = n_1 + n_2$ and a value $\delta > 0$, the decision algorithm outputs one of the following, either*
(i) $d_\mathcal{S}(T, B) \leq 3\delta$, or
(ii) $d_\mathcal{S}(T, B) > \delta$.
In any case, the output is correct. The algorithm runs in $O(n^3 \log n)$ time and $O(n)$ space.

4.3 Exact algorithm for vertex-restricted case

A similar strategy as the approximation algorithm for the general case gives an exact algorithm for deciding the vertex-restricted case. That is, given two polygonal curves T, B, and $\delta > 0$, we want to decide wether $\mathsf{d}_\mathcal{S}(T, B) \leq \delta$. For this, we again traverse the free space, and in each cell compute, additionally to the reachable free space from neighboring cells, the free space reachable using shortcuts between vertices. We observe that in the vertex-restricted case, tunnels can start and end only on grid lines, and hence the free space does not fragment. In the following, we assume that shortcuts may be taken on the curve corresponding to the vertical axis of the free space diagram, i.e., between horizontal grid lines.

Now, in each cell, instead of testing the shortest tunnel (as in the approximation algorithm), we need to test all tunnels between the upper horizontal cell boundary and (at most n^2) horizontal cell boundaries left and below the current cell. In fact, by the following lemma, (which is similar to Lemma 5) we only need to test each shortcut with the shortest possible subcurve of the other curve. Thus, we only need to test n tunnels.

LEMMA 6. *Let $\overline{B} = qq'$ be a segment, and let T_2 be a subcurve of T_1, s.t. the start and end points of T_2 have distance at most δ to q and q', respectively. Then it holds $\mathsf{d}_\mathcal{F}(T_1, \overline{B}) \leq \delta \Rightarrow \mathsf{d}_\mathcal{F}(T_2, \overline{B}) \leq \delta$.*

PROOF. Let σ be a homeomorphism realizing a distance $\delta \geq \delta$ between $\overline{B}$ to T_1. We can easily modify σ to a homeomorphism σ' realizing at most the same distance δ between $\overline{B}$ to T_2, as illustrated in Figure 6. □

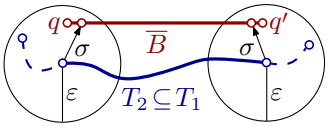


Figure 6: Illustration to proof of Lemma 6.

Now, assume we are handling free space cell C_{ij}. First, we compute reachability from neighboring cells, as usual. Next we consider reachability by tunnels. The lemma above implies, that for each of the $j-1$ possible shortcuts (starting at $q_h < q_j$ and ending at q_j), we only need to test the tunnel corresponding to the shortest possible subcurve on T, i.e., starting at the rightmost point on T. If this tunnel has a price larger than δ, then by the lemma so do all other tunnels starting at $p'_l < p_l$. If this tunnel has price at most δ, then the complete upper cell boundary is reachable and we do not need to test further tunnels. Thus, for all $h < j$ we test whether the tunnel from the rightmost point p_l to the leftmost point p_k on the current upper cell boundary has price $\leq \delta$. For this, we maintain for each vertex q_h on B the rightmost point p_l on T such that (p_l, q_h) is in reachable free space. This can be updated in constant time per cell, and linear space in total. To test all $l-1$ possible tunnels per cell, we use a similar strategy as for the approximation algorithm in the previous section. We build the line stabbing wedge, from "left to right", i.e., starting at q_j, and adding disks $p_i, p_{i-1}, \ldots$ For each $h < j$ we test if q_h is in the wedge for the corresponding p_l.

The modified tunnel procedure for cell C_{ij} takes $O(i \log i)$ time for computing the line-stabbing wedge and $O(j)$ time for testing tunnels, giving $O(i \log i + j) = O(n \log n)$ time in total. Thus, we can handle the complete free space diagram in $O(n^3 \log n)$ time.

The correctness and runtime analysis of the algorithm follow in the lines of the approximation algorithm. We conclude with the following theorem.

THEOREM 3. *Given two curves T and B and a value $\delta > 0$. One can decide whether the vertex-restricted shortcut Fréchet distance between T and B is less than or equal to δ in $O(n^3 \log n)$ time and $O(n)$ space.*

5. CONCLUSIONS

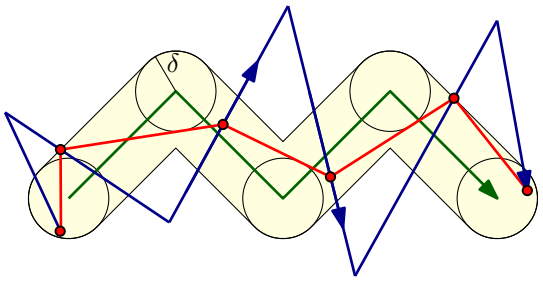
In this paper we studied the computational complexity of the shortcut Fréchet distance, that is the minimal Fréchet distance achieved by allowing shortcuts on one of two polygonal curves. Our main result is to prove NP-hardness of this problem and doing so, we provided the first NP-hardness result for a variant of the Fréchet distance between two polygonal curves in the plane.

It may also be interesting to allow non-simultaneous shortcuts on both curves as done in [4]. We note that our reduction trivially extends to this case. Shortcuts on the target curve can only either shorten or eliminate twists. However, any such shortcut on the target curve would have to be (at least partially) matched to a shortcut of the base curve and this is prevented by definition. Hence computing this two-sided shortcut Fréchet distance is also NP-hard.

Several open problems remain. Our reduction from SUBSET-SUM proves that the problem is weakly NP-hard. It is left open to determine whether it is strongly NP-hard and whether it even lies in NP. The base curve in our reduction cannot be c-packed for any placement of connector edges for any constant c. We do not know if the problem allows a polynomial-time solution for this restricted class of curves. Another open question is whether deciding the shortcut Fréchet distance is fixed-parameter tractable in the number of shortcuts. It may be possible to extend our construction to allow the choice between n different values using a single split gadget. This could be used in an FPT-reduction from k-SUM.

Finally, it is unknown if one can compute the shortcut Fréchet distance between two given curves. We note that one can approximate it up to additive error via the vertex-restricted case by sampling points on the curves.

The standard way to compute the Fréchet distance is to use a decision procedure in a binary (or parametric) search over critical values. In our case, the event that a monotone path in the free space diagram becomes feasible could involve a linear number of shortcuts, as the following example shows. A full characterization of critical values is yet to be done.



Acknowledgements. We thank Maarten Löffler for insightful discussions on the NP-hardness construction, and Sariel Har-Peled and anonymous referees for many helpful comments.

6. REFERENCES

[1] P. K. Agarwal, S. Har-Peled, N. H. Mustafa, and Y. Wang. Near-linear time approximation algorithms for curve simplification. *Algorithmica*, 42:203–219, 2005.

[2] H. Alt, A. Efrat, G. Rote, and C. Wenk. Matching planar maps. *J. Algorithms*, 49:262–283, 2003.

[3] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *Internat. J. Comput. Geom. Appl.*, 5:75–91, 1995.

[4] R. B. Avraham, O. Filtser, H. Kaplan, M. J. Katz, and M. Sharir. The discrete Fréchet distance with shortcuts via approximate distance counting and selection techniques. In *Proc. 30th ACM Sympos. Comput. Geom.*, 2014.

[5] S. Bereg, M. Jiang, W. Wang, B. Yang, and B. Zhu. Simplifying 3d polygonal chains under the discrete Fréchet distance. In *Proc. 8th Latin American Conf. on Theoretical Informatics*, pages 630–641, 2008.

[6] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *Proc. 31st Internat. Conf. on Very Large Data Bases*, pages 853–864, 2005.

[7] K. Buchin, M. Buchin, and J. Gudmundsson. Detecting single file movement. In *Proc. 16th ACM Internat. Conf. Adv. GIS*, pages 288–297, 2008.

[8] K. Buchin, M. Buchin, J. Gudmundsson, M. Löffler, and J. Luo. Detecting commuting patterns by clustering subtrajectories. *Internat. J. Comput. Geom. Appl.*, 21(03):253–282, 2011.

[9] K. Buchin, M. Buchin, W. Meulemans, and W. Mulzer. Four Soviets walk the dog—with an application to Alt's conjecture. In *Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA14)*, pages 1399–1413, 2014.

[10] K. Buchin, M. Buchin, and Y. Wang. Exact algorithm for partial curve matching via the Fréchet distance. In *Proc. 20th ACM-SIAM Sympos. Discrete Algorithms*, pages 645–654, 2009.

[11] M. Buchin, A. Driemel, and B. Speckmann. Computing the Fréchet distance with shortcuts is NP-hard. *CoRR*, abs/1307.2097, 2013.

[12] M. de Berg and M. Streppel. Approximate range searching using binary space partitions. *Computational Geometry: Theory and Appl.*, 33(3):139 – 151, 2006.

[13] J.-L. De Carufel, A. Gheibi, A. Maheshwari, J.-R. Sack, and C. Scheffer. Similarity of polygonal curves in the presence of outliers. *Computational Geometry: Theory and Appl.*, 47(5):625–641, 2014.

[14] A. Driemel. *Realistic Analysis for Algorithmic Problems on Geographical Data*. PhD thesis, Utrecht University, 2013. Submitted.

[15] A. Driemel and S. Har-Peled. Jaywalking your dog – computing the Fréchet distance with shortcuts. In *Proc. 23rd ACM-SIAM Sympos. Discrete Algorithms*, pages 318–337, 2011.

[16] A. Driemel, S. Har-Peled, and C. Wenk. Approximating the fréchet distance for realistic curves in near linear time. *Discrete & Computational Geometry*, 48(1):94–127, 2012.

[17] L. J. Guibas, J. Hershberger, J. S. B. Mitchell, and J. Snoeyink. Approximating polygons and subdivisions with minimum link paths. In *Proc. 2nd Internat. Sympos. Algorithms*, pages 151–162, 1991.

[18] A. Mascret, T. Devogele, I. L. Berre, and A. Hénaff. Coastline matching process based on the discrete Fréchet distance. In *Proc. 12th Internat. Symposium on Spatial Data Handling*, pages 383–400, 2006.

[19] T. Wylie and B. Zhu. A polynomial time solution for protein chain pair simplification under the discrete Fréchet distance. In *Proc. 8th Internat. Symposium on Bioinf. Research and Appl.*, volume 7292 of *LNCS*, pages 287–298, 2012.