On Computing the $k$-Shortcut Fréchet Distance

JACOBUS CONRADI, University of Bonn, Institute for Computer Science V, Germany
ANNE DRIEMEL, Hausdorff Center for Mathematics, Germany

The Fréchet distance is a popular measure of dissimilarity for polygonal curves. It is defined as a min-max formulation that considers all orientation-preserving bijective mappings between 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 $\delta$, 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 $\delta$? We study this problem for polygonal curves in the plane. We provide a complexity analysis for this problem with the following results: (i) there exists a decision algorithm with running time in $O(kn^{2k+2}\log n)$; (ii) assuming the exponential-time-hypothesis (ETH), there exists no algorithm with running time bounded by $n^{o(k)}$. 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 $O(kn^2\log^2 n)$ for fixed $\varepsilon$. 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.

CCS Concepts: • Theory of computation → Approximation algorithms analysis. Computational geometry.

Additional Key Words and Phrases: Fréchet distance, Partial similarity, Conditional lower bounds, Approximation algorithms

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 [Su et al. 2020]. 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 orientation-preserving bijective mappings 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 $\delta$ 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 $\delta$ 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 [Buchin et al. 2009] 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 [Driemel and Har-Peled 2012] 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 [Akitaya et al. 2017] presented an $(\alpha + \varepsilon)$-approximate decider algorithm with running time in $O(n^2\log n)$ for fixed $\varepsilon$. In contrast, we 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.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2024 Copyright held by the owner/author(s).
ACM 1549-6333/2024/5-ART
https://doi.org/10.1145/3663762
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 [Bronstein et al. 2009; Jacobs et al. 2000; Veltkamp 2001].

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. While the (discrete or continuous) Fréchet distance can be computed in roughly quadratic (in n) time for two polygonal curves of n vertices [Alt and Godau 1995; Aronov et al. 2006; Bringmann 2014; Bringmann and Mulzer 2016; Buchin et al. 2017, 2019], the overall picture on the computational complexity of the partial variants is very heterogeneous.

De Carufel, Gheibi, Maheshwari, and Sack [De Carufel et al. 2014] showed that the problem of computing the partial Fréchet distance is not solvable by radicals over 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 [Buchin et al. 2009]. Computing the shortcut Fréchet distance was shown to be NP-hard [Buchin et al. 2014] 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, Filtser, Kaplan, Katz, and Sharir [Avraham et al. 2015], which is even faster than computing the discrete Fréchet distance without shortcuts. The variant defined by Akitaya, Buchin, Ryvkin, and Urhausen [Akitaya et al. 2019] 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 [Driemel and Har-Peled 2012] introduced the shortcut Fréchet distance and described a near-linear time (3 + ε)-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 [Buchin et al. 2014] 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 \( O(n^3 \log n) \) for the case that shortcuts may start and end anywhere along edges. Prior to our work, there has been no study of exact algorithms for the non-restricted shortcut Fréchet distance. 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 \( T \) of complexity \( n \) is given by a sequence of \( n \) points \( p_1, \ldots, p_n \) in \( \mathbb{R}^d \) referred to as its vertices. The curve is then defined as the piecewise linear interpolation \([0, 1] \ni t \mapsto t p_i + (1 - t) p_{i+1}\) between consecutive vertices \( p_i \) and \( p_{i+1} \), for all \( i < n \).
**Definition 2** (Fréchet distance). Given two curves $T$ and $B$ in $\mathbb{R}^d$, the Fréchet distance between $T$ and $B$ is defined as

$$d_F(T, B) = \inf_{f, g : [0,1] \rightarrow [0,1]} \max_{t \in [0,1]} ||T(f(t)) - B(g(t))||,$$

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

In our definition of the Fréchet distance given above, we follow Alt and Godau [Alt et al. 1995]. This definition leads to the same (semi-)metric as the standard definition, which uses an infimum over orientation-preserving homeomorphisms instead of non-decreasing surjections [Deza et al. 2009; Fréchet 1906]. For a proof, see for example [Buchin et al. 2023, Lemma 2.4].

**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 $d^k_{F}(T, B)$ 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 $\delta$. The algorithm can also be used for the non-parameterized variant by setting $k = n - 1$. 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 $O(kn^{2k+2} \log n)$ and space in $O(kn^{2k+2})$ that decides whether $d^k_{F}(T, B) \leq \delta$.

Our algorithm for Theorem 5 iterates over the free-space diagram by Alt and Godau [Alt and Godau 1995] in $k$ rounds. Within the free-space diagram, a traversal 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, Hershberger, Mitchell and Snoeyink [Guibas et al. 1994]. Line-stabbing wedges were also used in the approximation algorithm by Buchin, Driemel, and Speckmann [Buchin et al. 2014]. In our case, since we perform exact computations, the reachable space fragments into a number of components, and this number may grow exponentially with the number of rounds.

In Section 4 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 [Impagliazzo and Paturi 1999]. 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 [Buchin et al. 2014]. 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 reduction by Buchin et al. 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.

In light of the above results, it is interesting to consider approximation algorithms and realistic input assumptions for this problem. In Section 5 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 [Driemel et al. 2010].

**Definition 7** \((c\text{-}packed curves)\). 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.

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

Since any polygonal curve of complexity \( n \) is \( c \)-packed for some \( c \leq 2n \), the theorem also implies a running time of \( O(kn^2\epsilon^{-5}\log^2(nc^{-1})) \) for polygonal curves in the plane—without any input assumptions.

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 a 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 decisions (within the approximation bounds), we make use of a property of shortcut prices that was first observed by Driemel and Har-Peled [Driemel and Har-Peled 2012]. 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 [Driemel and Har-Peled 2012] is that this shortcut may still start in the middle of an edge. Evaluating the cost of this shortcut using line-stabbing wedges would be too expensive. Instead, we use a data structure by Driemel and Har-Peled [Driemel and Har-Peled 2012] that allows to query the Fréchet distance of a line segment to a subcurve. We use this to implicitly approximate the line-stabbing wedge using 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 [Driemel et al. 2010], 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

**Definition 9** \((Free\text{-}space\ diagram)\). 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 \( i \)th edge of the target curve and the \( j \)th edge of the base curve \( C_{i,j} \). The \( \delta \)-free-space of \( T \) and \( B \) is defined as

\[
D_\delta(T, B) = \{(x, y) \in [0, 1]^2 \mid \|T(x) - B(y)\| \leq \delta\}
\]
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 simply will 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 \( \delta \) 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. [Alt and Godau 1995]. 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 \( \delta \), then \((f, g)\) induces traversals \((f', g')\) with associated cost at most \( \delta \) on every shortcut \( \mathcal{B}[s, t] \) and some corresponding subcurve \( T[u, v] \) of \( T \). To capture this, we use the notion of tunnels which was introduced in [Driemel and Har-Peled 2012] 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 \( x_p \leq x_q \) and \( y_p \leq y_q \). \( \tau(p, q) \) 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 \( \text{prc}(\tau(p, q)) = d_\delta(T[x_p, x_q], B[y_p, y_q]) \), refer to Figure 1.

**Definition 11** (Reachable space). We define the \((\delta, s)\)-reachable free-space of \( T \) and \( B \)
\[
\mathcal{R}_{\delta,s}(T, B) = \{(x_p, y_p) \in [0,1]^2 \mid d_\delta^s(T[0,x_p], B[0,y_p]) \leq \delta\}
\]
and again \( \mathcal{R}_{\delta,s}^{(i,j)}(T, B) = \mathcal{R}_{\delta,s}(T, B) \cap C_{i,j} \). We call the intersection \( \mathcal{R}_{\delta,s}^{(i,j)}(T, B) \cap C_{a,b} \) 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.

We will simply write \( \mathcal{R}_{\delta,s} \) and \( \mathcal{R}_{\delta,s}^{(i,j)} \) whenever \( T \) and \( B \) are fixed. Observe that the reachability intervals for every cell \( C_{i,j} \) and \( s \) are contained in the boundary set \( \partial C_{i,j} \), 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
We are given a parameter \( k \) with running time \( O(\log n) \). This line-stabbing wedge is described by \( O(n) \) circular arcs, and two tangents that go to infinity (see Figure 2). Guibas et al. described a connection between ordered stabbers and the Fréchet distance, which can be reformulated in terms of feasible tunnels as follows.

**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 \( n \) objects, with running time \( O(n \log n) \). This line-stabbing wedge is described by \( O(n) \) circular arcs, and two tangents that go to infinity (see Figure 2). Guibas et al. described a connection between ordered stabbers and the Fréchet distance, which can be reformulated in terms of feasible tunnels as follows.

**Observation 16.** Let \( T, B \) and \( \delta \) be given. Denote by \( v_1, \ldots, v_n \) 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 \( B[y_p, y_q] \) stabs through the ordered set \( \{ b_\delta(v_{a+1}), \ldots, b_\delta(v_i) \} \), if and only if \( \operatorname{pre}(\tau(p, q)) \leq \delta \).

### 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 \( k = n - 1 \). We first describe the algorithm in Section 3.1 and then analyse its correctness and running time in Section 3.2.

#### 3.1 Description of the algorithm

We are given a parameter \( k \), a value \( \delta \) and the two polygonal curves \( T \) and \( B \) in the plane. Our algorithm iterates over the \( \delta \)-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 \((\delta, s)\)-reachable space \(R_{\delta,s}(T, B)\) 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.

2. The monotone path could reach \(C_{i,j}\) with a proper tunnel (only for \(s \geq 1\)). We distinguish between vertical and diagonal tunnels (compare [Buchin et al. 2014; Driemel and Har-Peled 2012] 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.

Note that we do not consider (horizontal) tunnels starting in a cell \(C_{a,j}\) with \(a < i\), since we only consider proper tunnels. 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 \(B\) with \(\%_{B}^{(i,j)}\). The \((\delta, s)\)-reachable space is then obtained by taking the union of these sets over all rounds \(R_{\delta,s}^{(i,j)} = \bigcup_{0 \leq s' \leq s} \%_{s'}^{(i,j)}\).

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 \(\delta(T, B) \leq \delta\) otherwise the algorithm returns \(\delta(T, B) > \delta\).

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

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

We will usually intersect this set with \(D_{\delta}^{(i,j)}\) to obtain all points that are reachable from a point of \(P\) by a monotone path that stays inside the \(\delta\)-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 \(D_{\delta}^{(i,j)}\), 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 \(P_{i-1,j}^s\) and \(P_{i,j-1}^s\) by the time we handle cell \(C_{i,j}\) in round \(s\). Therefore, we can compute the incoming reachability intervals by intersecting \(P_{i-1,j}^s\) and \(P_{i,j-1}^s\) with \(C_{i,j}\). Now we apply the
function $Q$ to these sets and denote the result with $N_{i,j}^{s}$:

$$N_{i,j}^{s} = \left( Q(P_{i-1,j}^{s} \cap C_{i,j}) \cup Q(P_{i,j-1}^{s} \cap C_{i,j}) \right) \cap D_{\delta}^{(i,j)}$$

Refer to Figure 3 a).

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

The procedure is given a set of points $P_{a,b}^{s-1}$ in the $\delta$-free-space $D_{\delta}^{(a,b)}$ and computes all points in $D_{\delta}^{(i,j)}$ that are endpoints of tunnels starting in $P_{a,b}^{s-1}$ with price at most $\delta$. The procedure first projects $P_{a,b}^{s-1}$ onto the edge $e_{b}$ of the base curve. The resulting set consists of disjoint line segments $I = \{s_{1}, t_{1}, \ldots\}$ along $e_{b}$ (refer to Figure 3 d)). The procedure then computes the line-stabbing wedge $W$ through $s_{1}t_{1}$ and disks $b_{\delta}(v_{a+1}), \ldots, b_{\delta}(v_{b})$ 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 $D_{\delta}^{(i,j)}$ to obtain all endpoints of feasible shortcuts with price at most $\delta$ starting in $s_{1}t_{1}$. The procedure performs the above steps for every line segment $s_{1}t_{1} \in I$ and returns the union of these sets. The resulting set may look as illustrated in Figure 3 c).

**Step 2 (ii): Vertical tunnels.** (only for $s \geq 1$) Let $p$ denote a point in $\bigcup_{1 \leq j \leq 1} P_{i,j}^{s-1}$ with minimal $x$-coordinate, i.e., a leftmost point in this set. A feasible vertical tunnel always has price at most $\delta$. Therefore, we simply take all points in the $\delta$-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 $\delta$-free-space in $C_{i,j}$. We denote this set with $V_{i,j}^{s}$. Refer to Figure 3 b) for an example.

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

$$P_{i,j}^{s} = Q(N_{i,j}^{s} \cup D_{i,j}^{s} \cup V_{i,j}^{s}) \cap D_{\delta}^{(i,j)}$$

It remains to specify the initialization: We define $P_{1,1}^{0} = D_{\delta}^{(1,1)}$, if $(0, 0) \in D_{\delta}$, and otherwise $P_{1,1}^{0} = \emptyset$. Starting from this, we can compute the sets $P_{i,j}^{s}$ for $i, j \neq 1$ in a row-by-row fashion. For $s > 0$ we continue in rounds, as described above.
We now analyse the described algorithm.

3.2 Analysis

We now analyse the described algorithm.

3.2.1 Correctness. We argue that the structure of $B_{a,b}^s$ as computed by the algorithm is indeed as claimed. Namely for all $i,j$ and $s$ it holds that $R_{\delta,s}^{(i,j)} = \bigcup_{0 \leq s' \leq s} p_{a,b}^{s'}$. We begin with two lemmas, before we prove this statement.

**Lemma 17.** Let $T$ and $B$ be two polygonal curves with $n_1$ and $n_2$ edges respectively. For any $1 \leq i \leq n_1$, $1 \leq j \leq n_2$ and $1 \leq s \leq k$ let $D_{i,j}^s$ be the set of endpoints of diagonal tunnels, as computed in the algorithm described in Section 3.1, and let $R = \bigcup_{s=1}^{s_{\max}} \bigcup_{i,j} p_{a,b}^{s_{\max}}$ be the set of reachable points by exactly $s-1$ proper tunnels in the lower-left quadrant of $C_{i,j}$. For any $q \in D_{i,j}^s$ the tunnel $\tau(p,q)$ has price $\text{prc}(\tau(p,q)) \leq \delta$ for some $p \in R$ if and only if $q \in D_{i,j}^s$.

**Proof.** First let $a$ and $b$ be fixed and look at $P = P_{a,b}^{s_{\max}}$. The diagonal tunnel procedure begins by projecting $P$ onto the edge $e_i$ of $B$, resulting in $I$. By the correctness of the procedure presented by Guibas et al. the diagonal tunnel procedure computes among other things the set of points in $\mathbb{R}^2$ that are all endpoints of stabbers through $I$ and $b_{\delta}(v_{a+1}), \ldots, b_{\delta}(v_l)$ centered at vertices of $T$. Intersecting this set with $e_j$ results in all endpoints of stabbers through the ordered set ending on $e_j$, call this set $J$. For every point $B(y_p,y_q)$ in $I$, there is at least one point $B(y_p)$ in $I$, such that $B(y_p,y_q)$ stabs through $(b_{\delta}(v_{a+1}), \ldots, b_{\delta}(v_l))$. Hence, by Observation 16, every point $p \in D_{s}^{(a,b)}$, with $y$-coordinate $y_p$ and every point $q \in D_{s}^{(i,j)}$ with $y$-coordinate $y_q$ form a feasible tunnel $\tau(p,q)$ with price at most $\delta$. Since the line-stabbing algorithm correctly computes all possible endpoints of stabbers starting in $I$ and ending on $e_j$, the claim follows for $P_{a,b}^{s_{\max}}$ by Observation 16. That is, any $q$ such that there exists $p \in R$ with $\text{prc}(\tau(p,q)) \leq \delta$ also must be in $D_{i,j}^s$. As the algorithm iterates over all cells in the lower-left quadrant of $C_{i,j}$ and in the end defines $D_{i,j}^s$ as the union of above computed sets, the claim follows. \hfill $\square$

**Lemma 18.** Let $T$ and $B$ be two polygonal curves with $n_1$ and $n_2$ edges, respectively. For any $1 \leq i \leq n_1$, $1 \leq j \leq n_2$ and $1 \leq s \leq k$ let $V_{i,j}^s$ be the points reachable by a vertical tunnel as computed in the algorithm and let $R = \bigcup_{i,j} p_{a,b}^{s_{\max}}$ be the set of reachable points by exactly $s-1$ proper tunnels in the column below $C_{i,j}$. For any $q \in C_{i,j}$ the tunnel $\tau(p,q)$ has price $\text{prc}(\tau(p,q)) \leq \delta$ for some $p \in R$ if and only if $q \in V_{i,j}^s$.

**Proof.** Note that any vertical tunnel costs at most $\delta$ if it is feasible by Observation 14. Furthermore note that the leftmost point $p$ in $R$ is stored in $l_{i,j-1}^{s_{\max}}$, hence, we can retrieve $p$. Now assume $\tau(r,q)$ is an arbitrary vertical tunnel with $r \in R$ and $q \in C_{i,j}$. Since a tunnel must be monotone $x_r \leq x_q$. Because $p$ is the leftmost point in $R$ we have
which \% \> 0. From the way we constructed \( V_{i,j}^s \) (intersecting a vertical closed halfplane to the right of \( p \) with \( D_{\delta}^{(i,j)} \)) it follows that \( q \in V_{i,j}^s \).

**Theorem 19.** Let \( T \) and \( B \) be two polygonal curves in the plane with overall complexity \( n \) together with a value \( \delta > 0 \). Let \( P_{i,j}^s \) be the set of reachable points with exactly \( s \) proper tunnels as computed in the algorithm for all \( i, j \) and \( s \). It holds that

\[
\bigcup_{s \leq s} P_{i,j}^s = R_{\delta,\delta}^{(i,j)}(T, B).
\]

Thus the algorithm correctly decides, whether the \( k \)-shortcut Fréchet distance of \( T \) and \( B \) is at most \( \delta \).

**Proof.** We show that the reachable space \( R_{\delta,\delta}^{(i,j)} \) is correctly computed via induction in \( i, j \) and \( s \). Note that \( R_{\delta,\delta}^{(i,j)} \) is computed correctly for all \( s' \leq k \) since \( D_{\delta}^{(i,j)} \) is convex and the algorithm checks whether \( (0, 0) \in D_{\delta}^{(i,j)} \). Thus, if \( (0, 0) \in D_{\delta}^{(i,j)} \), \( R_{\delta,\delta}^{(i,j)} = D_{\delta}^{(i,j)} = P_{i,j}^0 \) is computed in the first step, by convexity of \( D_{\delta}^{(i,j)} \), otherwise it is empty. For \( s' > 0 \) the set \( P_{i,j}^{s'} \) is empty since no cell is below or to the left of it. Hence, \( R_{\delta,\delta}^{(i,j)} = R_{\delta,\delta}^{(i,j)} \) is also computed correctly.

By induction all cells \( C_{\leq m, \leq j} \) and \( C_{<i, j} \) and in particular \( C_{i-1, j} \) and \( C_{i, j-1} \) have been handled correctly up to round \( s \) and \( P_{i,j}^s \) and is stored for every correctly handled cell. Assume now that some point \( q \in R_{\delta,\delta}^{(i,j)} \) is given. By Observation 13, the point \( q \) corresponds to a monotone path with \( s' \leq s \) proper tunnels. There are three possible ways via which this point in the parametric space is reachable. The path reaching \( q \) could take \( s' \) shortcuts to reach \( C_{i-1, j} \) or \( C_{i, j-1} \), and enter via a monotone path through the boundary into \( C_{i,j} \) at some point \( a \in \partial C_{i,j} \). As \( C_{i-1, j} \) and \( C_{i, j-1} \) have been handled correctly for \( s' \), the incoming reachability intervals on the boundary have been computed correctly containing \( a \), thus \( q \) is also in \( P_{i,j}^{s'} \).

Alternatively the path could enter some cell \( C_{i,j} \) with \( s' = 1 \) shortcuts and then take a vertical shortcut into \( C_{i,j} \) for some \( i < j \) and then possibly taken another monotone path inside the cell to \( q \). Lemma 18 implies that \( q \) is in \( P_{i,j}^{s'} \).

Lastly the path could take a diagonal shortcut and could similarly end with a monotone path inside \( C_{i,j} \) to \( q \). Lemma 17 implies that \( q \) then again is in \( P_{i,j}^{s'} \).

Now let \( q \in P_{i,j}^{s'} \) for \( s' \leq s \). Then \( q \) is either in (i) \( N_{i,j}^{s'} \), (ii) \( V_{i,j}^{s'} \), (iii) \( D_{i,j}^{s'} \) or (iv) reachable by a monotone path from some point \( q' \) in one of the preceding cases. Thus we can reduce this to the first three cases.

However Cases (i) follow immediately since cells \( C_{i,j-1} \) and \( C_{i-1,j} \) have been handled correctly up to round \( s' \), and thus \( q \) must also be in \( R_{\delta,\delta}^{(i,j)} \).

For Cases (ii) and (iii) Lemma 18 and Lemma 17 imply that \( q \) must be in \( R_{\delta,\delta}^{(i,j)} \) respectively. Thus \( R_{\delta,\delta}^{(i,j)}(T, B) = \bigcup_{s' \leq s} P_{i,j}^{s'} \).

As we store the reachable space and the leftmost point, this information is available for all upcoming iterations. \( \square \)

### 3.2.2 Running time.

**Lemma 20.** 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 \( O(n^{2k+2} \log n) \) and uses \( O(n^{2k+2}) \) space.

**Proof.** Note that the sets \( N_{i,j}^{s}, D_{i,j}^{s} \) and \( V_{i,j}^{s} \) computed by the algorithm are described as intersections of \( D_{\delta}^{(i,j)} \) with halfplanes, and unions of these. For a fixed \( P_{i,j}^{s} \) we define \( n_{i,j,s} \) as the total number of such operations from which \( P_{i,j}^{s} \) was obtained. As such, \( O(n_{i,j,s}) \) bounds the complexity of this set.
The complexity of \( N_{i,j} \) and \( V_{i,j} \) is constant. The complexity of \( D_{i,j} \) is bounded by the sum of the complexities of all cells to the lower left:

\[
    n_{i,j,s} \in O \left( \sum_{b \leq a < i} \sum_{0 \leq b < j} n_{a,b,s-1} \right).
\]

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

Computing \( D_{i,j} \) takes \( O(\sum_{a < i} \sum_{b < j} n_{a,b,s-1} \log n + n^3 \log n) = O(n^{2k} \log n) \) time. This follows from the fact, that we compute \( O(n) \) 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 \( P_{a,b}^s \). Computing \( N_{i,j}^s \) takes \( O(n_{i-1,i,s} + n_{i,j-1,s}) = O(n^{2k}) \) time, as we need to compute the reachability intervals from neighbouring cells. Computing \( V_{i,j}^s \) takes \( O(\sum_{b < j} n_{i,b,s-1}) = O(n^{2k-1}) \) time, as we need to compute the leftmost point \( l_{i,j-1}^s \). The space required to store \( P_{i,j}^s \) as required by latter iterations and cells is in \( O(n^{2k}) \). Computing \( O(N_{i,j}^s \cup V_{i,j}^s \cup D_{i,j}^s) \) takes linear time in the complexity of \( N_{i,j}^s \cup V_{i,j}^s \cup D_{i,j}^s \), i.e. \( O(n^{2k}) \).

As we do this for every cell in every round, the running time overall is \( O(kn^{2k+2} \log n) \), and the space is bounded by \( O(kn^{2k+2}) \).

Lemma 20 together with Theorem 19 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 \( n - 1 \) proper tunnels. We obtain the following corollary.

**Corollary 21.** 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 \( O(n^{2\delta+1} \log n) \) and space in \( O(n^{2\delta+1}) \) that decides whether the shortcut Fréchet distance of \( T \) and \( B \) is at most \( \delta \).

### 4 HARDNESS

We next explore conditional lower bounds for the problem of deciding, whether the \( k \)-Shortcut Fréchet Distance is at most \( \delta \). We reduce the decision problem to the \( k \)-SUM problem. More specifically, we use a variant called \( k \)-Table-SUM, which is defined as follows.

**Definition 22 (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 \( \sigma \). We want to decide whether there are indices \( i_1, \ldots, i_k \) such that \( \sum_{i=1}^{k} s_{i,i_k} = \sigma \). We call \( \sigma = \sum_{i=1}^{k} s_{i,i_k} \) the \( j \)th partial sum.

We prove that for deciding whether the \( k \)-shortcut Fréchet distance is less than or equal to a given value there exists no algorithm that runs in \( n^{o(k)} \) time, unless ETH fails. For this we construct a \((4k+2)\)-shortcut Fréchet distance instance based on a \( k \)-Table-SUM instance, where the distance is exactly 1 if and only if the \( k \)-Table-SUM instance has a solution and more than 1 otherwise.

We first establish that \( k \)-Table-SUM cannot be solved in \( n^{o(k)} \) time. This theorem is well-known by a reduction from \( k \)-SUM. We provide a proof for the sake of completeness.

**Definition 23 (k-SUM).** We are given a list \( S \) of \( n \) non-negative integers \( \{s_1, \ldots, s_n\} \) and a non-negative integer \( \sigma \). We want to decide whether there are \( k \) distinct indices \( i_1 < \ldots < i_k \) such that \( \sum_{i=1}^{k} s_{i,i_k} = \sigma \).

**Theorem 24 (Folklore).** Assuming the exponential time hypothesis, for any fixed \( k > 0 \) the \( k \)-Table-SUM problem cannot be solved in \( n^{o(k)} \) time.

**Proof.** The exponential time hypothesis states that the well known 3-SAT problem in \( n \) variables cannot be solved in \( 2^{o(n)} \) time [Impagliazzo and Paturi 1999]. Assuming the exponential time hypothesis, Pătraşcu and Williams in [Pătraşcu and Williams 2010] showed, that \( k \)-SUM cannot be solved in \( n^{o(k)} \) time.
To reduce a $k$-SUM instance to a $k$-Table-SUM instance, we begin by randomly partitioning the original integer list into $k$ non-empty parts. With probability $k^k/k^k > e^{-k}$ any given solution is then split, with one item in each of the $k$ lists. This can be derandomized, by computing a $k$-perfect family of hash functions, introduced by Schmidt and Siegel in [Schmidt and Siegel 1990]. A deterministic construction for a suitable $k$-perfect family of hash functions can found in [Alon et al. 1995] resulting in a family of size $2^{O(k)} \log n$. Thus overall we can solve one $k$-SUM instance, by solving $2^{O(k)} \log n$ instances of the $k$-Table-SUM problem. This in turn implies, that there is at least one $k$-Table-SUM instance, that can not be solved in $n^{o(k)}/(2^{O(k)} \log n) = n^{o(k)}/2^{O(k)}$ time. There exists a constant $c$, such that

$$n^{o(k)}/(2^{O(k)} \log n) \geq n^{o(k)}/(c^k \log n) = n^{o(k)}/(n^{k/\log n} \log n) \geq n^{o(k)-k/\log n-1} = n^{o(k)},$$

where the last equation holds for any fixed $k$, and any $n \geq n_0$ if we choose $n_0$ large enough. □

Without loss of generality we assume, that each table has a minimum entry of value 0. This is equivalent to the above stated $k$-Table-SUM problem by subtracting the minimum value of each list from every value of that list as well as the sum of all minimum values from the target value $\sigma$. Another slight modification we need to introduce is that all $k$ lists are required to be sorted. This reduction takes $O(k \log n)$ time by for example $k$ applications of a suitable sorting algorithm. During the construction we will always refer to this sorted version of $k$-Table-SUM.

4.1 General idea

A $k$-Table-SUM instance consists of $k$ lists of integers and a target value and asks whether the target value can be rewritten as a sum of values, one from each list. Based on such an 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 described property, that they have a distance of 1 if and only if the underlying instance has a solution.

The target curve $T$ will lie on a horizontal line going to the right. 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 on the boundary of the hippodrome. All other edges of the base curve will lie outside the hippodrome. Any shortcut curve of B that has Fréchet distance at most 1 to T we will call feasible. It is easy to see that any feasible shortcut curve must lie completely inside the hippodrome. Since any edge of the base curve inside the hippodrome lies on the boundary of it and is oriented in the opposite direction of the base curve, no feasible shortcut curve consists of any subcurve of the target curve. Hence, every shortcut on a feasible shortcut curve has to start where the previous shortcut ended. To restrict the set of feasible shortcut curves even further, we place so called twists on the target curve. A twist can only be traversed by a shortcut by going through precisely one point. We call this point the focal point or projection centre. For a simplified structural view of the curves refer to Figure 5. These twists are constructed by going a distance of 2 to the left, before continuing rightwards. 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.

We keep track of the partial sum encoded by any feasible shortcut curve by tracking the position on each mirror edge from which such a photon bounces off of i.e. where the feasible shortcut curve starts a new shortcut. By the careful placement of the mirror edges we ensure that the relative position on each mirror edge where such a photon bounces off of, is similar to the relative position of where the photon arrives on the next mirror edge. Our construction presents a shortcut curve with a choice of diverging paths, with each diverging path affecting
Fig. 5. Simplified global layout of the target curve and its focal points in green, and the base curve consisting of red mirror edges and blue connector edges. A feasible shortcut curve is drawn in black, hippodrome in gray.

the relative offset along the later mirror edges differently. We place multiple edges, one for each item in a list \( S_i \) of the \( k \)-Table-SUM instance, at distances between \( \frac{1}{2} \) and 1 of the base curve. The shortcut curve has to choose between these edges and this choice encodes which element of \( S_i \) is taken.

These can be thought of as semi-transparent mirrors, as a shortcut curve can either end a shortcut on such a mirror edge or pass through it ending on a different mirror edge, akin to a photon either bouncing off of a semi-transparent mirror or traveling through it. Bouncing off of such a semi-transparent mirror corresponds to taking the corresponding item from a list in the \( k \)-Table-SUM instance. Since the distance from these edges to the target curve may be less than \( \frac{1}{2} \), it may happen that a feasible shortcut curve traverses the edge before taking the next shortcut. Therefore the relative position along an edge no longer encodes precise values but approximates the partial sums. We can introduce a scaling in the horizontal direction to contain this error. A second problem that occurs is that edges may overlap in the vertical direction, such that photons may visit multiple edges. We will fix this by stretching the instance even further.

4.2 Construction

In this section we describe the construction of the curves \( T \) and \( B \) given a \( k \)-Table-SUM instance.

We first describe the overall layout of the instance. The instance consists of \( k + 2 \) basic blocks, which we will call gadgets. It will consist of an initialization gadget \( g_0 \), \( k \) encoding-gadgets \( g_1, \ldots, g_k \) that encode the individual lists of the \( k \)-Table-SUM instance and a terminal gadget \( g_{k+1} \) used to verify that the relative offset encoded by a feasible shortcut curve is the same as the target value \( \sigma \). Each gadget \( g_i \) will consist of two curves \( T_i \) and \( B_i \), which we concatenate to get \( T \) and \( B \) in the end. We denote by \( H_y \) the horizontal line at \( y \) in \( \mathbb{R}^2 \) and by \( H_{\geq y} \) all points above \( H_y \). Similarly for \( \leq \), \( < \), and \( > \). And finally \( H_{\geq a}^\perp = H_{\geq a} \cap H_{< b} \). The target curve \( T \) will lie in \( H_0 \).

The base curve will have leftwards horizontal edges in \( H_{\geq 1/2}^\perp \) and \( H_{\leq -1/2}^\perp \), we will call mirror edges. All other edges of \( B_i \) that connect these mirror edges we will call connector edges. The connector edges will mostly lie outside of the hippodrome. The placement for connector edges that lie outside of the hippodrome is irrelevant. We have to carefully look at any exception, since we want any feasible shortcut curve to only interact with the mirror edges. Since all points lie on a small set of horizontal lines, we will occasionally denote the \( x \)-coordinate of a point and the point itself with the same variable but in different fonts. For example the point \( x_j \) has \( x \)-coordinate \( x'_j \).

The edges of the target curves \( T_i \) will, with the exception of twists, be oriented in positive \( x \)-direction. A twist centred at the focal point \((p, 0)\) is a subcurve defined by the points \((p - 1, 0), (p + 1, 0)\) and \((p - 1, 0)\) connected by straight lines. Around each focal point we introduce a buffer rectangle of length \( 2e = 5 \) and height \( 3 \), where we let \( e \) be a global constant for the construction. The base curve will never intersect these buffer zones, which is important for the twists to restrict the feasible shortcut curves as intended.

The instance will have two more global parameters. The first parameter \( \gamma \geq 1 \) is a global scaling factor in \( y \)-direction, which ensures that feasible shortcut curves will never enter connector edges. Furthermore it will
ensure that the approximate encoding of two different partial sums will stay disjoint. The parameter \( y \) will be in \( O(k) \). Lastly \( \beta \) is a spacing parameter ensuring that edges are far enough apart from one another.

Before we look at the precise construction, let us convince ourselves of the correctness of twists. For the following paragraph refer to Figure 6. Assume we have two mirror edges of length \( \lambda \), one placed from \((0, 1)\) to \((\lambda, 1)\), the other from \((2\lambda + 2\epsilon, -1)\) to \((\lambda + 2\epsilon, -1)\), which are connected by connector edges. We have a twist centred at \((\lambda + \epsilon, 0)\) on an otherwise rightwards facing target curve. Assume furthermore that we have a partial feasible shortcut curve, which reaches some point \((\lambda, 1)\) on the first mirror edge. Since the distance to the target curve is precisely \(1\), any reparametrization with a distance at most \(1\) for the shortcut Fréchet distance has to pair the point \((\lambda, 1)\) to \((\lambda, 0)\). Since the target curve is oriented in the opposite direction to the mirror edge, the only way to continue the feasible shortcut curve is by a shortcut to the right. It cannot jump to any point on the first mirror edge, since all those points lie left of \((\lambda, 1)\). The shortcut has to traverse the buffer zone of the twist. And since there are no edges of the base curve in the buffer zone, the shortcut has to traverse it completely. To analyse all shortcuts at a distance of at most \(1\), we place two auxiliary disks centred at \((\lambda + \epsilon \pm 1, 0)\) of radius \(1\). Any feasible shortcut curve traversing the buffer zone must traverse both of these disks, since otherwise no reparametrization can pair to the points \((\lambda + \epsilon \pm 1, 0)\) at distance at most \(1\), which are part of the target curve.

Since the twist first goes to \((\lambda + \epsilon + 1, 0)\) and then to \((\lambda + \epsilon - 1, 0)\), any feasible shortcut curve must also traverse the disks in this order. The first disk lies to the right of the second disk, and we try to traverse these disks from the left. The only possible way to traverse them with a straight line is through the intersection of the disks. And the only point in the intersection is exactly the focal point. So any shortcut of a feasible shortcut curve that traverses the buffer zone of a twist must traverse its focal point. A possible partial traversal is given in the upper plot in Figure 6. Note that it is in \(t-x\)-space, corresponding to how the two points paired by the reparametrizations traverse the curves in the \(x\)-direction.

\[\text{Fig. 6. Traversal of a shortcut through a twist. The target curve is distorted, to emphasize the structure of the twist.}\]
4.2.1 Initialization gadget $g_0$. For the construction refer to Figure 7. Both curves $T_0$ and $B_0$ 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 $a_0^1 = (3\gamma + 2\varepsilon, -1)$ to $b_0^1 = (\gamma + 2\varepsilon, -1)$.

4.2.2 Encoding gadget $g_j$. 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 $i$th table $S_i = \{s_i^1, \ldots, s_i^n\}$ of the $k$-Table-SUM instance. The construction of the precise values is given in Table 1. As for the parameters, $\lambda$ is the length of the entry edge, determined by the previous gadget $g_{i-1}$, and $\beta$ is the global spacing parameter. We will not give $\Delta_1^i$ and $\Delta_2^i$ explicitly but instead observe that we can choose them to be in $O(\text{poly}(n, \lambda^1, \beta, \varepsilon, \max S_i))$ resulting in a correct reduction. Excluding the entry edge the base curve $B_i$ consists of $2n + 2$ mirror edges and $O(n)$ connector edges. For $1 \leq j \leq n$ the first $n$ mirror edges $e_j^i$ are defined by $c_j^i$ and $d_j^i$, and the second $n$ mirror edges $e_j^i$ are defined by $c_j^i$ and $d_j^i$. The last two mirror edges are defined by $\bar{a}_i$ and $\bar{b}_i$, and $a_i^0$ and $b_i^0$. All of these mirror edges lie in either $H_{\leq \varepsilon/2}^{\leq 1}$ or $H_{\geq -\varepsilon/2}^{\leq -1}$ by construction. The target curve $T_i$ has four twists centred at $p_i^1, \ldots, p_i^4$. 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.

The intuition behind the construction is as follows: The first two steps place the first projection point at a distance from the entry edge, such that $n$ copies of the entry edge fit into the projection cone. By projection cone we denote the cone we get, when projecting the edge through a projection centre. The edges must satisfy further constraints, namely that all of the edges lie in $H_{\geq \varepsilon/2}$ and they have sufficient distance in $x$-direction.

These edges offer the choice, which item should be taken. Step 3 places an edge $e'$ from $\bar{a}$ to $\bar{b}$, where all the diverging paths have to meet, and then places $n$ copies of the entry edge in the $n$ disjunct projection cones such that their projections onto $e'$ have a relative offset according to the values in the list. Step 4 defines the entry...
Fig. 8. Construction of the encoding gadget. Mirror edges are red, connector edges blue and the target curve is green. Projection cones are black.

ACM Trans. Algor.
We now want to argue that this construction is correct. That is, there exists a feasible shortcut curve with
4.2.3 Terminal gadget

items in the list \( S_i \).

A shortcut curve traversing this gadget will look as follows. A shortcut curve reaches some point in the entry
eXchange edge to the next gadget, since we have to mirror the data once more to not introduce sign errors, due to every
central point 'flipping' the 'image' i.e. the values, like a pinhole camera would. The edges in Step 3 and 4 are used
to recombine the diverging paths making sure that the offset between the paths corresponds to the value of the

4.2.3 Terminal gadget \( g_{k+1} \). 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 \( T_{k+1} \) has a single

twist at \((b_k^+ + \lambda^k + \varepsilon, 0)\) and ends at \((b_k^+ + 2\lambda + 2\varepsilon - \gamma(\sigma + 1), 0)\). The base curve \( B_{k+1} \) 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 \( e_k^i \) has to start precisely at \( x \)-coordinate \( b_k^+ + \gamma(\sigma + 1) \) to hit the vertex.

4.3 Correctness

We now want to argue that this construction is correct. That is, there exists a feasible shortcut curve with \((4k + 2)\)
shortcuts if and only if the original \( k \)-Table-SUM instance has a solution. We begin by showing this for a subset of
shortcut curves we call \textit{one-touch}. For general shortcut curves this will be shown in Section 4.5. These one-touch
shortcut curves consist of only shortcuts and will never take subcurves of the base curve \( B \). In the following
section we often have to argue with distances that get preserved, when getting projected through a projection
point. This argument is captured in the following observation.

\textbf{Observation 25.} \textit{If an edge lies on some} \( H_\beta \) \textit{with length} \( \lambda \), \textit{and some point} \( p \) \textit{on} \( H_\alpha \) \textit{is given, we can then project
the edge through} \( p \) \textit{onto some} \( H_\alpha \) \textit{of length} \( \lambda' \). \textit{This forms two congruent triangles such that} \( \lambda' = \frac{\lambda_1}{p} \). \textit{Refer to} \( e_i^{\lambda-1} \)
and \( e_3 \) in Figure 8 as an example.
Together with Lemma 27 we have

This follows immediately from following the projections: the following argument to Figure 10 and Observation 25. For all choices of

Proof. We prove this via induction. For

Definition 26 (One-touch encoding). Let \( I = i_1, \ldots, i_k \) be an index set of a \( k \)-Table-SUM instance. 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 \( \mathbf{v}_0^0 \), then we project it through the first projection center \( p_0 \) onto \( e_0^0 \) to obtain \( \mathbf{v}_0^0 \). Now for \( 1 \leq i \leq k \) we project \( \mathbf{v}_i^{i-1} \) through \( p_i \) to land on \( e_i^0 \) to obtain \( \mathbf{v}_i^i \). We continue by projecting \( \mathbf{v}_i^i \) through \( p_{i+1} + 1 \) onto \( B_I \) to obtain \( \mathbf{v}_i^{i+1}_j \) for \( 1 \leq i \leq 3 \) (refer to Figure 10). Since these projections are all

Finally, we choose \( B(1) \) as the last vertex of our shortcut curve.

Lemma 27. For any \( 1 \leq i \leq k \) and \( 1 \leq j \leq n \) let \( x_j \) be the leftmost point on \( e_j \) reachable by projections starting on edge \( e_j \). Then \( x_j - b_j = y s_j \).

Proof. This follows directly from the construction (refer to Figure 10 and Table 1) and repeated application of Observation 25. \( x_j \) is determined by the projection of \( c_j \) through \( p_2 \), which is \( d_j \). Projecting this through \( p_3 \) lands on \( \overline{a}_j \) which by another projection through \( p_4 \) lands on \( x_j \). The offset between \( \overline{a}_j \) and \( \overline{a} \) is precisely the offset between \( x_j \) and \( b_j \). And this offset is by construction \( y s_j \).

Lemma 28. Given a shortcut curve \( B_I \), which is a one-touch encoding, let \( \mathbf{v}_i \) be the vertex of \( B_I \) on the entry-edges \( e_i \) of gadgets \( g_i \) for all applicable \( i \). Then \( ||\mathbf{v}_i - b_i|| = y(\sigma_i + 1) \), where \( \sigma_i \) is the \( i \)th partial sum of the index set \( I \) encoded by \( B_I \).

Proof. We prove this via induction. For \( i = 0 \) this is correct by construction of the initialization gadget. Refer for the following argument to Figure 10 and Observation 25. For all choices of \( j \) we have \( ||\mathbf{v}_{i-1} - b_{i-1}|| = ||\mathbf{v}_i - x_j|| \).

This follows immediately from following the projections:

Together with Lemma 27 we have

\[
||\mathbf{v}_i - b_i|| = ||\mathbf{v}_i - x_j|| + ||x_j - b_i|| = y(\sigma_i + 1) + y s_{i,j} = y(\sigma_i + 1).
\]
Lemma 29. Let \( a_j = (j-1)(\lambda + 1) + \beta \). Then the points \( c_j', d_j', c_j'' \) and \( d_j'' \) are given by

\[
\begin{align*}
c_j' &= p_1' + \frac{(\Delta_1' - \epsilon)\Delta_2'}{\Delta_1' + o_j'}, \\
d_j' &= p_1' + \frac{(\Delta_1' - \epsilon - \lambda)\Delta_2'}{\Delta_1' + o_j'}, \\
c_j'' &= p_2' + \frac{(\epsilon + o_j')\Delta_2'}{\Delta_2' - \gamma s_j + o_j'}, \\
d_j'' &= p_2' + \frac{(\epsilon + o_j')\Delta_2'}{\Delta_2' - \gamma s_j + o_j'}.
\end{align*}
\]

Proof. We will only show this for \( d_j' \), as the computations for the other points are very similar. We translate the instance such that \( p_1' \) coincides with the origin. Then \( d_j' \) is defined as the intersection of a line \( l_1 \) from \((0, 0)\) to \((\Delta_1' - \epsilon - \lambda + 1, 1)\) and \( l_2 \) from \((\Delta_1' - \epsilon - \lambda + 1 - (j-1)(\lambda + 1) + \beta, 1)\) to \((\Delta_1', 0)\). Thus the \( x \)-coordinate of the intersection point satisfies

\[
d_j' = \frac{\Delta_1' - \epsilon - \lambda}{\Delta_1' + (j-1)\lambda + 1 + (j-1)\beta} - 1 - \frac{\Delta_1' - \epsilon - j\lambda + (j-1)\beta}{\Delta_1' + (j-1)\lambda + 1 + (j-1)\beta}
\]

and hence

\[
d_j' = \frac{(\Delta_1' - \epsilon - \lambda)\Delta_2'}{\Delta_1' + (j-1)\lambda + 1 + (j-1)\beta}.
\]

Since \( l_1 \) has slope \((\Delta_1' - \epsilon - \lambda + 1)^{-1}\) we have

\[
d_j' = (\Delta_1' + (j-1)(\lambda + 1) + \beta)^{-1}((\Delta_1' - \epsilon - \lambda + 1)\Delta_1', \Delta_2').
\]

Lemma 30. Assume \( \Delta_1' \) and \( \Delta_2' \) are given such that \( \Delta_1' \geq 3\epsilon + \lambda + 1 \) and \( \Delta_2' \geq 2\epsilon + \gamma s_j + (j-1)(\lambda + 1) + \beta + 2\lambda + 1 \) holds for every \( 1 \leq i \leq k \) and \( 1 \leq j \leq n \). Then the constructed base curve never enters any buffer zone centred at a projection centre.

ACM Trans. Algor.
Proof. We will not explicitly discuss the connector edges outside the hippodrome, since they can easily be placed such that they do not enter buffer zones. We first consider the encoding gadget \( g_1 \). In this proof we omit the top index \( i - 1 \) from \( \lambda^{i-1} \) and top index \( i \) from all other variables as we only look at a single gadget at a time. For the buffer zones centred at \( p_2 \) and \( p_4 \) for \( 1 \leq i \leq k \) the claim is implied by construction.

For the buffer zone centred at \( p_1 \) the closest edge in \( x \)-direction inside the hippodrome is by construction \( e_n \). And of this edge \( d_n \) is the closest point. Lemma 29 and the fact that \( \Delta_1 \geq 2\epsilon + \lambda + \epsilon \) holds imply that \( d_n \geq p_1 + \frac{\Delta_1 - \lambda - \epsilon}{2} > p_1 + \epsilon \) holds. This implies the claim for this projection centre as well. For the buffer zone centred at \( p_3 \) the closest edge in \( x \)-direction inside the hippodrome is by construction \( e'_n \). And of this edge \( e'_n \) is the closest point. Lemma 29 and the fact that \( \Delta_2 \geq 2\epsilon + s_n + \alpha_j + 2\lambda \) holds imply that we get \( e'_n \geq \frac{\Delta_2}{2} \) and thus \( p_3 \geq e'_n + \Delta_2/2 \geq c'_n + \epsilon \). The last two buffer zones left to analyse are around the projection centre in the initialization and end gadget, however the claim follows directly from construction.

\[ \square \]

Lemma 31 (4-monotonicity [Buchin et al. 2014]). Any feasible shortcut curve is rightwards 4-monotone. That is if \( x_1 \) and \( x_2 \) are the \( x \)-coordinates of two points that appear on the shortcut curve in that order, then \( x_2 + 4 \geq x_1 \). Furthermore it lies inside or on the boundary of the hippodrome.

Proof. Any point on the feasible shortcut curve has to lie within distance 1 to some point of the target curve, thus the curve cannot leave the hippodrome. As for the monotonicity, assume for the sake of contradiction that there exist two such points with \( x_2 + 4 < x_1 \). Let \( \tilde{x}_1 \) be the \( x \)-coordinate of the point on the target curve matched to \( x_1 \) and let \( \tilde{x}_2 \) be the one for \( x_2 \). By the Fréchet matching it follows that \( \tilde{x}_2 - 1 + 4 < \tilde{x}_1 + 1 \). This would imply that the target curve is not 2-monotone which contradicts the way we constructed it.

\[ \square \]

Lemma 32. For every \( \lambda > 0 \), \( \beta \geq 5 \), \( \epsilon > 0 \) and integer \( n > 0 \) there are values \( \Delta_1^i, \Delta_2^i \in \Theta(\text{poly}(\epsilon, \lambda, (\beta - 4)^{-1}, \beta, \gamma \max S_j, n)) \) for every \( 1 \leq i \leq n \) such that any two mirror edges of the gadget \( g_i \) are at least 4 apart and all mirror edges lie inside the hippodrome.

Proof. We first consider the encoding gadget \( g_1 \). For this we omit the top index \( i - 1 \) from \( \lambda^{i-1} \) and top index \( i \) from all other variables as we only look at a single gadget at a time. Recall from Lemma 29

\[ d_j = \frac{1}{\Delta_1 + (j - 1)(\lambda + \beta)} \left( (\Delta_1 - \epsilon - \lambda)\Delta_1, \Delta_1 \right) \]

and

\[ e_j = \frac{1}{\Delta_1 + (j - 1)(\lambda + \beta)} \left( (\Delta_1 - \epsilon)\Delta_1, \Delta_1 \right). \]

Hence for any \( \Delta_1 \geq (n - 1)(\lambda + \beta) \) the \( y \)-coordinate of any such edge lies in \([1/2, 1]\). Next we show that \( e_{j+1} + 4 < d_j \) holds, implying that the edges \( \{e_1, \ldots, e_n\} \) have a pairwise distance of at least 4. The expression \( e_{j+1} + 4 < d_j \) is equivalent to

\[ \lambda\Delta_1 j(\lambda + \beta) + 4(\Delta_1 + j(\lambda + \beta))\left( (\Delta_1 + (j - 1)(\lambda + \beta)) + (\lambda + \beta)\Delta_1 \epsilon < \beta\Delta_1^2. \]

As both sides are second degree polynomials in \( \Delta_1 \) and the second order coefficient on the left hand side is 4 whereas on the right hand side it is \( \beta \) (recall \( \beta \geq 5 \) there is a value \( \delta_1 \in O(\text{poly}(\lambda, (\beta - 4)^{-1}, \beta, \epsilon, n)) \) such that for every \( \Delta_1 > \delta_1 \) the expression \( e_{j+1} + 4 < d_j \) holds for all \( 1 \leq j < n \).

Let us next look at the edges \( e'_j \). Similarly recall from Lemma 29 that

\[ c'_j = \frac{1}{\Delta_2 - y s_j + (j - 1)(\lambda + \beta)} \left( (\epsilon + j\lambda + (j - 1)\beta)\Delta_2, -\Delta_2 \right) \]

and

\[ d'_j = \frac{1}{\Delta_2 - y s_j + (j - 1)(\lambda + \beta)} \left( (\epsilon + (j - 1)\lambda + (j - 1)\beta)\Delta_2, -\Delta_2 \right). \]
Hence there is a $\delta_2 \in O(\operatorname{poly}(\max_i s_i, n, \lambda, 4))$ such that for any $\Delta_2 > \delta_2$ the $y$-coordinate of any such edge lies in $[-1, -1/2]$. Lastly the expression $e_j + 4 < d_{j+1}$ is equivalent to
\[
\left(e + j \lambda + (j-1)\beta \right) (y(s_j - s_{j+1}) + \lambda + \beta) \Delta_2 + 4 \left( \Delta_2 - y s_j + (j-1)(\lambda + \beta) \right) < \beta \Delta_2 \left( \Delta_2 - y s_j + (j-1)(\lambda + \beta) \right).
\]
We again have two second order polynomials in $\Delta_2$ on both sides, with the second order coefficient on the left hand side being 4 and on the right hand side being $\beta \geq 5$. Hence there is a $\delta'_2 \in O(\operatorname{poly}(\max_i s_j, n, \lambda, (\beta - 4)^{-1}, \beta, 4))$ such that all $\Delta_2 > \delta'_2$ satisfy this equation. Thus we can chose $\Delta_1, \Delta_2 \in \Theta(\operatorname{poly}(\epsilon, \lambda, (\beta - 4)^{-1}, \beta, \max_i s_j, n))$ such that $\Delta_1 > \delta_1$ and $\Delta_2 > \max(\delta_2, \delta'_2)$. Thus all edges $\{e_1, \ldots, e_n\}$ have a pairwise distance of at least 4, and all edges $\{e'_1, \ldots, e'_n\}$ have a pairwise distance of at least 4. Further they all lie within the hippodrome and lie at in either $H_{\leq \frac{1}{2}}$ or $H_{\leq \frac{1}{2}}$. As all other mirror edges are separated by buffer zones and their position is trivially inside the hippodrome and at distance at least 1/2 to $B$, this concludes the proof.

\section*{Observation 33.}
If $\beta \geq 5$ and $\epsilon > 2$ holds, we can choose $\Delta_i$ and $\Delta'_i$ in $\Theta(\operatorname{poly}(\epsilon, \lambda^{-1}, (\beta - 4)^{-1}, \beta, \max_i s_j, n))$ for all $1 \leq i \leq k$ such that the conditions of both Lemma 30 and Lemma 32 hold.

\section*{Lemma 34.}
If $\epsilon > 2$ then a feasible shortcut curve passes through every buffer zone of the target curve via its projection centre and furthermore it does so from left to right.

\section*{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. By Lemma 30 the base curve 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 $2\epsilon > 4$, then the only way 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. The intersection lies at the centre of the buffer zone, as we saw in the beginning.

\section*{Lemma 35.}
If $\epsilon > 2$, $\beta \geq 5$ and all $\Delta_i$ and $\Delta'_i$ are chosen according to Observation 33 then a feasible shortcut curve that is one-touch visits exactly one of the edges $e_i$, and exactly one of the edges $e'_i$ for $1 \leq j \leq n$ in every gadget $g_i$ for $1 \leq i \leq k$. Furthermore it visits all edges $e_i$, for $0 \leq i \leq k$.

\section*{Proof.}
By Lemma 31 any feasible shortcut curve is 4-monotone. Furthermore it starts at $B(0)$ and ends at $B(1)$. By Lemma 34 it goes through all projection centres of the target curve from left to right. We first want to argue that it visits at least one mirror edge between two projection centres i.e. that it cannot 'skip' such a mirror edge by matching to two twists in one shortcut. Such a shortcut would have to lie on $H_0$ since it has to go through the two corresponding projection centres lying on $H_0$. By construction the only possible endpoints of such a shortcut lie on the connector edges that connect to mirror edges. Assume such a shortcut could be taken by a shortcut curve starting from $B(0)$. Then there must be a connector edge which intersects a line from a point on a mirror edge through the projection centre. In particular since the curve has to go through all projection centres, one or more of the following must be true for some $1 \leq i \leq k$:
- there exists a line through $p_{i-1}$ intersecting a mirror edge $e_i$ and a connector edge of $e'_j$,
- there exists a line through $p_i$ intersecting a mirror edge $e'_i$ and a connector edge of $e'_j$ for some $l$, or
- there exists a line through $p'_j$ intersecting a mirror edge $e'_i$ and a connector edge of $e'_i$ for some $l > j$.  
However this was prevented by the careful placement of these connector edges.

It remains to prove that for each $1 \leq i \leq k$ the shortcut curve cannot visit more than one $e'_i$ and cannot visit more than one $e'_i$ and therefore visits exactly one mirror edge between two projection centres. The shortcut curve has to lie inside or on the boundary of the hippodrome and is 4-monotone (Lemma 31). At the same time, we constructed the gadget such that the mirror edges between two consecutive projection centres have distance \(
\pi_{\Delta_i}\).
at least 4 to one another by Lemma 32 choosing \( \Lambda_1 \) and \( \Lambda_2 \) in the process. Furthermore inside the projection cone from \( e_i \) to \( p_i \) all mirror edges come before (as parametrized by the base curve) \( e_i^j \), implying the claim. \( \square \)

**Corollary 36.** A feasible shortcut curve consists of exactly \( 4k + 2 \) shortcuts. One shortcut for the initialization and end gadgets and 4 shortcuts in each encoding gadget.

Putting the above lemmas together implies the correctness of the reduction for shortcut curves that are one-touch i.e. which visit every edge in at most one point.

**Lemma 37.** If \( \epsilon > 2 \) and \( \beta \geq 5 \) and all \( \Lambda_1 \) and \( \Lambda_2 \) are chosen according to Observation 33, then for any feasible one-touch shortcut curve \( B \), it holds that the index set \( I \) encoded by \( B \) sums to \( \sigma \). Furthermore for any index set \( I \) that solves the \( k\)-Table-SUM instance there is a feasible one-touch shortcut curve that encodes it.

**Proof.** Lemma 34 and Lemma 35 imply that \( B \) must be a one-touch encoding as defined in Definition 26. By Lemma 28 the second last vertex of \( B \) is the point on the edge \( e_i \) which is at distance \( \gamma(\sigma + 1) \) to \( b_i \) where \( \sigma \) is the sum encoded by the subset selected by \( B \). The last vertex of \( B \) is equal to \( B(1) \), which we placed in distance \( \gamma(\sigma + 1) \) to the projection of \( b_i \) through \( p_i^{k+1} \). Thus the last shortcut of \( B \) passes through the last projection centre of the target curve if and only if \( \sigma = \sigma \). It follows that if \( \sigma \neq \sigma \), then \( B \) cannot be feasible. For the second part of the claim we construct a one-touch encoding as defined in Definition 26. By the above analysis it will be feasible if the subset sums to \( \sigma \), since the curve visits every edge of \( B \) in at most one point and in between uses shortcuts which pass through every buffer zone from left to right and via the buffer zones projection centre. \( \square \)

### 4.4 Size of coordinates

**Lemma 38.** The curves can be constructed in \( O(kn) \) time. Furthermore if we choose \( \epsilon = \frac{5}{2} \) and \( \beta = 5 \) and \( \Lambda_1 \) and \( \Lambda_2 \) according to Observation 33, then the coordinates used are in \( O(\text{poly}(k,n,\gamma \sum_i \max S_i)) \).

**Proof.** From the construction we know that \( p_i^j - p_i^{j-1} = 2\Lambda_1 + 2\Lambda_2 \). Further by Observation 33 we know that we can choose \( \Lambda_1 \) and \( \Lambda_2 \) to be in \( \Theta(\text{poly}(\epsilon, n, \lambda^{i-1}, \gamma \max S_i, (\beta - 4)^{-1}, \beta)) = \Theta(\text{poly}(n, \lambda^{i-1}, \gamma \max S_i)) \) as \( \epsilon = \frac{5}{2} \) and \( \beta = 5 \).

The length of the entire instance is given by the combined lengths of all of the gadgets. Thus the length is given by

\[
\epsilon + \gamma + \sum_{i=1}^{n} (2\Lambda_1^i + 2\Lambda_2^i) + \epsilon + \lambda^n + (\epsilon + \lambda^n - \gamma(\sigma + 1)).
\]

This is in \( O(\text{poly}(k,n,\gamma \sum_i \max S_i)) \), as \( \lambda^0 = 2\gamma \) and \( \lambda^i = \lambda^{i-1} + \gamma \max S_i \).

As for the complexity, each of the constructed gadgets uses \( O(n) \) vertices, since we need to place \( O(n) \) mirror and connector edges. Because we construct \( k + 2 \) gadgets, the overall number of vertices used is in \( O(kn) \). The curves \( T \) and \( B \) can be constructed using a single iteration from left to right, therefore the overall construction takes \( O(kn) \) time. \( \square \)

### 4.5 Correctness for general shortcut curves

When we consider general feasible shortcut curves that might not necessarily be one-touch, they might follow a mirror edge for a short while instead of immediately taking the next shortcut. This results in a small error when comparing the shortcut curve with a one-touch curve encoding the same index set. We now want to contain this incremental error with the control parameter \( \gamma \).

**Lemma 39.** Choose \( \epsilon > 2 \) and \( \beta \geq 5 \) and all \( \Lambda_1 \) and \( \Lambda_2 \) according to Lemma 32. Given a feasible shortcut curve \( B_0 \), let \( u_i \) be any point of \( B_0 \) on the exit-edge \( e_i \) of the gadget \( g_i \). For all \( i \) let \( \sigma_i \) be the partial sum encoded by \( B_0 \). If we choose \( \gamma > \frac{\epsilon}{\beta} \), then
Note that
\[
b_i^* + \gamma(\sigma_1 + 1) - \xi_i \leq \nu_i \leq b_i^* + \gamma(\sigma_1 + 1) + \xi_i
\]
holds where \(\xi_i = 16i + 5\) is an upper bound of the maximum error possible for any shortcut curve traversing up to gadget \(g_i\).

**Proof.** We prove this claim by induction on \(i\). For \(i = 0\) the claim follows by construction of the initialization gadget: As \(B_0\) has to start at \(B(0)\) and it has to lie completely in the hippodrome it has to take a shortcut, and since it has to pass a twist it must traverse its projection centre. The only point where this shortcut can end is on the entry edge of \(g_1\). By construction this point is at a distance of \(\gamma\) from \(b_0^*\). Since the edge is oriented leftward \(B_0\) can only walk in that direction. However \(B_0\) is rightwards \(4\)-monotone. It follows that
\[
b_0^* + \gamma - 4 \leq \nu_1 \leq b_0^* + \gamma.
\]
Since \(\xi_0 = 5 > 4\) and \(\sigma_0 = 0\) this implies the claim for \(i = 0\).

For \(i > 0\) the curve \(B_i\) entering gadget \(g_i\) from edge \(e_i^{* - 1}\) has to pass the first twist and has to do so through the projection point. By induction
\[
b_i^{* - 1} + \gamma(\sigma_i - 1 + 1) - \xi_{i - 1} \leq \nu_{i - 1} \leq b_i^{* - 1} + \gamma(\sigma_i - 1 + 1) + \xi_{i - 1}.
\]
Since \(\gamma > \xi_{i - 1} = \xi_{i - 1} + 16\) it follows that the distance of \(\nu_{i - 1}\) to the endpoints of the edge is
\[
\gamma(\sigma_i - 1 + 1) - \xi_{i - 1} \geq \gamma - \xi_{i - 1} > 16
\]
and
\[
\gamma(\sigma_i - 1 + 1) + \xi_{i - 1} \leq (\lambda^{i - 1} - \gamma) + \xi_{i - 1} \leq \lambda^{i - 1} - 16,
\]
thus \(\nu_{i - 1}\) lies at a distance greater than 4 from the endpoints of the entry-edge of gadget \(g_i\).

Therefore the only edges that can be hit through the projection point \(p_i^1\) are \(e_j\). Denote by \(\sigma_{\max} = \gamma(\sigma_{i - 1} + 1) + \xi_{i - 1}\) and \(\sigma_{\min} = \gamma(\sigma_{i - 1} + 1) - \xi_{i - 1}\) the maximal and minimal offset \(\nu_{i - 1}\) may have from \(b_i^{* - 1}\). Furthermore let \(\alpha_j\) be the \(y\)-coordinate of the edge \(e_j\), and similarly \(\alpha_j^*\) for the edge \(e_j^\prime\). We will again omit the top index of \(i\) since it is fixed for the gadget \(g_i\) from now on. Then the interval of \(x\)-coordinates where the shortcut may end on \(e_j\) is
\[
[c_j - \alpha_j \sigma_{\max}, c_j - \alpha_j \sigma_{\min}].
\]
The length of the edge \(e_j\) is \(\alpha_j \lambda^{i - 1}\). Thus the endpoint lies inside the edge. Now \(B_i\) can walk on this edge as well. Again it can do so only leftwards. As the curve is rightwards \(4\)-monotone it may do so a distance of at most 4. But since \(\alpha_j \geq \frac{1}{2}\) and we already saw that \(\sigma_{\max} > \lambda + 16\) the shortcut curve cannot leave this edge by walking. Thus all possible points for \(B_i\) are determined by the interval
\[
[c_j - \alpha_j \sigma_{\max} - 4, c_j - \alpha_j \sigma_{\min}].
\]
Hence the shortcut curve must leave this edge via a shortcut through \(p_2\). It then may again walk up to 4 to the left resulting in the interval
\[
\left[d_j^* + \alpha_j' \sigma_{\min} - 4, d_j^* + \alpha_j' \left(\sigma_{\max} + \frac{4}{\alpha_j}\right)\right].
\]
Repeated application for the next two edges results in the interval for the edge \(e^\prime\)
\[
[a - y s_{i,j} - \left(\sigma_{\max} + \frac{4}{\alpha_j}\right) - 4, a - y s_{i,j} - \left(\sigma_{\min} - \frac{4}{\alpha_j'}\right)]
\]
Note that \(a_j = a - y s_{i,j}\) by construction. And for \(e_4\) it lands in the interval
\[
\left[b_4 + \sigma_{\min} + y s_{i,j} - \frac{4}{\alpha_j'} - 4, b_4 + \sigma_{\max} + y s_{i,j} + \frac{4}{\alpha_j} + 4\right]
\]
Since \( \alpha_j \geq \frac{1}{2} \) and \( \alpha'_j \geq \frac{1}{2} \) holds we get for the item \( s_{i,j} \) taken by the shortcut curve
\[
b^*_i + y(\sigma_i + 1) - \xi_i = b^*_i + y s_{i,j} + y(\sigma_{i-1} + 1) - \xi_{i-1} + 16 \leq v_i
\]
as well as
\[
v_i \leq b^*_i + y s_{i,j} + y(\sigma_{i-1} + 1) + \xi_{i-1} + 16 = b^*_i + y(\sigma_i + 1) + \xi_i,
\]
implying the claim.

\[ \Box \]

**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 \( n^{o(k)} \).

**Proof.** Let some \( k \)-Table-SUM instance be given. Let \( \epsilon = \frac{3}{2}, \beta = 5 \) and \( y = 16(k + 1) + 5 \). Choose \( \Delta_1 \) and \( \Delta_2 \) according to Lemma 32. Let \( B_s \) be any feasible shortcut curve of the constructed instance for the \( k \)-Table-SUM instance. Since \( B_s \) is feasible it must visit the exit edge of the last gadget \( q_t \) at distance \( y(\sigma + 1) \) to \( b^*_i \) since this is the only point that connects to \( B(1) \) via a shortcut. Let \( v_k = b^*_k + y(\sigma + 1) \) be the \( x \)-coordinate of this visiting point and let \( \sigma_k \) be the subset of the sum of the subset encoded by \( B_s \). Lemma 39 implies that
\[
b^*_k + y(\sigma_k + 1) - \xi_k \leq v_k = b^*_k + y(\sigma_k + 1) \leq b^*_k + y(\sigma_k + 1) + \xi_k,
\]
since \( y = 16(k + 1) + 5 > \xi_k \). Therefore,
\[
\sigma_k - \frac{\xi_k}{y} \leq \sigma \leq \sigma_k + \frac{\xi_k}{y}.
\]
Since \( y > \xi_k \) it follows that \( \sigma_k \) must be \( \sigma \) since both are integers. Hence any feasible shortcut curve solves the \( k \)-Table-SUM instance, implying the claim.

\[ \Box \]

5 APPROXIMATE DECISION ALGORITHMS

In light of the previous section we now describe a \((3 + \epsilon)\)-approximation algorithm for the decision problem of the \( k \)-shortcut Fréchet distance of two polygonal curves in the plane. The algorithm has a near-quadratic running time in \( n \). In Section 5.3 we show that the algorithm can be modified to have running time near-linear in \( n \), for the special class of \( c \)-packed curves.

5.1 Description of the 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 threshold \( \delta \) and approximation parameter \( \epsilon \). As before, the algorithm (see Algorithm 1) iterates over the cells of the free-space diagram and computes sets \( N_{ij}^s, V_{ij}^s \), and \( D_{ij} \) for each cell \( C_{i,j} \). The main difference now is 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 \( r_{i-1,j-1}^* \) in \( \bigcup_{a < b < d} P_{i,j}^a \), \( \epsilon \) and distance parameter \( 3\delta \). Crucially, the set resulting from one call to the procedure has constant complexity and is sufficient to approximate the set \( D_{ij} \). We then compute
\[
P_{i,j}^* = O( N_{ij}^* \cup D_{ij}^* \cup V_{ij}^s) \cap D_{\Delta(i,j)},
\]
similarly to Section 3. From this we compute (i) the leftmost point \( b_{ij}^* \) in \( \bigcup_{b \leq j} P_{i,b}^a \) based on \( P_{i,j}^* \) and \( P_{i,j-1}^* \), (ii) the rightmost point \( r_{i,j}^* \) in \( \bigcup_{a \leq i \leq b \leq j} P_{i}^a \) based on \( P_{i,j}^*, r_{i-1,j}^* \) and \( r_{i,j-1}^* \), and (iii) the outgoing reachability intervals of \( P_{i,j}^* \). 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.
Algorithm 1 Approximate Decider

1: procedure APPROXIMATEDECIDER’(curve T, curve B, δ > 0, 0 < ϵ ≤ 1)
2:   if ||T(0) − B(0)|| > δ or ||T(1) − B(1)|| > δ then Return 'd^k_S(T', B') > δ'
3:   Let T' be the data structure of Lemma 40 built on T with ε.
4:   Let g^s, g^1, A^s and A^1 be arrays of size n_1 for each 0 ≤ s ≤ k.
5:   for s = 0, ..., k do
6:     for j = 1, ..., n_2 do
7:       Copy array A^s into A^1
8:       for i = 1, ..., n_1 do
9:         Compute D_δ(i,j)
10:        if i = 1, j = 1 and s = 0 then
11:           P_s^i,j = D_δ(i,j)
12:      else
13:        // Compute set of points directly reachable from neighboring cells
14:          Let I_o = ∅ and I_h = ∅
15:           if j > 1 then Let I_o be the incoming reachability interval from A^1[i]
16:          if i > 1 then Let I_k be the incoming reachability interval from A^1[i − 1]
17:            Let N_{i,j} = (Q(I_o) ∪ Q(I_h)) ∩ D_δ(i,j)
18:          if s > 0 then
19:            // Approximate set of points reachable by diagonal tunnel
20:              Retrieve rightmost point r in the lower left quadrant of C_{i,j} from g^s_{i−1}.
21:              Let D^s_{i,j} = APXDIGONALTUNNEL(r, (i, j), ε, 3δ)
22:            // Compute set of points reachable by vertical tunnel
23:              Retrieve leftmost point l in the column below C_{i,j} from g^1_{i−1}.
24:              Let V^s_{i,j} = VERTICALTUNNEL(l, (i, j), δ)
25:          else
26:            Let D^s_{i,j} = ∅ and V^s_{i,j} = ∅
27:          // Putting things together
28:            P^s_{i,j} = Q(N_{i,j} ∪ D^s_{i,j} ∪ V^s_{i,j}) ∩ D_δ(i,j)
29:          if P^s_{i,j} ≠ ∅ then
30:            Store the rightmost point of P^s_{i,j} in g^s
31:            Store the leftmost point of P^s_{i,j} in g^1
32:            Compute outgoing reachability intervals and using P^s_{i,j} and store them in A^s[i].
33:        if (1, 1) ∈ A^1[n_1] then
34:          Return 'd^k_S(T', B') ≤ 3(1 + ε)^2 δ' with s ≤ k shortcuts
35:        else
36:          Return 'd^k_S(T', B') > δ' with at most k shortcuts
37:   procedure APPROXIMATEDECIDER(curve T, curve B, δ > 0, 0 < ϵ ≤ 1)
38:     Let ε' = ϵ/9
39:     Return APPROXIMATEDECIDER’(T, B, δ, ε')
Approximate Diagonal Tunnel

**Algorithm 2** Approximate Diagonal Tunnel

```plaintext
1: procedure apxDiagonalTunnel((r_T, r_B), (i, j), ε, δ) 
2:     Let r = B(r_B) 
3:     //r is the starting point of the shortcut 
4:     for t ∈ (G_{ε/3} ∩ b_{(1+ε)δ}((a_i))) do 
5:         Query F_t for the distance d_{F_t}(T[r_T, a_i]) and store the answer in δ'
6:         if δ' ≤ (1 + ε)^2δ then 
7:             Mark t as eligible 
8:         // t is an eligible endpoint of a shortcut 
9:     Compute the convex hull H of eligible points 
10:     if r ∈ H then 
11:         Return C = D^{(i,j)}_δ 
12:     else 
13:         Let U be the cone with apex r formed by tangents t_1 and t_2 from r to H 
14:         Let p_1 ∈ H be a supporting point of the tangent t_i for i ∈ {1, 2} 
15:         Let L be the subchain of δH with endpoints p_1 and p_2 which is facing r 
16:         Let H' ⊂ U be the set bounded by L and the rays supported by t_1 and t_2 facing away from r 
17:         Let C' be the intersection of H' with e_j 
18:     Return C = (e_j × C') ∩ D^{(i,j)}_δ 
```

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 40** (Distance oracle [Driemel and Har-Peled 2012]). Given a polygonal curve Z with n vertices in \( \mathbb{R}^d \) and \( \epsilon > 0 \), one can build a data structure \( F_ε \) in \( O(\chi^2 n \log^2 n) \) time, that uses \( O(n \chi^2) \) space such that given a query segment \( \overline{pq} \) and any two points \( u \) and \( v \) on the curve, one can \((1 + \epsilon)\)-approximate \( d_{F_ε}(\overline{pq}, Z[u, v]) \) in \( O(\epsilon^{-2} \log n \log \log n) \) time, where \( \chi = \epsilon^{-d} \log(\epsilon^{-1}) \).

**Definition 41** (Grid). We define the scaled integer grid \( G_δ = \{(δx, δy) \mid (x, y) ∈ \mathbb{Z}^2\} \).

Approximate diagonal tunnel procedure. The procedure (see Algorithm 2) is provided with parameters \( \epsilon, \delta \), 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 \( b_δ(a_{i+1}), \ldots, b_δ(a_i) \), and is contained in the set of all stabbers through disks of radius \((1 + \epsilon)^2\delta\) centered at the same vertices. We approximate this set of stabbers as follows.

We iterate over all grid points \( t \) in the disk \( G_{ε/3} ∩ b_{(1+ε)δ}((a_i)) \), and make queries to the precomputed distance oracle \( F_ε \) to determine if the Fréchet distance of the query segment \( T[r_T] \) 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 \((1 + \epsilon)^2\delta \). 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 11 illustrates this. We then intersect \( H' \) with the edge \( e_j \) resulting in a single horizontal slab in \( C_{i,j} \). This resulting set is then intersected with \( D^{(i,j)}_δ \) and returned.
We now analyse the described algorithm, namely the \textsc{ApproximateDecider} procedure.

\subsubsection{Correctness}
We argue that the structure of $P_{i,j}$ as approximated by the \textsc{ApproximateDecider} procedure is indeed as claimed. Namely for all $i$, $j$ and $s$ it holds that $R_{1,s}^{(i,j)} \subset \bigcup_{0 \leq s' \leq s} P_{1,s'}^{(i,j)} \subseteq R_{3(s+1)}^{(i,j)}$. 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 $3\delta$. To prove correctness, we use the following lemma by Driemel and Har-Peled. The lemma states that if a feasible tunnel $\tau(r, q)$ costs more than $3\delta$ then any feasible tunnel $\tilde{\tau}(p, q)$ with $x_p \leq x_r$ costs more than $\delta$.

\textbf{Lemma 42} (monotonicity of tunnels [Driemel and Har-Peled 2012]). 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 $B_1$ and $B_2$ such that $d_F(T_2, B_1) \leq \delta$ and the start and end point of $T_2$ is within distance $\delta$ to the start and end point of $B_2$ respectively, then $d_F(T_2, B_2) \leq 3\delta$.

In the following, we denote with $\{b_0(v_1), \ldots, b_0(v_m)\}$ a sequence of disks $\{b_0(v_1), \ldots, b_0(v_m)\}$ for some $m$.

\textbf{Lemma 43.} Let $a, b_1, b_2 \in \mathbb{R}^d$ together with a sequence of vertices $v_1, \ldots, v_n$ be given. If $\tilde{a}b_1$ stabs through disks $\{b_0(v_1), \ldots, b_0(v_m)\}$, and $ab_2$ stabs through $\{b_0(v_1), \ldots, b_0(v_m)\}$, then for any $t \in [0, 1]$ the line segment $\tilde{a}b(t)$ stabs through $\{b_0(v_1), \ldots, b_0(v_m)\}$ where $b(t) = (1-t)b_1 + tb_2$.

\textbf{Proof.} Refer to Figure 12. Consider the 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 $\tilde{a}b(t)$ 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 $\tilde{a}b_1$ and $q_i$ of $\tilde{a}b_2$, $p_i$ lies on the one and $q_i$ on the other side of $\tilde{a}b(t)$. Since $b_0(v_i)$ is convex and $p_i$ and $q_i$ are inside this disk, the intersection of $p_i q_i$ and $\tilde{a}b(t)$ is inside the disk as well. Call this intersection point $r_i$. The set $\{r_i\}$ are realising points for $\tilde{a}b(t)$. This follows directly from the fact that $\{p_i\}$ and $\{q_i\}$ are ordered along their respective line segments, and thus $p_i q_i$ never crosses another $\tilde{p}_j q_j$. Thus for $i < j$, $r_i$ appears before $r_j$ along $\tilde{a}b(t)$, implying the claim. \hfill $\Diamond$
Additionally, $ACM$ Trans. Algor.

**Lemma 44.** 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 $a_1 \overline{b_1}$ stabs through $\{b_\delta(v_i)\}_i$, and $||a_1 - b_1|| \leq \delta'$ and $||a_2 - b_2|| \leq \delta'$, then $a_1 \overline{b_1}$ stabs through $\{b_{\delta + \delta'}(v_i)\}_i$.

**Proof.** By Observation 14, $d_F(a_1, \overline{b_1}, a_2, b_2) \leq \delta'$, via the reparametrization $(f, g)$ with $f(t) = (1-t)a_1 + ta_2$ and similarly $g(t) = (1-t)b_1 + tb_2$. As $p = a_1 \overline{b_1}$ stabs through $\{b_\delta(v_i)\}_i$, there exist realizing points $p_i$ along $p$, with $p_i$ lying in the $\delta$-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 $q = a_2 b_2$, proving the claim. \hfill $\square$

**Lemma 45.** Given $r \in \mathbb{R}^2, C_{i,j}, \varepsilon$ and $\delta$ like in the apxDiagonalTunnel procedure. Denote by $S_\delta$ the set of endpoints of all $\delta$-stabbers (that is, stabbers through $b_\delta(v_m)$ for $a + 1 \leq m \leq i$) on the edge $e_j$ starting at $r$ and let $C'$ be the point set computed in line 17 of the procedure. Then

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

**Proof.** Let $q \in C'$. 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 $(1 + \varepsilon)\delta$-stabber. Hence, Lemma 43 implies that $\overline{r q}$ is also a $(1 + \varepsilon)\delta$-stabber, implying $C' \subseteq S_{(1+\varepsilon)\delta}$.

Now let $q \in e_j$ be an arbitrary point such that $\overline{r q}$ is a $\delta$-stabber. Let $t$ be the last realizing point of $\overline{r q}$. The line segment $\overline{r t}$ is a $\delta$-stabber and $t$ lies in $b_\delta(v_i)$. We claim that $t$ lies in $H$. Consider the set $G = \bigcap_{k \in S_{\delta}} b_{\delta k}(t)$. The properties of the grid, $t$ lies within the convex hull of $G$. Moreover $G \subseteq b_{(1+\varepsilon)\delta}(v_i)$. Lemma 44 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 $b_{(1+\varepsilon)\delta}(a_1)$, $s' t'$ is also a $((1 + \varepsilon)\delta)$-stabber, hence, $t'$ would have been marked as an eligible endpoint of a $((1 + \varepsilon)\delta)$-stabber. Since $H$ is the convex hull of eligible points, it follows that $t \in \operatorname{conv}(G) \subset H$. Therefore $q \in H'$ and thus $q \in C'$. \hfill $\square$

**Lemma 46.** For any $1 \leq i \leq n_1, 1 \leq j \leq n_2, 1 \leq s \leq k$ let $D_{i,j}^s$ be the endpoints of diagonal tunnels as computed in the APPROXIMATEDECIDER procedure, and let $R = \bigcup_{a=1}^{a} \bigcup_{b=1}^{b} p_{a,b}$ be the set of reachable points by exactly $s - 1$ proper tunnels in the lower left quadrant of the cell $C_{i,j}$. It holds that

(i) there exists a point $p \in R$ such that for any $q \in C_{i,j}$ the diagonal tunnel $r(p, q)$ has price $\operatorname{prc}(r(p, q)) \leq 3\delta$ then $q \in D_{i,j}^s$. If $q \in D_{i,j}^s$ then $\operatorname{prc}(r(p, q)) \leq 3(1 + \varepsilon)\delta$ and.

(ii) there exists no other $b \in C_{i,j} \setminus D_{i,j}^s$ that is the endpoint of a diagonal tunnel from $R$ with price at most $\delta$. 

ACM Trans. Algor.
Proof. The first part follows immediately from Lemma 45 together with the process described by the algorithm: The point \( p \) is simply the rightmost point in \( R \), which is maintained in \( g'_s \) (by an induction argument) at the time, where \( C_{i,j} \) is processed. Assume \( p = (x_p, y_p) \) lies in cell \( C_{a,b} \). We call the \textsc{APxDiagonalTunnel} procedure with \( p \) and the vertices \( v_{a+1}, \ldots, v_i \) between the \( a \)th and \( i \)th edge of the target curve. It returns points \( q = (x_q, y_q) \) inside the \( \delta \)-free-space such that \( \bar{B}[y_q, y_q] \) stabs through the sequence \( b_{3(1+\varepsilon)^2\delta}(v_{a+1}), \ldots, b_{3(1+\varepsilon)^2\delta}(v_i) \). Since \( p \) and \( q \) are in the \( \delta \)-free-space of \( B \) and \( T \), \( ||T(x_p) - B(y_p)|| \leq \delta \) and \( ||T(x_q) - B(y_q)|| \leq \delta \) which together imply \( d_T(\bar{B}[y_q, y_q], T[x_p, x_q]) \leq 3(1 + \varepsilon)^2\delta \).

Assume for the sake of contradiction the second part that such a point \( b \) does exist and the start point of the shortcut is \( s \in R \). Then by Lemma 42 all tunnels \( \tau(r, b) \) with \( x_s < x_r \) have price at most \( 3\delta \). In particular \( \text{prc}(\tau(p, b)) \leq 3\delta \), but then \( b \) would have been in \( D_s^i \) already.

**Lemma 47.** Given two polygonal curves \( T \) and \( B \) in the plane as well as parameters \( \varepsilon > 0 \) and \( \delta > 0 \), the \textsc{ApproximateDecider} computes a decision of either \( d^*_3(T, B) > \delta \) or \( d^*_3(T, B) \leq 3(1 + \varepsilon)^2\delta \).

**Proof.** We show that
\[
R_{\delta,s}(T, B) \subseteq \bigcup_{s' \leq s} P'_{i,j} \subseteq R_{3(1+\varepsilon)^2\delta,s}(T, B),
\]
for \( s \leq k \).

This proof is by induction over the order of handled cells. We show the inclusions from the theorem for each cell, i.e.
\[
R_{\delta,s}^{(i,j)}(T, B) \subseteq \bigcup_{s' \leq s} P'_{i,j} \subseteq R_{3(1+\varepsilon)^2\delta,s}^{(i,j)}(T, B).
\]

Assume that \( (0,0) \in D_{\delta}^{(1,1)} \), else the algorithm would have returned a correct decision in line 2. For \( i = j = 1 \) we have that \( P_{i,1}^0 = D_{\delta}^{(1,1)} \) which is correct by convexity of \( D_{\delta}^{(1,1)} \). For all other \( s \) we have that \( P_{i,1}^s = \emptyset \). This follows from the fact that there is no points in the column below or in the lower left quadrant of \( C_{1,1} \). Thus for \( i = j = 1 \) we have \( \bigcup_{s' \leq s} P'_{i,j} = R_{\delta,s}^{(i,j)} \subseteq R_{3(1+\varepsilon)^2\delta,s}^{(i,j)} \).

Consider the algorithm handling some cell \( C_{i,j} \). By induction all cells \( C_{\leq n, < j} \) and \( C_{< i,j} \) and in particular \( C_{i-1,j} \) and \( C_{i,j-1} \) have been handled correctly up to \( s \). Hence, their reachability intervals and left- and right most points can be have been computed correctly and are stored in their respective arrays. We need to show that \( R_{\delta,s}^{(i,j)} \subseteq \bigcup_{s' \leq s} P'_{i,j} \). Thus let \( q \in R_{\delta,s}^{(i,j)} \) be the endpoint of a monotone path from \((0,0)\) walking monotonously through \( D_{\delta}^{(i,j)} \) using \( s' \leq s \) proper tunnels of cost \( \delta \). There are three possibilities of how the path could have entered \( C_{i,j} \).

The path could have taken \( s' \) shortcuts to enter a neighboring cell and then walked into \( C_{i,j} \) through its boundary at some point \( a \). Since \( C_{i-1,j} \) and \( C_{i,j-1} \) have been handled correctly, \( a \) is in the computed reachability interval of the neighboring cell. Since the path must be monotone \( q \) lies in the closed halfplane fixed at the lower left end of the reachability interval in the respective directions, thus \( q \) is also in \( P'_{i,j} \).

Alternatively the path could have entered some cell \( C_{i,j} \) with \( s' = 1 \) shortcuts and then took a horizontal shortcut into \( C_{i,j} \) for some \( j < l \). By Lemma 18 together with the induction hypothesis for \( P_{i,j} \) we have that \( q \) is in \( P'_{i,j} \).

Similarly, if the path took a diagonal shortcut, we can apply Lemma 46 together with the induction hypothesis for \( P_{i,j} \), showing that \( q \) is in \( P'_{i,j} \), implying the left inclusion \( R_{\delta,s}^{(i,j)} \subseteq \bigcup_{s' \leq s} P'_{i,j} \).

Now let us assume that \( q \in P'_{i,j} \) for some \( s' \leq s \). We want to show that \( q \in R_{\delta,s}^{(i,j)} \). It must be that either (i) \( q \) is in \( N_{\delta}^i \), (ii) \( q \) is in \( V_{\delta,j} \), (iii) \( q \) is in \( D_{\delta,j} \), or (iv) \( q \) is in the upper right quadrant of some point \( p \), where \( p \) satisfies (i), (ii) or (iii). Thus we can reduce this to the first 3 cases.

In Case (i) the claim follows immediately, because \( P'_{i-1,j} \) and \( P'_{i,j-1} \) have been computed correctly.
In Case (ii) the claim follows immediately as well, due to the fact that \( P^Y_{i,j} \) have been computed correctly and the leftmost point is stored correctly in \( \bar{g}^{i-1} \), together with Lemma 18.

Case (iii) follows rather straightforward as well, since \( P'_{i,j} \) have been computed correctly and thus the rightmost point in the lower left quadrant of \( C_{i,j} \) that was reachable by \( s' \)−1 shortcuts is correctly stored in \( \bar{g}^{i-1} \). By Lemma 46 the apxDiagonalTunnel has precisely the guarantee that the endpoints of shortcuts are contained within the set of shortcuts with price at most \( 3(1 + \epsilon)^2 \delta \), the claim follows as well. Since \( P'_{i,j} \) fulfills all these requirements and thus computes all its left- and right-most points and reachability intervals correctly, the induction follows. Hence, \( R_{\delta,s}(T', B') \subseteq \bigcup_{s' \leq s} P'_{s'} \subseteq R_{\delta(1+\epsilon)^2, \delta s}(T', B') \). The algorithm output reflects the fact whether or not \((1, 1)\) is in \( P^{\leq k} \) proving the claim.

**Theorem 48.** Let \( T \) and \( B \) be two polygonal curves in the plane with overall complexity \( n \), together with values \( 0 < \epsilon \leq 1 \) and \( \delta > 0 \). The ApproximateDecider procedure correctly computes a decision of either \( d^S_{\delta}(T, B) > \delta \) or \( d^S_{\delta}(T, B) \leq (3 + \epsilon)\delta \).

**Proof.** This follows directly from the choice of \( \epsilon' \) and Lemma 47. This follows from the fact that \( \epsilon \leq 1 \) and \( \epsilon' = \epsilon / 9 \) hold, which implies \( d^S_{\delta}(T, B) \leq 3(1 + \epsilon')^2 \delta' < (3 + \epsilon)\delta \). \( \square \)

### 5.2.2 Running time

**Theorem 49.** Let \( T \) and \( B \) be two polygonal curves in the plane with overall complexity \( n \), together with values \( 0 < \epsilon \leq 1 \) and \( \delta > 0 \). There exists an algorithm with running time in \( O(kn^2 \epsilon^{-3} \log^3(n \epsilon^{-1})) \) and space in \( O(kn^2 \epsilon^{-4} \log^2(\epsilon^{-1})) \) which outputs one of the following: (i) \( d^S_{\delta}(T, B) \leq (3 + \epsilon)\delta \) or (ii) \( d^S_{\delta}(T, B) > \delta \). In any case, the output is correct.

**Proof.** We claim that the ApproximateDecider procedure fulfills these requirements.

As \( \epsilon' = \frac{\epsilon}{5} \), we can replace \( \epsilon' \) with \( \epsilon \) in the running time. For the precomputation we initialize the datastructure presented by Driemel and Har-Peled [Driemel and Har-Peled 2012] from Lemma 40. This precomputation takes \( O(\epsilon^{-4} \log^2(\epsilon^{-1}) n \log^2(n)) \) time. We iterate over all \( O(n^2) \) cells \( k \) times. The computation for each of these \( O(kn^2) \) steps is dominated by a call to the apxDiagonalTunnel procedure. This procedure iterates over \( \mathcal{O}(\epsilon^{-2}) \) gridpoints, thus queries the data structure \( \mathcal{O}(\epsilon^{-2}) \) times where each query takes \( O(\epsilon^{-2} \log n \log \log n) \) time. Finally, we construct a convex hull and intersect it with a line, taking \( O(\mathcal{O}(\mathcal{O}(\epsilon^{-2}) \log \epsilon^{-1}) \) time as the complexity of the convex hull is \( \mathcal{O}(\epsilon^{-2}) \). Thus the overall running time of the apxDiagonalTunnel procedure is \( \mathcal{O}(\epsilon^{-4} \log n \log \log n) \).

Thus the overall running time is

\[
\mathcal{O}(\epsilon^{-4} \log^2(\epsilon^{-1}) n \log^2(n) + kn^2 \epsilon^{-1}(\epsilon^{-4} \log n \log \log n)) = \mathcal{O}(\mathcal{O}(\epsilon^{-5} n \log^2(n)) + kn^2 \epsilon^{-5} \log n \log n) = \mathcal{O}(kn^2 \epsilon^{-5} \log^2(n \epsilon^{-1})).
\]

The space follows directly from the space needed for the approximate distance data structure. All other data structures necessary for the algorithm use \( \mathcal{O}(n) \) or \( \mathcal{O}(\epsilon^{-2}) \) space. Hence, the space is \( \mathcal{O}(n \epsilon^{-4} \log^3(\epsilon^{-1})) \), as described in [Driemel and Har-Peled 2012]. The correctness of the output is guaranteed by Theorem 48. \( \square \)

### 5.3 Modified algorithm for \( c \)-packed curves

In the case that the input curves are \( c \)-packed, for some constant \( c \), we can modify the algorithm and achieve in near-linear running time in \( n \). For this, we follow the approach of Driemel and Har-Peled [Driemel et al. 2010] to first simplify the curves.
We now turn to analysing the modified algorithm as described in Section 5.3.

5.4 Analysis for these points, whenever we are in a non-empty cell. If the cell lies in a quadrant of a cell, we use two dimensional range trees described in [Berg, de et al. 2008]. Lastly, in order to store and retrieve the left- and rightmost points in a column below and in the lower-left non-empty cells, we only want to iterate over these non-empty cells. We solve this with a sweep line algorithm presented by Bentley and Ottmann [Bentley and Ottmann 1979]. From these intersections we can then reconstruct, which cells have as the number of cells in the parametric space, with non-empty free-space. A detailed description of this straightforward modification can be found in Appendix A.

Thirdly, instead of iterating over all cells, we only want to iterate over these non-empty cells. We solve this with an output-sensitive algorithm for computing the intersections of edges and the boundary of a given parameter. Define

$$N_{\delta}(T, B) = \# \left\{ (i, j) \in \{1, \ldots, n_1\} \times \{1, \ldots, n_2\} \mid D(i, j) \neq \emptyset \right\}$$

as the number of cells in the parametric space, with non-empty \( \delta \)-free-space.

Observation 52. Given two polygonal curves \( T \) and \( B \) of total complexity \( n_1 + n_2 = n \), then \( N_{\delta}(T, B) \leq n_1 n_2 \in O(n^2) \). For any two \( \varepsilon \)-packed curves \( X \) and \( Y \) in \( \mathbb{R}^d \) of total complexity \( n \), and two parameters \( 0 < \varepsilon < 1 \) and \( \delta > 0 \), we have that \( N_{\delta}(\text{simpl}(X, \varepsilon \delta), \text{simpl}(Y, \varepsilon \delta)) \leq n(9c + 6c\varepsilon^{-1}) = O(cne^{-1}) \).

Corollary 54. For all \( s \geq 0 \) we have that

$$N_{\delta}(\text{simpl}(X, s\varepsilon \delta), \text{simpl}(Y, s\varepsilon \delta)) \leq n(9c + 6cse^{-1}) = O(cn + scne^{-1})$$.

The following lemma by Driemel and Har-Peled shows that the shortcut Fréchet distance is approximately preserved under simplifications.

Lemma 55 ([Driemel et al. 2010]). Given a simplification parameter \( \mu \) and two polygonal curves \( X \) and \( Y \), let \( X' = \text{simpl}(X, \mu) \) and \( Y' = \text{simpl}(Y, \mu) \) denote their \( \mu \)-simplifications respectively. For all \( k \in \mathbb{N} \) it holds that

$$d_S^k(X', Y') - 2\mu \leq d_S^k(X, Y) \leq d_S^k(X', Y') + 2\mu.$$
5.4.1 Correctness.

**Theorem 56.** Given two c-packed curves $T$ and $B$ in the plane, as well as parameters $0 < \varepsilon \leq 1$ and $\delta > 0$, the algorithm correctly computes a decision of either $d_S^B(T, B) > \delta$ or $d_S^B(T, B) \leq (3 + \varepsilon)\delta$.

**Proof.** The algorithm defines $\varepsilon' = \varepsilon/20$, and $\delta' = \delta/(1 - 2\varepsilon')$, and $\varepsilon'\delta'$-simplifies $T$ and $B$, resulting in $T'$ and $B'$ respectively. As the main part of the algorithm is not modified, Lemma 47 guarantees a correct decision of either $d_S^B(T', B') > \delta'$ or $d_S^B(T', B') \leq 3(1 + \varepsilon')^2\delta'$. By Lemma 55, this decision implies a correct decision of either $d_S^B(T, B) > (1 - 2\varepsilon')\delta' - 2\varepsilon'\delta'$ or $d_S^B(T', B') \leq 3(1 + \varepsilon')^2\delta' + 2\varepsilon'\delta'$. From the choices of $\varepsilon'$ and $\delta'$ as well as from the fact that $\varepsilon < 1$ it follows that $\delta' - 2\varepsilon'\delta' = \delta$, and $3(1 + \varepsilon')^2\delta' + 2\varepsilon'\delta' < 3 + \varepsilon$, thus implying the claim. □

5.4.2 Running Time. In this section we analyse the running time of the algorithm. We begin by proving that we can find all non-empty cells in an output-sensitive manner.

**Lemma 57.** Let $B$ and $T$ be two polygonal c-packed curves and $0 < \varepsilon \leq 1$ and $\delta > 0$ be given. Then the number of pairwise intersections in the set consisting of edges and $\delta$-neighbourhood boundaries of edges of $\text{simpl}(B, \varepsilon\delta)$ and $\text{simpl}(T, \varepsilon\delta)$ is in $O(cn\varepsilon^{-1})$.

**Proof.** This lemma is proven by repeated applications of Corollary 54. Let $B' = \text{simpl}(B, \varepsilon\delta)$ and $T' = \text{simpl}(T, \varepsilon\delta)$. For $s = 0$, $B$ and $T$, Corollary 54 implies that there are only $O(cn\varepsilon^{-1})$ many intersections between edges of $B'$ and edges of $T'$. For $s = 0$, $B$ and $B$, Corollary 54 implies that there are only $O(cn\varepsilon^{-1})$ many intersections between edges of $B'$ and edges of $B'$, similarly for $T'$. For $s = 1$, $B$ and $T$, Corollary 54 implies that there are only $O(cn\varepsilon^{-1})$ many intersections between edges of $B'$ and neighbourhoods of edges of $T'$. Similarly for edges of $B'$ and neighbourhoods $B'$, and edges $T'$ and neighbourhoods $T'$. And lastly for $s = 2$, Corollary 54 implies the same for intersections of neighbourhoods and neighbourhoods, implying the claim. □

**Corollary 58.** Let $B$ and $T$ be two polygonal c-packed curves in the plane and $0 < \varepsilon \leq 1$ and $\delta > 0$ be given. Then all $O(cn\varepsilon^{-1})$ non-empty cells in the $\delta$-free-space of $\text{simpl}(B, \varepsilon\delta)$ and $\text{simpl}(T, \varepsilon\delta)$ can be found in $O(cn\varepsilon^{-1}\log(cn\varepsilon^{-1}))$ time.

**Proof.** We first compute the intersections of the set of edges of $B$ and boundaries of $\delta$-neighbourhoods of the edges of $T$. For this we can use an output-sensitive intersection-finding algorithm, such as the Bentley-Ottmann algorithm [Bentley and Ottmann 1979], see also Appendix A. From these intersections we can then reconstruct the edge-pairs of $B$ and $T$ with non-empty $\delta$-free-space in $O(cn\varepsilon^{-1}\log(cn\varepsilon^{-1}))$ time, by tracing the edges of $B$ in the arrangement of $\delta$-neighbourhoods of $T$. Note that we also need to compute the edges that lie completely inside the $\delta$-neighbourhoods of other edges, however these are surrounded by edges intersecting (i.e. entering and leaving) the boundary of the same neighbourhood. Hence, we can process the edges of $B$ in the order along $B$ and find all edge-pairs of $B$ and $T$ with non-empty $\delta$-free-space. □

**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 $O(kcn\varepsilon^{-5}\log^2(ne^{-1}))$ and space in $O(kcn\varepsilon^{-4}\log^2(e^{-1}))$ which outputs one of the following: (i) $d_S^B(T, B) \leq (3 + \varepsilon)\delta$ or (ii) $d_S^B(T, B) > \delta$. In any case, the output is correct.

**Proof.** The non-trivial steps of the algorithm are: (i) Precomputation on the curves, (ii) finding all non-empty cells, (iii) iterating over these cells, (iv) the $\text{APXDIAGONALTUNNEL}$ procedure and (v) storing and restoring the rightmost gate.

As $\varepsilon' = \frac{\varepsilon}{20}$, we can replace $\varepsilon'$ with $\varepsilon$ in the running time. For the precomputation we initialize the datastructure presented by Driemel and Har-Peled [Driemel and Har-Peled 2012] from Lemma 40. This precomputation
takes $O(\varepsilon^{-4} \log^5(\varepsilon^{-1}) n \log^2(n))$ time. In Corollary 58 we showed, that finding all intersections can be done in $O(cn\varepsilon^{-1} \log(cn\varepsilon^{-1}))$ time. Sorting these intersections in $O(cn\varepsilon^{-1} \log(cn\varepsilon^{-1}))$ time alphanumerically by the two indices, allows us to iterate over these cells as described in the algorithm. In Section 5.1 we described the apxDiagonalTunnel procedure. This procedure iterates over $O(\varepsilon^{-2})$ gridpoints, thus queries the data structure $O(\varepsilon^{-2})$ times where each query takes $O(\varepsilon^{-2} \log n \log \log n)$ time. Finally, we construct a convex hull and intersect it with a line, taking $O(\varepsilon^{-2} \log \varepsilon^{-1})$ time as the complexity of the convex hull is $O(\varepsilon^{-2})$. Thus the overall running time of the apxDiagonalTunnel procedure is $O(\varepsilon^{-4} \log n \log \log n)$. We call this procedure $O(kcn\varepsilon^{-1})$ times, $k$ times for each nonempty cell.

To store the rightmost gate in the lower left quadrant we can use two dimensional range trees as described in [Berg, de et al. 2008]. We build this tree with $O(cn\varepsilon^{-1})$ points at the end of each outer loop storing all right- and left-most points for the next iteration in $O(cn\varepsilon^{-1} \log(cn\varepsilon^{-1}))$ time. As we do this $k$ times, this results in an overall running time of $O(kcn\varepsilon^{-1} \log(cn\varepsilon^{-1}))$, where the space used is $O(cn\varepsilon^{-1} \log(cn\varepsilon^{-1}))$.

Thus the overall running time is

$$O(cn\varepsilon^{-1} \log(cn\varepsilon^{-1}) + \varepsilon^{-4} \log^5(\varepsilon^{-1}) n \log^2(n) + kcn\varepsilon^{-1}(\varepsilon^{-4} \log n \log \log n)) = O(cn\varepsilon^{-1} \log(ne^{-1}) + \varepsilon^{-5} n \log^2(n) + kcn\varepsilon^{-5} \log n \log \log n) = O(kcn^{-5} \log^2(ne^{-1})).$$

The space follows directly from the space needed for the approximate distance data structure. All other data structures necessary for the algorithm use $O(n)$ or $O(\varepsilon^{-2})$ space. Hence, the space is $O(n^{-5} \log^2(\varepsilon^{-1}))$, as described in [Driemel and Har-Peled 2012]. The approximation ratio is guaranteed by Theorem 56.

**ACKNOWLEDGMENTS**

This work has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - 313421352 (FOR 2535 Anticipating Human Behavior).

**REFERENCES**

Hugo Alves Akitaya, Maike Buchin, Leonid Ryvkin, and Jérôme Urhausen. 2019. The k-Frèchet Distance: How to Walk Your Dog While Teleporting. In 30th International Symposium on Algorithms and Computation, Vol. 149. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 50:1–50:15. https://doi.org/10.4230/LIPIcs.ISAC.2019.50

Noga Alon, Raphael Yuster, and Uri Zwick. 1995. Color-Coding. Journal of the ACM 42, 4 (1995), 844–856. https://doi.org/10.1145/210332.210337

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

Helmut Alt and Michael Godau. 1995. Computing the Fréchet Distance between Two Polygonal Curves. International Journal of Computational Geometry & Applications 5 (1995), 75–91. https://doi.org/10.1142/S0218195995000064

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

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

J. L. Bentley and T. A. Ottmann. 1979. Algorithms for Reporting and Counting Geometric Intersections. IEEE Transactions on Computers 28, 9 (Sept 1979), 643–647. https://doi.org/10.1109/T-C.1979.1675432

M. Berg, de, O. Cheong, M.J. Kreveld, van, and M.H. Overmars. 2008. Computational geometry : algorithms and applications (3rd ed.). Springer, Germany. https://doi.org/10.1007/978-3-540-77974-2

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

Karl Bringmann and Wolfgang Mulzer. 2016. Approximability of the discrete Fréchet distance. Journal of Computational Geometry 7, 2 (2016), 46–76. https://doi.org/10.20382/jocg.v7i2a4
Alexander M Bronstein, Michael M Bronstein, Alfred M Bruckstein, and Ron Kimmel. 2009. Partial Similarity of Objects, or How to Compare a Centaur to a Horse. International Journal of Computer Vision 84, 2 (2009), 163. https://doi.org/10.1007/s11263-008-0147-3

Kevin Buchin, Maike Buchin, Wouter Meulemans, and Wolfgang Mulzer. 2017. Four Soviets Walk the Dog: Improved Bounds for Computing the Fréchet Distance. Discrete & Computational Geometry 58, 1 (2017), 180–216. https://doi.org/10.1007/s00454-017-9878-7

Kevin Buchin, Maike Buchin, and Yusu Wang. 2009. Exact Algorithms for Partial Curve Matching via the Fréchet Distance. In Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics, 645–654. https://doi.org/10.1137/1.9781611973068.71

Kevin Buchin, Tim Ophelders, and Bettina Speckmann. 2019. 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. Society for Industrial and Applied Mathematics, 2887–2901. https://doi.org/10.1137/1.9781611975482.179

Maike Buchin, Anne Driemel, and Dennis Rohde. 2023. Approximating (k,l)-Median Clustering for Polygonal Curves. ACM Transactions on Algorithms 19, 1, Article 4 (feb 2023), 32 pages. https://doi.org/10.1145/3559764

Maike Buchin, Anne Driemel, and Bettina Speckmann. 2014. Computing the Fréchet Distance with Shortcuts is NP-Hard. In Proceedings of the Thirtieth Annual Symposium on Computational Geometry. Association for Computing Machinery, 367–376. https://doi.org/10.1145/2582112.2582144

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

Elena Deza, Michel Marie Deza, Michel Marie Deza, and Elena Deza. 2009. Encyclopedia of distances. Springer.

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

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

Jeanette P. Schmidt and Alan Siegel. 1990. The Spatial Complexity of Oblivious k-Probe Hash Functions. SIAM Journal on Computing 19, 5 (1990), 775–786. https://doi.org/10.1137/0219054

R.C. Veltkamp. 2001. Shape matching: similarity measures and algorithms. In Proceedings International Conference on Shape Modeling and Applications. IEEE Computer Society, 188–197. https://doi.org/10.1109/SMA.2001.923589

ACM Trans. Algor.
A INTERSECTION FINDER

Algorithm 3 Intersection Finder

1: procedure IntersectionFinder($\varepsilon\delta$-simplifications $X'$ and $Y'$, $\delta$)
2: Insert every start- and end-point of edges on $X'$ and every start- and end-points of arcs of $\delta$-capsules like in Figure 13 into $E$, a priority queue
3: Sort $E$ by firstly the $x$-coordinate and secondly by the $y$-coordinate
4: Let $A$ be a self-balancing empty binary tree and $I$ an empty array
5: while $E \neq \emptyset$ do
6: pop the head object off $E$ into $x$
7: if $x$ is the first vertex to be inserted of an object $o$ then
8: sorted insert $o$ into $A$ by its current $y$-coordinate
9: compute the point of intersection of $x$ and its at most two neighbours
10: sorted insert this point by its $x$-coordinate into $E$
11: if $x$ is the second vertex to be inserted of an object $o$ then
12: remove $o$ from $A$
13: if $x$ corresponds to an intersection between $o$ and $o'$ then
14: insert the intersection to $I$
15: swap $o$ and $o'$ in $A$ and compute the point of intersection with their new neighbours updating $E$
16: Return $I$

For completeness sake we provide a detailed description of the modified version of the Bentley-Ottman sweep-line algorithm [Bentley and Ottmann 1979].

**Lemma 59.** Given two polygonal curves $X$ and $Y$ in $\mathbb{R}^2$, a parameter $\delta \geq 0$ and let $X'$ and $Y'$ be their $\varepsilon\delta$-simplifications. One can find all $N_{\delta}(X', Y')$ cells in the free-space diagram that have non-empty $\delta$-free-space in $O\left(\frac{\varepsilon^2}{\delta^2} \log \left(\frac{\varepsilon^3}{\delta^2} \right)\right)$ time.

**Proof.** Without loss of generality it suffices to find all edges of the curve $X'$ that enter and exit a $\delta$-neighbourhood of any edge of $Y'$ since any edge that is completely contained in this neighbourhood lies between two edges entering and leaving the neighbourhood. In the special case that the start or end vertex of $X'$ lies in such a neighbourhood it is easily checked by looking whether the first (resp. last) such edge is entering or leaving the neighbourhood. Entering and exiting such a neighbourhood is the same as intersecting its boundary. Thus we can modify for example the Bentley-Ottman algorithm [Bentley and Ottmann 1979] to find all intersections in a set of edges (refer to Algorithm 3). The main idea is to sweep along the $x$-axis and keep track of all objects that cross the sweeping line in an array of size $O(n)$. Every time a new object enters the array it checks with its at most two neighbours how far the sweeping line would have to sweep to get to the intersection point of the new object. If an intersection occurs at some time in the future, we add this event to the event queue of the sweeping line. If the sweeping line is at an intersection event, it swaps the two objects in question and updates all new $O(1)$ neighbours. We can modify this easily to work with capsules (the geometric shape of the $\delta$-neighbourhood of an edge) by introducing two sections of the capsule into the array instead of a...
single line, as can be seen in Figure 13. Intersections with its neighbours can still be checked and updated in \( O(1) \). The algorithm runs in \( O((n + k) \log(n + k)) \) time for \( k \) intersecting objects. From Lemma 57 we know that the number of intersections of the described objects is bound in \( O(2^n) \). Hence Lemma 53 implies the claim. □

B MODIFIED ALGORITHM

Here we restate the algorithm presented by Buchin, Driemel and Speckmann [Buchin et al. 2014] with our modifications resulting in an improved running time.

**Algorithm 4** Modified Algorithm

```plaintext
1: procedure MODIFIED_DECIDER(curves \( T \) and \( B \), \( \delta > 0 \) and \( 0 < \epsilon \leq 1 \))
2: Let \( \epsilon' = \frac{\epsilon \delta}{2n} \)
3: Assert that \( ||T(0) - B(0)|| \leq \delta \) and \( ||T(1) - B(1)|| \leq \delta \)
4: Let \( \mathcal{A}, \mathcal{A}_r, \mathcal{R}_r, g_l, g_r \) be arrays of size \( n_1 \)
5: for \( j = 1, \ldots, n_2 \) do
6: Update \( \mathcal{A} \leftarrow \mathcal{A}_r \leftarrow g_l, g_r \leftarrow g_r \)
7: for \( i = 1, \ldots, n_1 \) do
8: if \( i = 1 \) and \( j = 1 \) then
9: \( P_{i,j} = D_{\delta, \sigma}^{(i,j)} \)
10: else
11: Retrieve \( VR_{(i-1,j-1)} \) and \( HR_{(i-1,j)} \) from \( \mathcal{A}[i] \) and \( \mathcal{A}[i-1] \)
12: Compute \( N_{i,j} \) from \( V R_{(i,j-1)} \) and \( H R_{(i,j-1)} \)
13: Let \( V_{i,j} = VERTICAL_TUNNEL(\mathcal{A}_r[i], C_{i,j}, \delta) \)
14: Compute \( D_{i,j} = APX_DIAGONAL_TUNNEL(\mathcal{A}_r[i-1], C_{i,j}, \sigma', 3\delta) \)
15: Let \( P_{i,j} = Q(N_{i,j} \cup D_{i,j}) \cup V_{i,j} \cap D_{\delta, \sigma}^{(i,j)} \)
16: if \( P_{i,j} \neq \emptyset \) then
17: Update \( g_l[i] \) and \( g_r[i] \) using \( P_{i,j} \)
18: Compute \( V R_{(i,j)} \) and \( H R_{(i,j)} \) and store them in \( \mathcal{A}[i] \)
19: else
20: Update \( g_r[i] \) using \( g_r[i-1] \)
21: if \( (1, 1) \in \mathcal{A}[n_1] \) then
22: Return \( 'd_S(T, B) \leq (3 + \epsilon)\delta' \)
23: else
24: Return \( 'd_S(T, B) > \delta' \)
```

ACM Trans. Algor.