On optimal min-# curve simplification problem

DOI:
10.48550/arXiv.1809.10269

Document status and date:
Published: 26/09/2018

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

On Optimal Min–# Curve Simplification Problem

Mees van de Kerkhof[1], Irina Kostitsyna[2], Maarten Löffler[1], Majid Mirzanezhad[3], and Carola Wenk[3]

1 **Utrecht University, Netherlands**
2 **Eindhoven University of Technology, Netherlands**
3 **Tulane University, USA**

Abstract

In this paper we consider the classical min–# curve simplification problem in three different variants. Let $\delta > 0$, P be a polygonal curve with n vertices in $\mathbb{R}^d$, and $\mathsf{D}(\cdot,\cdot)$ be a distance measure. We aim to simplify P by another polygonal curve P' with minimum number of vertices satisfying $\mathsf{D}(P,P') \leq \delta$. We obtain three main results for this problem: (1) An $O(n^4)$-time algorithm when $\mathsf{D}(P,P')$ is the Fréchet distance and vertices in P' are selected from a subsequence of vertices in P. (2) An NP-hardness result for the case when $\mathsf{D}(P,P')$ is the directed Hausdorff distance from P' to P and the vertices of P' can lie anywhere on P while respecting the order of edges along P. (3) For any $\epsilon > 0$, an $O^*(n^2 \log n \log\log n)$-time algorithm that computes P' whose vertices can lie anywhere in the space and whose Fréchet distance to P is at most $(1+\epsilon)\delta$ with at most $2m+1$ links, where m is the number of links in the optimal simplified curve and O^* hides polynomial factors of $1/\epsilon$.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Geometrical problems and computations

Keywords and phrases Computational Geometry - Fréchet distance - Simplification- Polygonal Curves

Digital Object Identifier 10.4230/LIPIcs...

1 Introduction

Approximating a polygonal curve by another curve is a long-standing problem in computational geometry. One of the most well-known settings that has received considerable attention is the *min–# problem*. That is, given a polygonal curve $P = \langle p_1, p_2, \cdots, p_n\rangle$ in $\mathbb{R}^d$, a distance measure $\mathsf{D}(\cdot,\cdot)$ between two curves and a real value $\delta > 0$, find a polygonal curve $P' = \langle p'_1, p'_2, \cdots, p'_k\rangle$ with the minimum number of vertices such that $\mathsf{D}(P,P') \leq \delta$. We call the edges in P' *links*.

There are several variants of the min–# problem: (1) *vertex-restricted*, where vertices of P' have to be a subsequence of vertices of P, (2) *curve-restricted*, where vertices of P' can lie anywhere on P but have to respect the order along P, and (3) *non-restricted*, when vertices of P' can be anywhere in the ambient space. For all of the cases above, it is also a requirement that the start and end points of P' are identical to the start and end points of P. For the vertex-restricted and curve-restricted cases we also refer to links as *shortcuts*. Given a distance measure $\mathsf{D}(\cdot,\cdot)$ between two curves, such as Hausdorff δ_H, directed Hausdorff $\overrightarrow{\delta_H}$, or Fréchet distance δ_F, one can apply this distance in a *global* or *local* way in the min–# problem as follows: First, one can simply measure the distance $\mathsf{D}(P,P')$ between the two curves; we denote this as the *global* distance δ^g. In the local setting, for the vertex- and curve-restricted cases, one measures the distance between each link in P' and its corresponding subcurve in P whose endpoints are the same as the endpoints of the link,

Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

and returns the maximum of these distances. We denote such a local distance by δ^ℓ. More formally, let $P' = \langle P(s_1), P(s_2), ..., P(s_k)\rangle$ with $s_1 = 1 < s_2 < \ldots < s_k = n$ be a vertex- or curve-restricted simplification of P. Then:

$$\delta^\ell := \max_{1\leq i<k} \{\mathsf{D}(P[s_i, s_{i+1}], \langle P(s_i), P(s_{i+1})\rangle) ,$$

where $P[s_i, s_{i+1}]$ is a *subcurve of* P between two points $P(s_i)$ and $P(s_{i+1})$. We will provide the rest of our notation in the next sections.

In this paper, we focus on global distance measures under the three different variants of restrictions on where vertices of P' can be placed. To the best of our knowledge there are just a few works on this setting, obtaining preliminary results under global distance measures only for some variants of the problem. We believe it is beneficial to study the min–# problem under global distance measures more extensively so as to get a broader view on the topic.

1.1 Related Work

There have been numerous results on different variants of the min–# problem mostly for the vertex-restricted version under local distance measures. The classical algorithm proposed by Imai and Iri [13], uses a *shortcut graph* to solve the vertex-restricted min–# problem under $\overrightarrow{\delta^\ell_H}$ from P' to P. The edges of the shortcut graph are those shortcuts of P for which the distance between the shortcut and its corresponding subcurve is at most δ. Once all the shortcuts are processed, the shortest path in the shortcut graph represents P'. While their algorithm runs in $O(n^2 \log n)$ time, Chan and Chin [7] improved the running time to $O(n^2)$. Godau [10] considered the same problem under δ^ℓ_F and gave an $O(n^3)$-time algorithm. Guibas et al. [11] provided algorithms for computing minimum-link paths that stab a sequence of regions in order. One of the variants, presented in Theorems 10 and 14 of [11], computes the non-restricted δ^g_F in the plane in $O(n^2 \log^2 n)$ time. Bereg et al. [6] considered global discrete Fréchet distance (δ^g_{dF}) and obtained an $O(n^2)$ algorithm for the vertex-restricted variant and an $O(n \log n)$ time algorithm for the non-restricted variant. Agarwal et al. [1] gave a near linear time approximation algorithm for the vertex-restricted version under δ^ℓ_F for any L_p metric, where the number of edges of the simplified curve returned by their algorithm with respect to δ is at most the number of edges of an optimal simplified curve with respect to $\delta/2$.

There has been a lot of progress on solving the min–# simplification problem under the local variant for different types of distance measures. There also exist some results on the global variants. Recently Van Kreveld et al. [15] considered the global variant under different distance measures. They proved that the vertex-restricted min–# problem under $\delta^g_H(P, P')$ is NP-hard, whereas they gave an output sensitive polynomial time dynamic programming algorithm under $\delta^g_F(P, P')$. They distinguished between the results on the directed Hausdorff distance and the (undirected) Hausdorff distance. They gave a polynomial-time algorithm for the min–# simplification under $\overrightarrow{\delta^g_H}(P', P)$ (directed Hausdorff from P' to P), however they showed that the problem under $\overrightarrow{\delta^g_H}(P, P')$ (directed Hausdorff from P to P') is NP-hard. See Table 1 for an overview of existing results on different variants of the min–# simplification problem.

Distance type	Vertex-restricted	Curve-restricted	Non-restricted
$\overrightarrow{\delta_H^g}(P, P')$	NP-hard [15]	N/A	N/A
$\overrightarrow{\delta_H^g}(P', P)$	$O(n^4)$ [15] $O(n^2 \log n)$☆ (Section 5.2)	NP-hard☆ (Section 4)	poly(n) [14]
$\delta_H^g(P, P')$	NP-hard [15]	N/A	N/A
$\delta_F^g(P, P')$	$O(mn^5)$ [15] $O(n^4)$☆ (Section 3) $O(n^3)$☆ EREW PRAM	$O(n)$ in $\mathbb{R}^1$☆ (Section 5.1)	$O(n^2 \log^2 n)$ in $\mathbb{R}^2$ [11] $O^*\left(n^2 \log n \log\log n\right)$☆ (Section 5)
$\delta_{dF}^g(P, P')$	$O(n^2)$ [6]	N/A	$O(n \log n)$ [6]
$\overrightarrow{\delta_H^\ell}(P, P')$	N/A	N/A	N/A
$\overrightarrow{\delta_H^\ell}(P', P)$	$O(n^2)$ [7] $O(n^2 \log n)$ [13] $O(n^{4/3+\epsilon})$ [2]	N/A	N/A
$\delta_H^\ell(P, P')$	N/A	N/A	N/A
$\delta_F^\ell(P, P')$	$O(n^3)$ [10] $O(n \log n)$ [1]	N/A	N/A

Table 1 Known results for the min-# problem under global and local distance measures. Results with ☆ are presented in this paper.

1.2 Our results:

In this paper, we consider the simplification problems formed by the combination of two global distance measures, Fréchet distance and directed Hausdorff distance (δ_F^g and $\overrightarrow{\delta_H^g}$), with the three possible variants on vertex placement for P', vertex-restricted; curve-restricted; and non-restricted (see Table 1). In Section 3 for the vertex-restricted version under $\delta_F^g(P, P')$, we propose a $O(n^4)$-time dynamic programming algorithm using $O(n^3)$ space. This is an improvement of the $O(mn^5)$-time algorithm that uses $O(mn^2)$ space presented by Kreveld et al. [15] where m is the number of links in P'. Next in Section 4 we prove that computing $\overrightarrow{\delta_H^g}(P', P)$ for the curve-restricted min-# simplification becomes NP-hard. To the best of our knowledge, this is the first result in the curve-restricted setting under the global directed Hausdorff distance. In Section 5, we give an approximation algorithm running in $O^*(n^2 \log n \log\log n)$ time for any $\epsilon > 0$, that computes P' with at most $2m+1$ links whose local Fréchet distance to P is at most $(1+\epsilon)\delta$ where m is the number of links in optimal simplified path with respect to δ under global Fréchet distance, d is the dimension of the space and O^* hides polynomial factors of $1/\epsilon$. Compared to the algorithm in [11] that only applies to curves in 2D within $O(n^2 \log^2 n)$ time and $O(n)$ space, our algorithm approximately computes the simplified curve in any arbitrary dimension d in an slightly faster running time using linear space. In fact, our approximation provides an informative relationship between the number of links in simplified curves under local and global Fréchet distances unlike the approximation in [1] which only respects δ and does not explicitly guarantee any bounds on the number of links returned by the algorithm.

We end the paper by elaborating on some possible improvements on other different variants of the problem in Table 1 and leave some open problems.

2 Preliminaries

Let $P = \langle p_1, p_2, \cdots, p_n \rangle$ be a polygonal curve. We treat a *polygonal curve* as a continuous map $P : [1, n] \to \mathbb{R}^d$ where $P(i) = p_i$ for an integer i, and the i-th edge is linearly parametrized as $P(i+\lambda) = (1-\lambda)p_i + \lambda p_{i+1}$, for integer i and $0 < \lambda < 1$. A *re-parametrization* $\sigma : [0, 1] \to [1, n]$ of P is any continuous, non-decreasing function such that $\sigma(0) = 1$ and $\sigma(1) = n$. We denote the subcurve between $P(s)$ and $P(t)$ by $P[s, t]$, where $1 \le s \le t \le n$. Given two points x and y in $\mathbb{R}^d$, we denote the straight-line segment connecting between them by $\langle xy \rangle$.

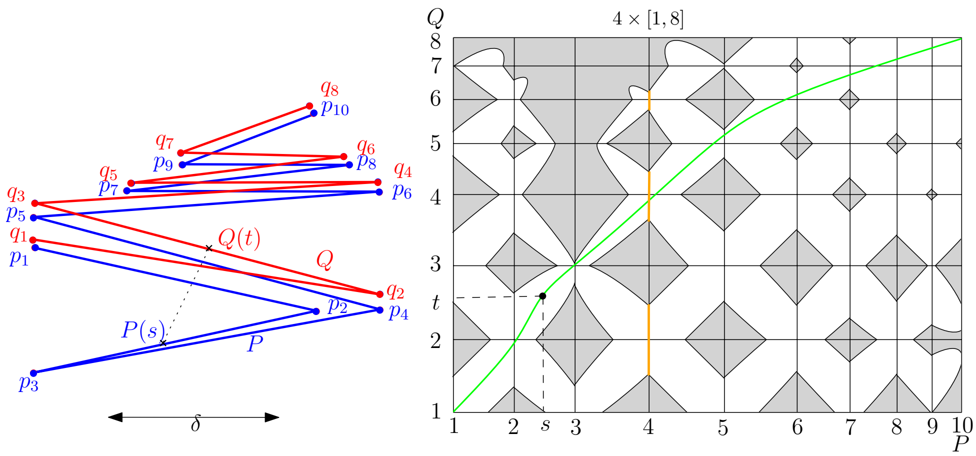

Figure 1 (s, t) is a free point on a reachable path in $\mathsf{FSD}_\delta(P, Q)$, where white space contains free points and gray space contains blocked points.

To compute the Fréchet distance between P and Q, Alt and Godau [4] introduced the notion of *free-space diagram*. For any $\delta > 0$, we denote the free-space diagram between P and Q by $\mathsf{FSD}_\delta(P, Q)$. This diagram has the domain of $[1, n] \times [1, m]$ and it consists of $(n - 1) \times (m - 1)$ cells, where each point (s, t) in the diagram corresponds to two points $P(s)$ and $Q(t)$. A point (s, t) in $\mathsf{FSD}_\delta(P, Q)$ is called *free* if $\|P(s) - Q(t)\| \le \delta$, and *blocked* otherwise. The union of all free points is referred to as the *free space*. The intervals induced by free space in $\mathsf{FSD}_\delta(P(i), Q)$ for all $i = 1, \cdots, n$ is called *free space intervals* of the range $i \times [1, m]$. *Fréchet matching* between P and Q is a pair of re-parameterizations (σ, θ) corresponding to an xy-monotone path from $(1, 1)$ to (n, m) within the free space in $\mathsf{FSD}_\delta(P, Q)$. The *Fréchet distance* between two curves is defined as $\delta_F(P, Q) = \inf_{(\sigma,\theta)} \max_{0 \le t \le 1} \|P(\sigma(t)) - Q(\theta(t))\|$, where (σ, θ) is a Fréchet matching and $\max_{0 \le t \le 1} \|P(\sigma(t)) - Q(\theta(t))\|$ is called the *width* of the matching. Let $a = (x_1, y_1)$ and $b = (x_2, y_2)$ be two points in $\mathsf{FSD}_\delta(P, Q)$ with $x_1 \le x_2$ and $y_1 \le y_2$. We say b is *reachable* from a if there exists a Fréchet matching from a to b within $\mathsf{FSD}_\delta(P, Q)$. A Fréchet matching in $\mathsf{FSD}_\delta(P, Q)$ from a to b is also called a *reachable path* between a and b denoted by $\mathcal{P}(a, b)$. If $a = (1, 1)$ we denote it by $\mathcal{P}(1, b)$. See Figure 1 for an illustration of these concepts. Note that for each reachable path there is a combinatorially equivalent piecewise linear reachable path, due to the convexity of free space within each cell [4]. We therefore assume that reachable paths are piecewise linear. Alt and Godau [4] compute a reachable path by propagating reachable points across free space cell boundaries in a dynamic programming manner, which requires the exploration of the entire $\mathsf{FSD}_\delta(P, Q)$ and takes $O(mn)$ time.

3 Vertex Restricted Problem Under $\delta_F^g(P, P')$ in $\mathbb{R}^d$

Let $P = \langle p_1, p_2, \cdots, p_n \rangle$ be a polygonal curve with n vertices, where $P : I \to \mathbb{R}^d$ with $I = [1, n]$. We construct a DAG $G = (V, E)$ such that $V = \{1, 2, \cdots, n\}$ and $E = \{(i, j) \mid 1 \leq i < j \leq n\}$. Here, we consider each vertex $v \in V$ to be embedded at p_v and each edge $(u, v) \in E$ to be embedded as the straight line segment shortcut $\overline{p_u p_v}$ between p_u and p_v, parameterized linearly by $\overline{p_u p_v}(j) = \frac{v-j}{v-u} p_u + \frac{j-u}{v-u} p_v$ for all $j \in [u, v]$. Note that E contains all the (directed) shortcuts in P whose start and end points are vertices of P. For any $\delta > 0$ we now define the free space surface for P and G; see [4, 3] for definitions of free space and free space surface.

▶ **Definition 1** (Strips and Spines)**.** Let $(u, v) \in E$. The *δ-strip*

$$\mathsf{ST}_\delta(u, v) = \{(i, j) \mid 1 \leq i \leq n, u \leq j \leq v, ||P(i) - \overline{p_u p_v}(j)|| \leq \delta\}$$

is the free space between P and $\overline{p_u p_v}$. The *strip* is defined as $\mathsf{ST}(u, v) = I \times [u, v]$. We have $\mathsf{ST}_\delta(u, v) \subseteq \mathsf{ST}(u, v)$. The *$\delta$-spine*

$$\mathsf{SP}_\delta(v) = \{(v, i) \mid 1 \leq i \leq n, ||P(i) - p_v|| \leq \delta\}$$

is the free space between P and p_v. The *spine* is defined as $\mathsf{SP}(v) = I \times v$. We have $\mathsf{SP}_\delta(v) \subseteq \mathsf{SP}(v)$.

For any edge $(u, v) \in E$, both spines centered at the vertices of the edge are subsets of the strip: $\mathsf{SP}_\delta(u), \mathsf{SP}_\delta(v) \subseteq \mathsf{ST}_\delta(u, v)$ and $\mathsf{SP}_\delta(u)$ is a subset of all strips with respect to edges incident on u. The *free space surface* of P and G, denoted by $\mathsf{FSD}_\delta(P, G)$, is the collection of all strips (and spines) over all $v \in V$ and $(u, v) \in E$; see Figure 2. Note that the edges in E are directed, and hence any reachable path has to visit a sequence of spines that corresponds to an increasing sequence of vertices in V.

We consider the spines to be "vertical" and shortcut edges in G to be "horizontal". Each free space cell C in $\mathsf{ST}(u, v)$ contains two vertical boundaries on $\mathsf{SP}(u)$ and $\mathsf{SP}(v)$, and two

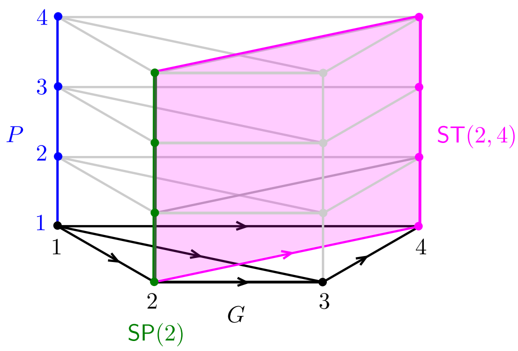

Figure 2 Schematic example of a free space surface for a polygonal curve $P = \langle p_1, p_2, p_3, p_4 \rangle$ (in blue) and its shortcut graph G (in black). The strip $\mathsf{ST}(2, 4)$ is highlighted in pink, and the spine $\mathsf{SP}(2)$ is highlighted in green. Free space is not shown.

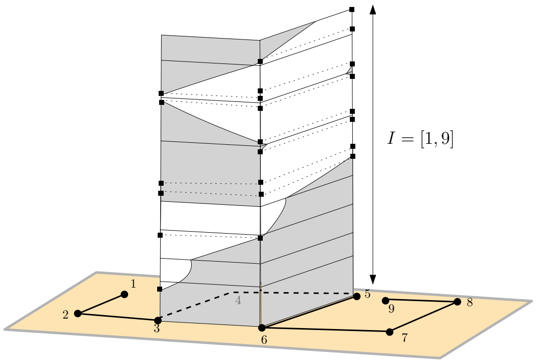

Figure 3 Elementary intervals in each spine are created by overlaying all free space intervals onto each spine.

horizontal boundaries. The free space on each of these boundaries is known to be a single interval (which may be empty), see [4].

The goal of our algorithm is to compute a reachable path in the free space surface from $(1,1)$ to (n,n) that uses the minimum number of strips. The main idea is to use dynamic programming to propagate a reachable path with a minimum number of strips from spine to spine. Each spine $\mathsf{SP}(v) = I \times v$ contains a sequence of free space intervals of $\mathsf{SP}_\delta(v)$. Let $S \subseteq I$ be the union of all interval endpoints of free space intervals of $\mathsf{SP}_\delta(v)$ for all $v \in V$, projected onto I. The set S induces a partition of I into intervals. For each $v \in V$ let $\mathsf{L}_\delta(v)$ be the ordered list of free space intervals obtained by subdividing the intervals of $\mathsf{SP}_\delta(v)$ with all points in S. We call the intervals in $\mathsf{L}_\delta(v)$, whose ending points are excluded along I, *elementary intervals*; see Figure 3.

We assume that elementary intervals in $\mathsf{L}_\delta(v)$ are ordered in increasing order of their starting point. When clear from the context we may identify elementary intervals with their projections onto I, and use $<$ and $=$ to compare intervals in the resulting total ordering of all elementary intervals along I.

We now extend the definition of a reachable path $\mathcal{P}(a,b)$ whose starting and ending points are defined with respect to two points a and b in $\mathsf{FSD}_\delta(P,Q)$ to the one that is defined with respect to two elementary intervals $r \in \mathsf{L}_\delta(u)$ and $e \in \mathsf{L}_\delta(v)$ with $r \le e$ and $u \le v$. We denote a *elementary reachable path* from an elementary interval r to elementary interval e by $\overline{\mathcal{P}}(r,e)$. $\mathbf{L}(\overline{\mathcal{P}}(r,e))$ denotes the length of a reachable path that is the number of strips visited by $\overline{\mathcal{P}}(r,e)$. We can also define the same notion of length for a reachable path $\mathcal{P}(a,b)$ between two points a and b. If an elementary reachable path starts from $(1,1)$ and ends to $e \in \mathsf{L}_\delta(v)$ we denote it by $\overline{\mathcal{P}}(1,e)$ for more simplicity in our notation.

We define the cost function $\phi : V \times I \to \mathbb{N}$ for any point $z = (v,x) \in \mathsf{SP}_\delta(v)$ with $v \in V$ as $\phi(v,z) = \min_{\mathcal{P}(1,z)} \mathbf{L}(\mathcal{P}(1,z))$, where the minimum ranges over all reachable paths $\mathcal{P}(1,z)$ in the free space surface. If no such path exists then $\phi(v,z) = \infty$. We will show how to use dynamic programming to propagate ϕ across the free space surface. Lemma 2 below shows some properties of elementary intervals and their sufficiency to propagate ϕ values across $\mathsf{FSD}_\delta(P,Q)$.

▶ **Lemma 2** (Elementary Intervals Properties). *Let $v \in V$. The following statements are true:*

1. $r \cap e = \emptyset$ *and* $\cup_{o \in \mathsf{L}_\delta(v)} \{o\} = \mathsf{L}_\delta(v)$ $(\forall (r,e) \in \mathsf{L}_\delta(v)\ r \neq e)$,
2. $|\mathsf{L}_\delta(v)| \le 2n^2 + n$,
3. *For any two points* $x = (v,a)$ *and* $y = (v,b)$ *with* $x, y \in e$ *and* $e \in \mathsf{L}_\delta(v)$, $\phi(v,x) = \phi(v,y)$

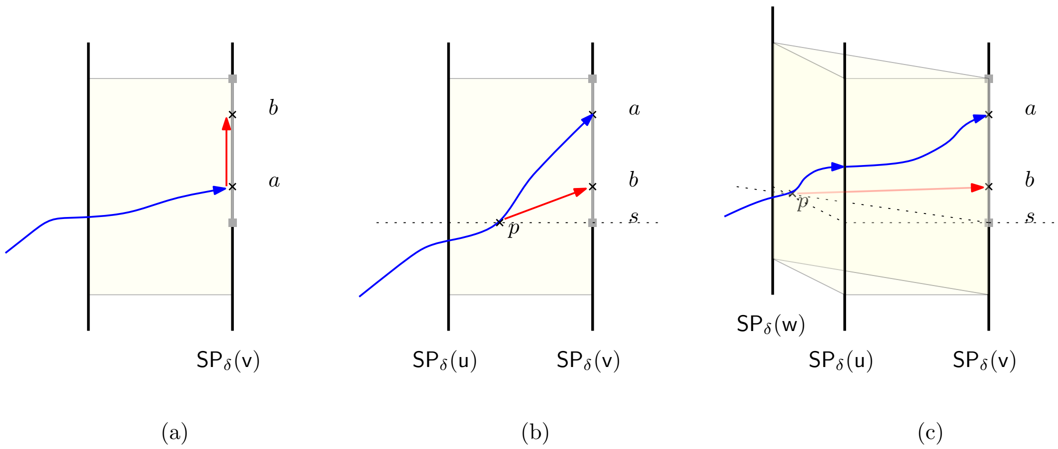

Figure 4 Illustration of the proof of property 3. a and b belong to the same elementary interval e highlighted in gray. (a) if $b > a$ (b) if $a > b$ and $p \in \mathsf{ST}_\delta(u, v)$ and (c) if $a > b$ and $p \in \mathsf{ST}_\delta(w, v)$.

Proof. 1. It follows from the definition that elementary intervals are induced by overlaying free space intervals of all spines onto each spine $\mathsf{SP}_\delta(v)$, $v \in V$. Thus clearly r and e both in $\mathsf{L}_\delta(v)$ are disjoint and the union over all elementary intervals o is the list $\mathsf{L}_\delta(v)$ that o belongs to.

2. Consider $I = [1, n]$. Each spine can split I into $2n+1$ subintervals since there exist at most n free space intervals with $2n$ endpoints in each spine. We have n spines since $|V| = n$, therefore I can be divided into $2n^2 + n$ many pieces of elementary intervals. As $\mathsf{SP}_\delta(v) \subseteq I$, therefore $|\mathsf{L}_\delta(v)| \leq 2n^2 + n$, for all $v \in V$.

3. If $x = y$ then it is trivial to see $\phi(v, x) = \phi(v, y)$. Now for the sake of contradiction, assume there exist $x, y \in e$, with $x \neq y$ such that $\phi(v, x) \neq \phi(v, y)$. Assume without loss of generality that $\phi(v, x) < \phi(v, y)$ and let $m = \phi(v, x)$. Let $\mathcal{P}(1, x)$ be a reachable path with $\mathbf{L}(\mathcal{P}(1, x)) = \phi(v, x)$. (i) If $b > a$, then extending $\mathcal{P}(1, x)$ to continue vertically from x to y in e yields a reachable path $\mathcal{P}(1, y)$ from $(1, 1)$ to y with $\mathbf{L}(\mathcal{P}(1, y)) = \mathbf{L}(\mathcal{P}(1, x))$; see Figure 4(a). Therefore $\phi(v, y) \leq \phi(v, x)$ which is a contradiction.

(ii) Now, assume that $a > b$. Let $\mathsf{SP}_\delta(u)$ be the last spine traversed by $\mathcal{P}(1, x)$ before reaching $\mathsf{SP}_\delta(v)$. That means $\mathcal{P}(1, x)$ traverses $\mathsf{ST}_\delta(u, v)$. Now let s be the start point of e. Clearly the horizontal line at s hits $\mathcal{P}(1, x)$ at some point p where $p \in \mathsf{FSD}_\delta(P, Q)$. From the definition of the elementary interval we observe that $\mathcal{P}(p, x)$ lies within free space, hence does $\mathcal{P}(p, y)$ as well. Therefore, a straight line from p to y lies within the same free space. Now if $p \in \mathsf{ST}_\delta(u, v)$ then $\mathbf{L}(\mathcal{P}(1, y)) = \mathbf{L}(\mathcal{P}(1, x))$, thus $\phi(v, y) \leq \phi(v, x)$ and contradiction. If $p \in \mathsf{ST}_\delta(w, v)$ with $w < u < v$, then $\mathbf{L}(\mathcal{P}(1, y)) < \mathbf{L}(\mathcal{P}(1, x))$, thus $\phi(v, y) < \phi(v, x)$ and again a contradiction. This completes the proof. ◀

Since by Property 3 of Lemma 2 ϕ is constant on each elementary interval, we will write $\phi(v, e)$ for each elementary interval $e \in \mathsf{L}_\delta(v)$ for all $v \in V$. Note that the domain of ϕ allows us to define ϕ not only for elementary intervals but also for reachable intervals. A subset of elementary intervals in $\mathsf{SP}_\delta(v)$ that are reachable from an elementary interval $r \in \mathsf{L}_\delta(u)$ with $u \leq v$ is called *reachable interval* in $\mathsf{L}_\delta(v)$ from r denoted by $\overline{\mathcal{I}}(v, r)$ where $\overline{\mathcal{I}}(v, r) \subseteq \mathsf{SP}_\delta(v)$. We denote ϕ for $\overline{\mathcal{I}}(v, r)$ by $\phi(v, \overline{\mathcal{I}}(v, r))$. Lemma 3 below shows a recursive formula for ϕ, which we will be used in our dynamic programming algorithm.

▶ **Lemma 3.** *1. For all elementary intervals $e \in \mathsf{L}_\delta(1)$: If e is reachable from $(1, 1)$ then $\phi(1, e) = 0$, otherwise $\phi(1, e) = \infty$.*

2. *For all $v \in V \backslash \{1\}$ and all elementary intervals $e \in \mathsf{L}_\delta(v)$: $\phi(v,e) = \min_{u \leq v} \min_{r \leq e} \phi(u,r) + 1$, where the second minimum is taken over all $r \in \mathsf{L}_\delta(u)$ with $r \leq e$ such that there is a reachable path from r to e within $\mathsf{ST}_\delta(u,v)$ with $u \leq v$.*

Proof. We use proof by induction. (1) Inductive base ($v = 1$): assume that e is reachable from $(1,1)$. Since e belongs to the first spine i.e. $\mathsf{L}_\delta(1)$, then $\mathbf{L}(\overline{\mathcal{P}}(1,e)) = 0$. Thus, $\phi(1,e) = \mathbf{L}(\overline{\mathcal{P}}(1,e)) = 0$. If e is not reachable from $(1,1)$ then there is no reachable path from $(1,1)$ to e. This means any reachable path from $(1,1)$ to (n,n) cannot pass through e and e cannot be part of the respective solution and should be disregarded. Therefore, for all $\mathcal{P}(1,e)$, $\mathbf{L}(\mathcal{P}(1,e)) = \infty$ and $\phi(1,e) = \min_{\overline{\mathcal{P}}(1,e)} \mathbf{L}(\overline{\mathcal{P}}(1,e)) = \infty$.

(2) Inductive hypothesis ($v \in V \setminus \{1\}$). Suppose all ϕ values are already computed for all $r \in \mathsf{L}$ where $\mathsf{L} = \cup_{u=1}^{v-1} \mathsf{L}_\delta(u) \cup \mathcal{I}_v$ such that $\mathcal{I}_v = \{r \mid r \in \mathsf{L}_\delta(v) \text{ and } r \leq e\}$. Now given v and $e \in \mathsf{L}_\delta(v)$, since we are looking for a simplified path P' whose vertices are going to be selected from a subsequence of the vertices in P, hence we have to use $\mathsf{L}_\delta(u)$ for all $u \leq v$. Due to the monotonicity of any reachable path from $(1,1)$ to e, we consider r that is smaller than or equal to e along I i.e. $r \leq e$. Thus $r \in \mathsf{L}$. Since we also aim for finding an elementary reachable path $\overline{\mathcal{P}}(1,e)$ whose length is equal to $\phi(v,e)$, we compute $\phi(u,r)$ that is minimum over all $r \in \mathsf{L}$ and $u \leq v$. In other words:

$$\min_{u \leq v} \min_{r \leq e} \phi(u,r).$$

Note that e must be reachable from r. Therefore this adds another link to $\phi(v,e)$ and we have:

$$\phi(v,e) = \min_{u \leq v} \min_{r \leq e} \phi(u,r) + 1$$

as claimed. ◀

Algorithm 1 shows our dynamic programming algorithm that computes ϕ using the recursive formula in Lemma 3. Our algorithm consists of three main steps: (1) *Initialization*, (2) *Propagation* and (3) *Completion*. We compute the free space surface and all elementary intervals (Initialization). We process the free space surfaces spine by spine to propagate the reachable paths up to each spine with respect to the maintenance of minimum value of ϕ (Propagation). Finally, we get the last spine and need to report those spines that already carried out reachable path of minimum ϕ value from $(1,1)$ to (n,n) (Completion).

We provide a more detailed description of the pseudocode in the following:

1. **Initialization (lines 1-5):** In line 1 of Algorithm 1, we first compute the free space surface induced by G and P. This can be done by having only strips (and spines) connecting together with respect to the adjacency of spines in G. For every spine we compute all the elementary intervals in it using ElemInterval procedure and store them into a list (line 3). We skip the description of this procedure since its computation seems obvious from the definition introduced earlier. Next in line 4, we set the cost of each elementary interval throughout the free space surface to infinity. For those elementary intervals that are reachable from $(1,1)$ we set their cost function to zero (line 5).
2. **Propagation (lines 6-15):** We process spines increasingly in V (expect the first one) since G, whose vertices are representing the spines, is a DAG and the increasing order of its vertices is given (line 6). For a $\mathsf{SP}_\delta(v)$ we compute the reachable intervals $\overline{\mathcal{I}}(v,r)$ originating from any elementary interval r in previous spines $\mathsf{SP}_\delta(u)$ for all, $u = 1, \cdots, v$ (line 10) by means of a procedure called ReachInterval. The arguments of this procedure are an elementary interval and a spine. We will discuss this procedure further with

Algorithm 1: Compute Vertex-Restricted Min–# Simplification Under $\delta_F^g(P, P') \leq \delta$

1 Compute $\mathsf{FSD}_\delta(P, G)$;
2 **for each** $v \in V$ **:**
3 $\quad \mathsf{L}_\delta(v) \leftarrow \textsc{ElemInterval}(\mathsf{FSD}_\delta(P, G), \mathsf{SP}_\delta(v))$; *// Compute elementary intervals*
4 $\quad$ **for each** $e \in \mathsf{L}_\delta(v)$ **:** $\phi(v, e) = \infty$; *// Initialize* ϕ
5 **for each** $e \in \mathsf{L}_\delta(1)$ *that is reachable from* $(1, 1)$ **:** $\phi(1, e) = 0$;
6 **for each** $v \in V - \{1\}$ **:**
7 $\quad U_v = \emptyset$;
8 $\quad$ **for each** $u \in \{1, \cdots, v\}$ **:**
9 $\quad\quad$ **for each** $r \in \mathsf{L}_\delta(u)$ **:**
10 $\quad\quad\quad \overline{\mathcal{I}}(v, r) \leftarrow \textsc{ReachInterval}(r, \mathsf{SP}_\delta(v))$; *// Compute reachable interval*
11 $\quad\quad\quad \phi(v, \overline{\mathcal{I}}(v, r)) = \phi(u, r) + 1$; *// Assign reachable interval's cost*
12 $\quad\quad\quad U_v = \overline{\mathcal{I}}(v, r) \cup U_v$; *// Take the union of all reachable intervals in* $\mathsf{SP}_\delta(v)$
13 $\quad \mathcal{L} \leftarrow \textsc{Subdivide}(U_v, \mathsf{L}_\delta(v))$; *// Subdivide* U_v *within* $\mathsf{L}_\delta(v)$ *with respect to min* ϕ *values and store the list of subdivided intervals into* $\mathcal{L}$
14 $\quad$ **for each** $e \in \mathsf{L}_\delta(v)$ **:**
15 $\quad\quad \phi(v, e) = \min_{\ell \in \mathcal{L}} \phi(v, \ell)$, where $e \in \ell$. *// Assing the min value in* $\mathcal{L}$ *to elementary intervals in* $\mathsf{L}_\delta(v)$
16 Return vertices of P' by tracing back on ϕ values.

more computational details. Next, we assign reachable interval $\overline{\mathcal{I}}(v, r)$ a cost incremented by one derived from the cost of its origin, which is an elementary interval r (line 11). We take the union of all reachable intervals originating from all elementary intervals in spines prior to $\mathsf{SP}_\delta(v)$ and store them into a set U_v (line 12). Now, we call a procedure called $\textsc{Subdivide}$ whose arguments are the union set and the current spine. The goal of this procedure is to compute a subdivision of reachable intervals in U_v across $\mathsf{SP}_\delta(v)$ in which each subdivided interval has minimum ϕ value over all ϕ values of reachable intervals in U_v that contain the subdivided interval. This provides a subdivision of min-ϕ values across the current spine stored into $\mathcal{L}$ (line 13). We can assign the cost on each subdivided interval to each elementary interval in the current spine by linearly traversing both $\mathcal{L}$ and $\mathsf{L}_\delta(v)$.

3. **Completion (line 16):** We skip explanation on this part since it appears to be clear from the context and pseudocode.

We now provide further explanations on computational side of the two procedures $\textsc{ReachInterval}$ and $\textsc{Subdivide}$.

Elementary Reachable Interval Procedure. The two arguments of this procedure are an elementary interval $r \in \mathsf{L}_\delta(u)$ and a spine $\mathsf{SP}_\delta(v)$, $u \leq v$. This procedure computes (possibly noncontinuous) interval in $\mathsf{L}_\delta(v)$ that is reachable from r within $\mathsf{ST}_\delta(u, v)$. All we need is to find start and end pointers of an reachable interval in $\mathsf{SP}_\delta(v)$. This can be done by using the algorithm proposed in Lemma 3 of [3]. We have the following lemma:

▶ **Lemma 4.** *Let u and v be two integers such that $1 \leq u < v \leq n$. One can compute all $\cup_{r \in \mathsf{L}_\delta(u)} \overline{\mathcal{I}}(v, r)$ in $O(|\mathsf{L}_\delta(u)|)$ time.*

▶ **Corollary 5.** *Let u and v be two integers such that $1 \leq u < v \leq n$. One can compute all $\cup_{r \in \mathsf{L}_\delta(u)} \overline{\mathcal{I}}(v, r)$ in parallel in $O(\sqrt{|\mathsf{L}_\delta(u)|})$ time using $O(\sqrt{|\mathsf{L}_\delta(u)|})$ processors.*

Proof. Consider lists $L_1, L_2, \cdots L_k$ of elementary intervals in $\mathsf{L}_\delta(u)$ where $k = |\mathsf{L}_\delta(u)|/s$ for some $s > 1$ and $L_i \neq L_j$ with $i \neq j$. We associate to each list L_i with $i = 1, 2, \cdots, k$, a processors that can compute $\cup_{r \in L_i} \overline{\mathcal{I}}(v, r)$ in $O(s)$ according to Lemma 4. Taking $s = \sqrt{|\mathsf{L}_\delta(u)|}$ completes the proof. ◀

Subdivide Procedure. We are given a set of reachable intervals U_v with their ϕ values. We need to efficiently compute a subdivision of reachable intervals in U_v where each subdivided interval takes the minimum value of ϕ from all subintervals in U_v which overlap the respective subdivided interval. To achieve this, we overlay all intervals in U_v vertically onto the XY-coordinate plane in such a way that their start and end points are (increasingly) encountered along Y-axis and their ϕ values are (increasingly) set along X-axis. Due to the monotonicity of reachable intervals their end and start points are sorted along $\mathsf{SP}_\delta(v)$ and correspondingly along Y-axis as well. Now all we need is to compute the lower envelope (in terms of ϕ values) over all intervals in U_v (see Figure 5). The algorithm described in Section 5 of [5] allows us to achieve this efficiently for the case where the ordering in which the start and end points of intervals are being visited along the Y-axis is given. We have the following lemma:

▶ **Lemma 6.** *Let $S = \{S_1, \cdots, S_N\}$ be a set of segments in the plane and the left-to-right order of endpoints of segments in S is given. One can compute a lower envelope of size $O(N)$ induced by segments in S in $O(N)$ time.*

▶ **Corollary 7** (Discussion in [12]). *Let $S = \{S_1, \cdots, S_N\}$ be a set of segments in the plane. The lower envelope with linear complexity can be computed by parallelization in $O(\log^2 N)$ time using $O(N/\log N)$ processors in the EREW (Exclusive Read, Exclusive Write) PRAM Model.*

Applying Lemma 6 to U_v yields a subdivision of intervals into some subdivided subintervals each with their exclusive ϕ value (see Fig 5). We immediately have the following lemma:

▶ **Lemma 8.** *For any $v \in V \setminus \{1\}$,* Subdivide $(U_v, \mathsf{L}_\delta(v))$ *computes a list $\mathcal{L}$ as a subdivision of U_v in $O(|U_v|)$ time, where $|U_v| = v \cdot \left(\max_{u=1}^{v} |\mathsf{L}_\delta(u)|\right)$ and $|\mathcal{L}| = O(|U_v|)$.*

Proof. There are v many $\mathsf{L}_\delta(u)$ for all $u = 1, \cdots, v$ on which r can be placed and $\overline{\mathcal{I}}(v, r)$ is reachable interval starting from $r \in \mathsf{L}_\delta(u)$. Each $\mathsf{L}_\delta(u)$ has $|\mathsf{L}_\delta(u)|$ many such a r as defined above. Thus, the number of reachable intervals in U_v is $|U_v| = v \cdot \left(\max_{u=1}^{v} |\mathsf{L}_\delta(u)|\right)$. Now following Lemma 6, computing the bottommost envelope of $N = |U_v|$ intervals in XY-coordinate plane resulting in subdivided intervals takes $O(|U_v|)$. Therefore the runtime follows the latter upper bound. It again follows from Lemma 6 that $\mathcal{L}$ has complexity of the number of reachable intervals in U_v which is $|\mathcal{L}| = O(|U_v|)$. ◀

▶ **Theorem 9.** *Algorithm 1 has runtime of $O(n^4)$ using $O(n^3)$ space.*

Proof. In line 1 of Algorithm 1 we compute the free space surface implicitly by computing all strips. There are $O(n^2)$ strips and each takes $O(n)$ to compute its free space diagram, hence the whole line takes $O(n^3)$. In line 3, the ElemInterval procedure takes $O(n^3)$ since we overlay all $O(n)$ free space on all spines intervals onto each spine to compute elementary intervals for one spine which takes $O(n^2)$. We have a v-loop hence ElemInterval takes $O(n^3)$. Line 4 also initialize all elementary intervals in $O(n^2)$ time since there is a r-loop on $\mathsf{L}_\delta(v)$ and also note that $|\mathsf{L}_\delta(v)| = O(n^2)$ by Lemma 2, Property (2). Line 5 obviously takes $O(n^2)$ by a similar argument made for the latter line.

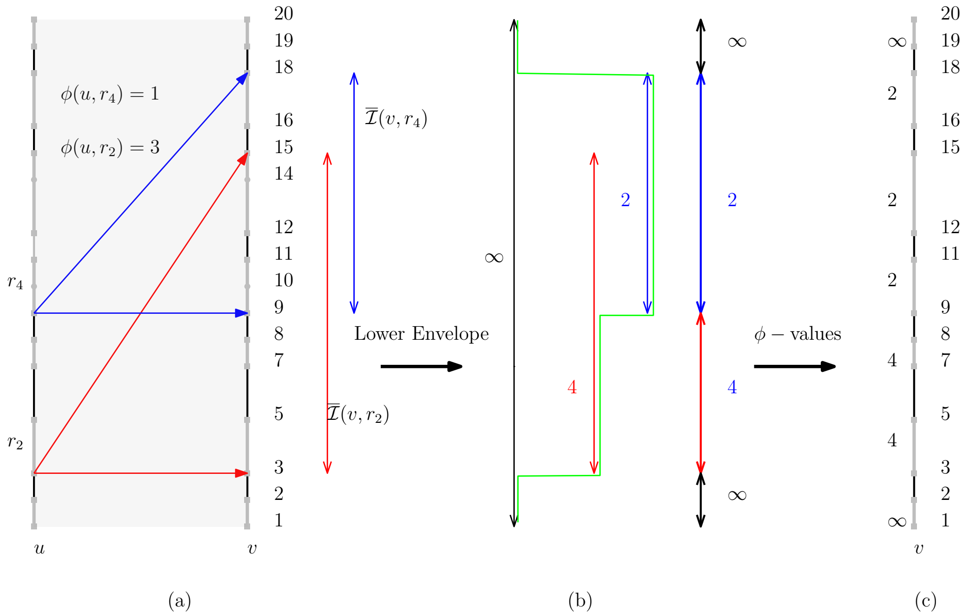

Figure 5 (a) and (b) give an illustration of subdivide procedure and (c) is the process occurred in line 15 of Algorithm 1. In this example let $\mathsf{L}_\delta(v)$ contain elementary intervals distributed within the interval $I = [1, 20]$. In (a) there are two reachable intervals $\overline{\mathcal{I}}(v, r_2)$ and $\overline{\mathcal{I}}(v, r_4)$ of ϕ values 4 and 2, respectively. In (b) the lower envelope technique results in having a subdivision of min ϕ values on $\mathsf{SP}_\delta(v)$. Here the green curve indicates the lower envelope of the intervals in terms of the ϕ values. In (c) each elementary interval in $\mathsf{L}_\delta(v)$ can take the min ϕ value obtained from (b).

Line 10 with the r-loop together takes $O(|\mathsf{L}_\delta(u)|)$ by Lemma 4 which would be $O(n^2)$ following Lemma 2. Also line 11 and line 12 with the r-loop together take $O(n^2)$. Taking the two outer u-loop and v-loop into account yields us $O(n^4)$ runtime so far.

Line 13 has SUBDIVIDE procedure that runs in $O(|U_v|)$ by Lemma 8. Since we still have an outer v-loop thus this line takes in the total order of

$$\sum_{v=1}^{n} |U_v| = \sum_{v=1}^{n} v \cdot (\max_{u=1}^{v} |\mathsf{L}_\delta(u)|) = \sum_{v=1}^{n} v \cdot n^2 = O(n^4).$$

Line 15 is computable by a linear traverse over $\mathcal{L}$ and $\mathsf{L}_\delta(v)$. Having $|\mathcal{L}| = O(|U_v|)$ by Lemma 8, $|\mathsf{L}_\delta(v)| = O(n^2)$ from Lemma 2 and an outer v-loop we get $O(n^4)$. Line 16 obviously traverse all elementary intervals in the free space surface that they have number of $O(n^3)$. Therefore Algorithm 1 has total running time of $O(n^4)$.

Note that the space is depending on the number of ϕ values that is the same as the number of elementary intervals throughout the free space surface. By Lemma 2, the total number of elementary intervals is $\sum_{v=1}^{n} |\mathsf{L}_\delta(v)| = O(n^3)$. Therefore the space required for Algorithm 1 is $O(n^3)$. ◀

▶ **Corollary 10.** *Algorithm 1 can run in $O(n^3)$ time using $O(n^4/\log n)$ space in EREW PRAM Model.*

Proof. The only difference compared to the proof of Theorem 9 is the application of Corollary 5 and Corollary 7 in line 10 and line 13 of Algorithm 1, respectively. In line 10, we can compute REACHINTERVAL in $O(\sqrt{|\mathsf{L}_\delta(u)|})$ time that is equivalent to $O(n)$ by Lemma 2, Property 2. Also using $O(n)$ processors requires $O(n)$ space per spine thus, $O(n^2)$ space in total by now. We have $O(n^2)$ strips and computing ReachInterval takes $O(n)$ per strip therefore $O(n^3)$ time so far.

In line 13, given $|U_v|$ many reachable intervals on $\mathsf{SP}_\delta(v)$ we can compute the lower envelope of them in $O(\log^2 |U_v|)$ time using $O(|U_v|/\log |U_v|)$ space. Therefore, the total runtime on Subdivide procedure is

$$\begin{aligned}\sum_{v=1}^{n} O(\log^2 |U_v|) &= \sum_{v=1}^{n} O(\log^2 (v \cdot (\max_{u=1}^{v} |\mathsf{L}_\delta(u)|)) \\ &= \sum_{v=1}^{n} O(\log^2 (v.n^2)) = \sum_{v=1}^{n} O\big(\log^2 v + \log^2 n + 2\log v \log n\big) \\ &= \sum_{v=1}^{n} O(\log^2 v) + \sum_{v=1}^{n} O(\log^2 n) + 2\sum_{v=1}^{n} O(\log v \log n) \\ &= O(n \log^2 n).\end{aligned}$$

Since both the initialization part of the algorithm and ReachInterval take $O(n^3)$ in any ways, hence the formers dominate the latter and therefore the total runtime is $O(n^3)$.

The total space required for the algorithm increases since we use $O(|U_v|/\log |U_v|)$ processors to handle Subdivide procedure, therefore the total space used by this procedure is:

$$\sum_{v=1}^{n} O(|U_v|/\log |U_v|) = \sum_{v=1}^{n} O(vn^2/\log(vn^2)) = O(n^4/\log n),$$

as claimed. ◀

▶ **Theorem 11** (Vertex-Restricted Fréchet Distance). *Let P be a polygonal curve with n vertices and δ be a positive real value. One can compute vertex-restricted min–# simplification problem under global Fréchet distance in $O(n^4)$ time using $O(n^3)$ space. In the case that EREW PRAM Model is allowed, one can compute the problem in $O(n^3)$ time using $O(n^4/\log n)$ space.*

4 Curve-Restricted Min–# Under The Directed Hausdorff $P' \to P$

4.1 Preliminaries

The directed Hausdorff distance between two curves is defined as

$$\overrightarrow{\delta_H^g}(P, Q) = \max_{1\le i\le n} \min_{1\le j\le n} ||P(i) - Q(j)||$$

The Subset Sum problem is an NP-hard problem where we are given a set of integers A and an integer B. The goal is to find a set $A' \subseteq A$ that sums to exactly B.

In the next subsection we look at the decision variant of Curve-Restricted Min–#, meaning that given a curve P, a real value δ, a distance metric $\mathsf{D}(\cdot,\cdot)$, and an integer k we want to determine if there exists a simplification P' of at most k links such that $\mathsf{D}(P, P') \le \delta$. The proof in this section draws inspiration from the NP-hardness proof for the min–# path problem on polyhedral surfaces given by Kostitsyna et al. [14]. However, it does not trivially follow from that proof. Instead of being able to directly construct a polyhedral surface with the holes and gates we want, we must now construct a curve such that the set of points with a distance to P greater than δ has the desired shape. This also makes it impossible to use vertical fences like in [14].

4.2 NP-hardness

▶ **Theorem 12.** *Let $U = (A = \{a_1, \ldots, a_n\}, B)$ with $a_1, \ldots, a_n, B \in \mathbb{N}$ be an instance of the* Subset Sum *problem. It can then be transformed in polynomial time to instance $U' = (P, \delta, \overrightarrow{\delta_H^g}(P', P), k)$ of the* Curve-Restricted Min-# *problem so that the curve P can be simplified to a polygonal curve P' from p_1 to p_n of k links and $\overrightarrow{\delta_H^g}(P', P) \leq \delta$ iff there is a set $A' \subseteq A$ where $\sum_{a_i \in A'} a_i = B$.*

Our transformation works as follows: We set δ to be equal to $\max_{a_i \in A} a_i$. Let γ then be an arbitrarily small constant such that $0 < \gamma \ll \delta$. We set k to be $2n - 1$. Without loss of generality we require our set A to be sorted in such a way that $0.5a_n < B$. We then create four sets of subcurves $M = \{m^0, \ldots, m^{n-1}\}, R = \{r^0, \ldots, r^{n-1}\}, F = \{f^1, \ldots, f^{n-1}\}, L = \{l^1, \ldots, l^{n-1}\}$. There is also one additional subcurve t. We refer to the vertices defining the curves by subscript, e.g. the first vertex of curve m^0 is m^0_0. The curves are defined by the description given in Appendix A. Important to note is that the subcurves in sets R, L as well as t between t_2 and t_9 are almost completely horizontal except for some protrusions on them which we call *γ-spikes* that have the curve move a vertical distance of 0.5γ from the rest of the curve and then immediately back again to continue in a horizontal manner. The x-coordinates of half of these spikes are associated with the integers in A, the other half have an x-coordinate close to 0.

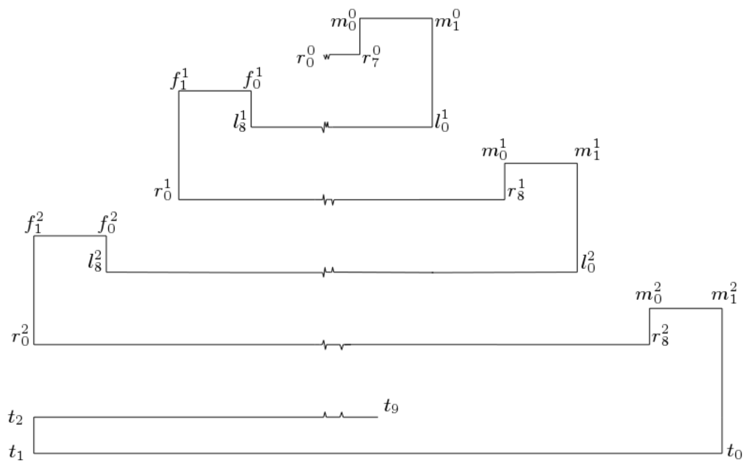

Figure 6 Curve P that is created by transforming an instance U with $A = \{1, 2, 4\}, B = 6$. The sizes of the γ-spikes created by points $\{r^0_2, r^0_5, t_4, t_7\} \cup \{r^i_2, r^i_3, r^i_6, l^i_2, l^i_5, l^i_6 | 0 < i \leq n-1\}$ have been exaggerated for clarity.

The subcurves are concatenated in the cyclical order $r \to m \to l \to f$ and eventually appended by t to form our curve P.
So the complete curve P is equal to:

$$\{r^0_0, \ldots, r^0_7, m^0_0, m^0_1, l^1_0, \ldots, l^1_8, f^1_0, f^1_1, r^1_0, \ldots, m^{n-1}_1, t_0, \ldots, t_9\}$$

See Figure 6 for an example.

Given P and the allowed deviation δ the following lemma is true:

▶ **Lemma 13.** *Any curve P' that simplifies P and has at most $2n-1$ links has exactly one vertex on each subcurve $r \in R$, one vertex on each subcurve $l \in L$, one vertex on t, and no other vertices.*

Proof. Let $N_\delta(P)$ be the *δ-neighborhood* of P, defined as

$$N_\delta(P) = \{(x,y) | \overrightarrow{d_H^g}((x,y),P) \leq \delta\}$$

The constraint that our simplified curve P' must have $\overrightarrow{d_H^g}(P',P) \leq \delta$ means that every link of P' must be completely contained in $N_\delta(P)$. Inversely, all points that are further than δ from their closest point in P form an *implicit obstacle* O, defined

$$O = \{(x,y) | (x,y) \notin N_\delta(P) = \{(x,y) | \overrightarrow{d_H^g}((x,y),P) > \delta\}\}$$

No part of P' can intersect O. O is shown in Figure 7.

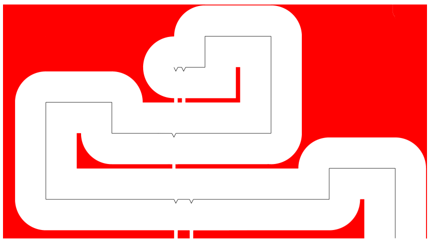

Figure 7 Implicit obstacle of a section of a transformation curve P shown in red. The size of the downward pointing γ-spikes and the holes in the obstacle they induce have been exaggerated for clarity, the upward pointing γ-spikes are not depicted. The leftmost holes alternate between having x-coordinate 0.5γ and 0 and so are not naturally aligned.

The curve P has been carefully constructed to shape O for our purposes. P's subcurves from sets R and L have vertical distance $2\delta + \gamma$ between them, meaning the δ-neighborhood around those sections only has a gap of thickness γ between them. The γ-spikes make it so that there are holes in this gap at the x-coordinate of the spikes that allow P' to shortcut from a point on any l^i to r^i and from any r^i to l^{i+1}.

If we consider the subcurves in $\{R, L\}$ and t between t_2 to t_9 to be *vertical levels* it is clear from Figure 7 that no link of P' can traverse more than two vertical levels without intersecting O. If we want our simplification to traverse downwards we either need to pass through one of the holes induced by γ-spikes or go around the horizontal end of the vertical level, which takes at least 2 links. Furthermore, on each vertical level there are only 2 specific points from where we would be able to align the next two holes in order to traverse two vertical levels with one link (See Figure 8). All other links can traverse one hole, and thus one vertical level, at most. But, as we will show below, skipping a level by aligning holes will make it impossible for P' to reach t_9 in $2n-1$ links or less. Since we have $2n$ vertical levels and we cannot skip any, it is clear that the only way for a simplification to have $2n-1$ links is to have exactly one vertex on each vertical level, meaning that each of our links needs to pass through exactly one of the induced holes in O.

The reason skipping vertical levels will never result in a simplification that reaches t_9 in $2n-1$ links is that since skipping still needs 2 links per 2 vertical levels, a simplification can only reach t_9 in $2n-1$ links if it has only one vertex on t. So the final link of P' needs to pass through a hole between r^{n-1} and t and then end in t_9. If we look at the x-coordinates

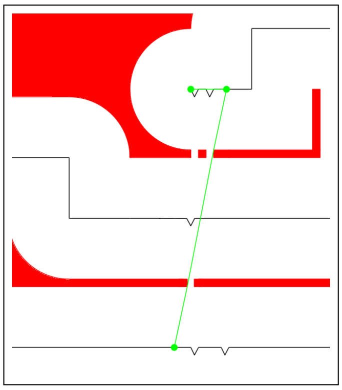

Figure 8 By first moving to a specific position so the next two holes are aligned, it is possible to skip a vertical level. This takes the same number of links as traversing the holes one at a time.

from which we can align the holes and skip a level it becomes clear that once we have used a skip this becomes impossible. To skip any level , we must first move to a vertex to align the next two holes. We will call this vertex a *skip vertex*. To skip level l^i,we need to give our skip vertex either x-coordinate 0.75γ or $0.75\gamma + 0.75a_i$. If we want to skip level r^i we must give it x-coordinate -0.25γ or $-0.25\gamma - 0.25a_{i+1}$. (For details on how these x-coordinates are computed, see Appendix B.) After the skip, our new vertex has x-coordinate $-0.25\gamma, -0.25\gamma - 0.25a_i, 0.75\gamma$, or $0.75\gamma + 0.75a_{i+1}$ respectively. Without using a link to readjust, this fractional γ-component of the x-coordinate will propagate downwards with every new link. Since using a link to readjust would require our simplification to have more than $2n - 1$ links and since t_9's x-coordinate has an integer γ-component, no minimum-link simplification can use a skip. ◀

Let $C_\mu(u)$ be the set of points on subcurve u that are reachable from the starting point of P in at most μ links without skipping vertical levels. The following lemma is then also true:

▶ **Lemma 14.** *The x-coordinates of the points in $C_{2i-1}(l^i)$ encode all possible subsets of $A_i = \{a_1, a_2, \ldots, a_{i-1}, a_i\}$.*

Proof. Because of Lemma 13, we know that any path to a point in $C_{2i-1}(l^i)$ passes through an γ-spike-induced hole with every link. Because the holes are vertically precisely in the middle of two vertical levels, we have for each path vertex v_j on any minimum link path that

$$v_{j_x} = h_x + [h_x - v_{j-1_x}]$$

where v_{j_x} is the x-coordinate of vertex v_j and h_x is the x-coordinate of the hole that the link passes through to reach v_j. (which has an identical x-coordinate to the γ-spike that induces it.) For each point $c \in C_{2i-1}(l^i)$ reachable this way the x-coordinate is equal to $i\gamma + \sum_{a \in A_c} a$, where $A_c \subseteq A$ are the integers associated with the holes the path went though to reach c. The proof of this follows by induction:
(1) Consider our base case where we want to make our first link between the starting point $(0, 0)$ and a point on l^1. Finding the ways we can do this is equivalent to finding the set $C_1(l^1)$. There are two possible holes to go through, with x-coordinates equal to 0.5γ and $0.5a_1 + 0.5\gamma$. Of this second hole we say it is *associated* with a_1. The two points on l^1 we can reach by cutting through of these two holes have x-coordinates γ and $a_i + \gamma$, respectively.

We see that the invariant holds, the first point encoding the empty set, and the second point encoding the set $\{a_1\}$.

(2) Now consider the induction step: We have a set $C_{2i-1}(l^i)$ of points encoding all possible subsets of A_i by having the x-coordinate of each point $c \in C_{2i-1}(l^i)$ be equal to $i\gamma + \sum_{a \in A_c} a$. There is only one hole between l^i and the next vertical level r^i. So the set $C_{2i}(r^i)$ only contains the points gotten by cutting from each c through the hole, which has x-coordinate 0. This means that for each c there is a point $c' \in C_{2i}(r^i)$ with x-coordinate $-i\gamma - \sum_{a \in A_c} a$. There are then two possible holes that can be taken to get to a point on the next level l^{i+1} (or t if we have reached the end of P). The holes have x-coordinates 0.5γ and $0.5a_{i+1} + 0.5\gamma$. For each point in $C_{2i}(r^i)$ there are then two reachable points in $C_{2(i+1)-1}(l^{i+1})$. Because $v_{j_x} = h_x + [h_x - v_{j-1_x}]$, their x-coordinates are $(i+1)\gamma + \sum_{a \in A_c} a$ and $(i+1)\gamma + a_{i+1} + \sum_{a \in A_c} a$ respectively. Because our set $C_{2i-1}(l^i)$ encoded all possible subsets up until then and for each point there are two new points in $C_{2(i+1)-1}(l^{i+1})$ for both the case where the next variable is added to the subset and the case where it isn't, the set $C_{2(i+1)-1}(l^{i+1})$ encodes all new possible subsets. So our invariant holds. (For a visualization, see Figure 9) ◀

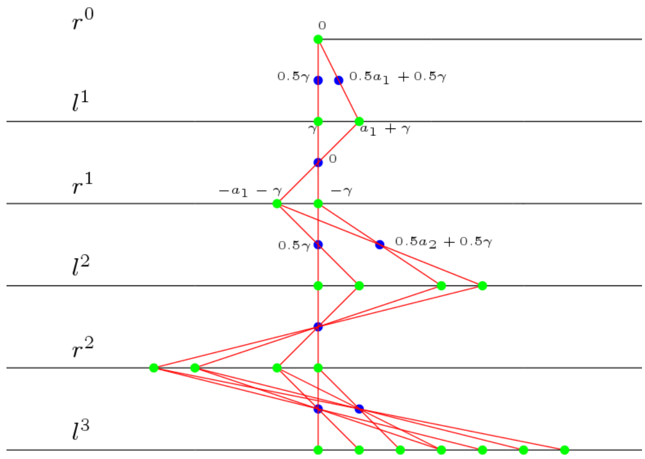

Figure 9 Example of all possible paths between r^0 and l^3. Holes in the implicit obstacle are marked in blue. For some points the x-coordinate is given. The x-coordinates of the bottommost points are, from left to right: $3\gamma, a_1 + 3\gamma, a_3 + 3\gamma, a_1 + a_3 + 3\gamma = a_2 + 3\gamma, a_1 + a_2 + 3\gamma, a_2 + a_3 + 3\gamma, a_1 + a_2 + a_3 + 3\gamma$

Given Lemma 13 and Lemma 14 we know that all points on t that are reachable with $2n-1$ links without skips encode a possible subset of A. Given that the x-coordinate of the endpoint of P is on t and has an x-coordinate equal to $B + n\gamma$ we know that if the endpoint is reachable in $2n-1$ links there must be a subset of A that sums to B, solving our instance of Subset Sum.

Seeing as our transformation only requires a linear number of subcurves, and each subcurve has a constant number of vertices, our transformation is clearly polynomial. This means that Curve Restricted Min–# using $\overrightarrow{\delta^g_H}(P', P)$as a distance metric is NP-hard.

4.3 Extension to non-degenerate curves

To keep the proof simple our constructed curve P is degenerate seeing as the distance between each γ-spike and the next curve is exactly 2δ, giving 0-width holes. The proof can be easily adjusted to use a non-degenerate curve however: Aside from γ, we introduce another small constant ζ where $0 < \zeta \ll \gamma \ll \delta$. We then lower the tip of each downward pointing spike by ζ so the holes have a positive, if extremely small, width. The reachable points on each l^i are now intervals, but they are small and spread out enough that we can still associate each interval with a subset of A.

5 Non-Restricted Min–# Under $\delta_F^g(P, P')$ in $\mathbb{R}^d$

In this section we present a simple approximation algorithm for non-restricted variant of min–# that discretizes the feasible space in which an optimal simplified curve can be placed. As we mentioned earlier Guibas et al. [11] proposed an $O(n^2 \log^2 n)$-time algorithm that solves non-restricted simplification under $\delta_F^g(P, P')$ in the plane. In this section we show our algorithm runs slightly faster in any arbitrary dimension d. The shortcut graph approach proposed by Imai and Iri [13] is somehow concealed in the spirit of our algorithm. More precisely, we compute a set of polynomially many links in the discretized space and we validate whether each link is part of the graph or not by checking its Fréchet distance to some subcurve of P. To speed up the validation process, we use a data structure to decide whether the Fréchet distance between a link and a subcurve of P is at most δ or not. We incrementally add such links to our graph G until all of the links in discretized space are processed. Once construction of G is completed, we compute the shortest path in G and return the links in P'. For a better understanding of our pseudocode, here we define some notations. Consider a ball $B(o, r)$ of radius $r > 0$ centered at $o \in \mathbb{R}^d$. A partitioning of the space into a set of disjoint cells of side length l by a set of axis parallel hyperplanes is denoted by $\mathsf{Prt}(\mathbb{R}^d, l)$. A discretization of $B(o, r)$ into a set $\mathcal{G}_o$ of cells of side length l is an intersection between $\mathsf{Prt}(\mathbb{R}^d, l)$ and $B(o, r)$ denoted by $\mathsf{Prt}(B(o, r), l)$. For any $p_i \in P$ we denote $\mathcal{G}_{p_i}$ as $\mathcal{G}_i$ for brevity.

As we can see Algorithm 2 is a straightforward computation of valid shortcuts and shortest path in the shortcut graph G. The Validate procedure takes as arguments a segment $\langle c_1 c_2 \rangle$ and a subcurve $P[i, j]$ and its task is to (approximately) decide whether $\delta_F(\langle c_1 c_2 \rangle, P[i, j]) \leq \delta$ or not, more precisely it returns 'true' if $\delta_F(\langle c_1 c_2 \rangle, P[i, j]) \leq (1 + \epsilon)\delta$ and 'false' otherwise. we efficiently implement the Validate procedure by (lines 9, 11, 13) by means of the data structure in [9]. Here we slightly rephrase their lemma according to our terminology:

▶ **Lemma 15** (Theorem 5.9 in [9]). *Let P be a polygonal curve in $\mathbb{R}^d$ with n vertices. One can construct a data structure of size $O\big((\epsilon^{-d} \log^2(1/\epsilon))n\big)$ and construction time of $O\big((\epsilon^{-d} \log^2(1/\epsilon))n \log^2 n\big)$, such that for any query segment $\langle ab \rangle$ and two vertices p_i and p_j in P, one can compute a $(1 + \epsilon)$-approximation of $\delta_F(\langle ab \rangle, P[i, j])$ in $O(\epsilon^{-2} \log n \log \log n)$ query time.*

Now we have the following lemma:

▶ **Lemma 16.** *Let $0 < \epsilon \leq 1$ and let $\langle ab \rangle$ be a segment in $\mathbb{R}^d$ such that a and b are confined inside two cells $h' \in \mathcal{G}_i$ and $h'' \in \mathcal{G}_j$, respectively. If $\delta_F(\langle ab \rangle, P[i, j]) \leq \delta$, then for all corners $c' \in h'$ and $c'' \in h''$ Validate$(\langle c' c'' \rangle, P[i, j])$ returns 'true'.*

Algorithm 2: Compute Non-Restricted Min–# Simplification Under $\delta_F^g(P,P') \le \delta$

forall $i \in \{1, \cdots, n\}$ **do**
  Compute $\mathsf{Prt}\big(B(p_i,\delta), \epsilon\delta/2\sqrt{d}\big)$ and $\mathcal{G}_i$.
$E \leftarrow \emptyset$;
$V \leftarrow \emptyset$;
forall $i \in \{2, \cdots, n-2\}$ **and** $j \in \{i+1, \cdots n-1\}$ **do**
  forall $h_1 \in \mathcal{G}_i$ **and** $h_2 \in \mathcal{G}_j$ **do**
    forall *corner* $c_1 \in h_1$ **and** *corner* $c_2 \in h_2$ **do**
      if $\texttt{VALIDATE}(\langle c_1c_2\rangle, P[i,j]) = \mathit{true}$ **then**
        $E \leftarrow E \cup \langle c_1c_2\rangle$ **and** $V \leftarrow V \cup \{c_1, c_2\}$
      if $\texttt{VALIDATE}(\langle p_1c_1\rangle, P[1,i]) = \mathit{true}$ **then**
        $E \leftarrow E \cup \langle p_1c_1\rangle$ **and** $V \leftarrow V \cup \{p_1, c_1\}$
      if $\texttt{VALIDATE}(\langle c_2p_n\rangle, P[j,n]) = \mathit{true}$ **then**
        $E \leftarrow E \cup \langle c_2p_n\rangle$ **and** $V \leftarrow V \cup \{c_2, p_n\}$
Return the shortest path in $G = (V,E)$.

Proof. Let c' be an arbitrary corner of h' and c'' an arbitrary corner of h''. Note that $\mathsf{Diam}(h') = \mathsf{Diam}(h'') = \sqrt{d} \cdot (\epsilon\delta/2\sqrt{d}) = \epsilon\delta/2$, where $\mathsf{Diam}(h')$ and $\mathsf{Diam}(h'')$ are the diameter of cells h' and h'', respectively. Hence $\ell_1 = \|a - c'\| \le (\epsilon/2)\delta$ and $\ell_2 = \|b - c''\| \le (\epsilon/2)\delta$. Given the two segments $\langle ab\rangle$ and $\langle c'c''\rangle$ the Fréchet distance between them is $\delta_F(\langle c'c''\rangle, \langle ab\rangle) = \max\{\ell_1, \ell_2\} = (\epsilon/2)\delta$ by [4]. Now by applying a triangle inequality between the segments and path $P[i,j]$ we have: $\delta_F(\langle c'c''\rangle, P[i,j]) \le \delta_F(\langle ab\rangle, P[i,j]) + \delta_F(\langle c'c''\rangle, \langle ab\rangle) \le \delta + \epsilon\delta/2 = (1+\epsilon/2)\delta$. We build the data structure of Lemma 15 for $\langle c'c''\rangle$ with respect to parameter $\epsilon/3$ that yields a $(1+\epsilon/3)$-approximation of the Fréchet distance between $\langle c'c''\rangle$ and $P[i,j]$ denoted by D. We have:

$$D \le (1+\epsilon/3)\delta_F(\langle c'c''\rangle, P[i,j]) \le (1+\epsilon/3)(1+\epsilon/2)\delta = (1+5\epsilon/6+\epsilon^2/6)\delta \le (1+\epsilon)\delta,$$

for $\epsilon < 1$. Therefore, for all corners $c' \in h'$ and $c'' \in h''$ such that $\texttt{VALIDATE}(\langle c'c''\rangle, P[i,j])$ returns 'true' as claimed. ◀

▶ **Lemma 17.** *Let $\delta > 0$ be a real number. For any real number $0 < \varepsilon \le 1$, Algorithm 2 returns a simplified curve P' such that $\delta_F^\ell(P,P') \le (1+\epsilon)\delta$ and P' has complexity of at most $2m+1$ where m is the number of links in optimal simplified curve P^* under $\delta_F^g(P,P^*) \le \delta$.*

Proof. Let $\mathcal{P}$ be the shortest path in G returned by the algorithm. It follows from Lemma 16 that for every edge $e = c_1c_2$ in $\mathcal{P}$, there exist a pair (i,j) with $1 \le i < j \le n$ such that $\texttt{VALIDATE}(e, P_e)$ returns 'true', i.e., $\delta_F(e, P_e) \le (1+\epsilon)\delta$, where $c_1 \in B(p_i,\delta)$, $c_2 \in B(p_j,\delta)$ and $P_e = P[i,j]$. Therefore,

$$\delta_F^\ell(\mathcal{P}, P) = \max_{e\in\mathcal{P}}\{\delta_F(P_e, e)\} \le (1+\epsilon)\delta.$$

Now let m be the minimum number of links in P^* such that $\delta_F^g(P,P^*) \le \delta$. Let (σ,θ) be a Fréchet matching realizing $\delta_F^g(P,P^*) \le \delta$. Let t_k and t_{k+1} be two real values such that $0 \le t_k < t_{k+1} \le 1$ and consider a fixed link $\langle l_kl_{k+1}\rangle$ in P^* where $l_k = P^*(\theta(t_k))$ and $l_{k+1} = P^*(\theta(t_{k+1}))$. Let p_i and p_j be the first and last vertices along $P[x,y]$, respectively where $i < j$, $x = P(\sigma(t_k))$ and $y = P(\sigma(t_{k+1}))$. We know that there exists some t_i with $t_k \leq t_i$ such that $p_i = P(\sigma(t_i))$ and $a = P^*(\theta(t_i))$. This means $\|p_i - a\| \leq \delta$ and $a \in B(p_i, \delta)$, thus $a \in \mathcal{G}_i$. We denote the subcurve $P^*[\theta(t_i), \theta(t_j)]$ by C_k. Using similar argument we can see $b = P^*(\theta(t_j))$ for some $t_j > t_i$ and $b \in \mathcal{G}_j$. By applying Lemma 16 to the subsegment $\langle ab \rangle$ and $P[i, j]$, there are two corners p and q of some hypercells $h_i \in \mathcal{G}_i$ and $h_j \in \mathcal{G}_j$, respectively near a and b such that $\textsc{Validate}(\langle pq \rangle, P[i, j])$ returns 'true', i.e. $\delta_F(\langle pq \rangle, P[i, j]) \leq (1 + \epsilon)\delta$.

Now let t_{i-1} and t_{j+1} be two values such that $p_{i-1} = P(\sigma(t_{i-1}))$, $a' = P^*(\theta(t_{i-1}))$ and $p_{j+1} = P(\sigma(t_{j+1}))$, $b' = P^*(\theta(t_{j+1}))$. We denote the subcurve $P^*[\theta(t_{i-1}), \theta(i)]$ which is to the left of C_k by L_k. We similarly denote subcurve $P^*[\theta(t_j), \theta(j + 1)]$ which is to the right of C_k by R_k. Now observe that since L_k lies entirely within the cylinder of width δ around segment $P[i - 1, i]$, thus the shortcut $\langle a'a \rangle$ connecting the two endpoints of L_k entirely lies within the cylinder defined. Similarly the shortcut $\langle bb' \rangle$ connecting the endpoints of R_k lies within the cylinder defined with respect to $P[j, j+1]$. Note that $\delta_F(\langle a'a \rangle, P[i-1, i]) \leq \delta$ and $\delta_F(\langle bb' \rangle, P[j, j+1]) \leq \delta$, hence by another using of Lemma 16, two corner s of some hypercell $h_{i-1} \in \mathcal{G}_{i-1}$ and t of another hypercell $h_{j+1} \in \mathcal{G}_{j+1}$ such that $\textsc{Validate}(\langle sp \rangle, P[i - 1, i])$ and $\textsc{Validate}(\langle qt \rangle, P[j, j + 1])$ return 'true'. Now Algorithm 2 computes $\langle sp \rangle$, $\langle pq \rangle$ and $\langle qt \rangle$ in P' instead of L_K, C_k and R_k in P^*. Observe that the number of links in P^* covered by $|C_k|$ is smaller than 1, i.e., $|C_k| \leq (k + 1) - k = 1$ for all $k \in \{1, \cdots, m\}$. Also realize that $|L_k| = |R_{k-1}|$ for all $k \in \{2, \cdots, m\}$. The number of links in P^* is equal to $m = \sum_{k=1}^{m} ((k + 1) - k) > \sum_{k=1}^{m} |C_k|$. Therefore, the number of links in P' is equal to:

$$\begin{aligned}\sum_{k=1}^{m} (|L_k| + |C_k| + |R_k|) &= |L_1| + |R_m| + \sum_{k=1}^{m} |C_k| + \sum_{k=1}^{m-1} |R_k| \\ &< |L_1| + |R_m| + m + \sum_{k=1}^{m} |R_k|\end{aligned}$$

Now since, P^* has $m + 1$ vertices and every R_k covers exactly one vertex in P^* we have:

$$\begin{aligned}|L_1| + |R_m| + m + \sum_{k=1}^{m} |R_k| &= 1 + 1 + m + m - 1 \\ &= 2m + 1\end{aligned}$$

as claimed. ◀

▶ **Lemma 18.** *Algorithm 2 has runtime of* $O\Big(\epsilon^{-d} n \log n \big(\log^2(1/\epsilon) \log n + \epsilon^{-(d+2)} n \log\log n\big)\Big)$.

Proof. The number of hypercells in each ball is bounded by $(\delta/(\epsilon\delta/\sqrt{d}))^d = O(\epsilon^{-d})$. There are n^2 possible valid shortcuts $\overline{pq}$ as the corners of some hypercells in G_i and some other G_j. Hence, we have $O(\epsilon^{-2d} n^2)$ shortcuts for validation process that whether $\delta_F(\overline{pq}, P[i, j]) \leq (1 + \epsilon)\delta$ or not. On the other hand by Lemma 15 we speed up validate procedure. The construction time takes $O\big((\epsilon^{-d} \log^2(1/\epsilon)) n \log^2 n\big)$ and its query takes $O(\epsilon^{-2} \log\log\log n)$ per shortcut. Having $O(\epsilon^{-2d} n^2)$ shortcuts, the whole validation process takes $O((\epsilon^{-2d+2} n^2 \log n \log\log n))$. Therefore the total runtime would be the sum of construction time of the data structure of Lemma 15 and the latter one which is: $O\Big(\epsilon^{-d} n \log n \big(\log^2(1/\epsilon) \log n + \epsilon^{d+2} n \log\log n\big)\Big)$ as claimed. ◀

We summarize this section with the following theorem:

▶ **Theorem 19.** *Let P be a polygonal curve with n vertices in $\mathbb{R}^d$ and $\delta > 0$. For any $0 < \epsilon \leq 1$, Algorithm 2 computes 'Non-Restricted Min-#' in* $O\Big(\epsilon^{-d} n \log n \big(\log^2(1/\epsilon) \log n +$

$\epsilon^{-(d+2)} n \log\log n)\big)$ *time which returns a simplified curve* P' *that has the length at most* $3m$ *under* $\delta_F^{\ell}(P, P') \leq (1+\varepsilon)\delta$*, where* $2m+1$ *is the number of links in optimal simplified curve* P^* *such that* $\delta_F^{g}(P, P^*) \leq \delta$*.*

5.1 Curve Restricted problem Under $\delta_F^g(P, P')$ in $\mathbb{R}^1$

We consider the curve restricted min–# problem in $\mathcal{R}^1$ under the global Fréchet distance . We propose a greedy algorithm using the man-dog terminology as follows:

1. If the man and dog are staying at the same point, then let the dog go until the leash length gets δ.
2. Let the dog drag the man by the leash of length δ. If the leash gets cut off, then move the man to the dog's position and go to 1. Otherwise, let man and dog continue walking in the same pace along P by the leash of length δ. If a shorter leash is sufficient, then let the man stay and let the dog go until the length of leash get δ and repeat 2.
3. Once the dog gets p_n, let the man get p_n. Return the man's walking as P'.

5.2 Vertex Restricted min–# Under $\delta_H^g(P', P)$

We thicken the edges of P by amount δ. This induces a simple polygon $\mathcal{P}$ with (possibly) $h = O(n)$ holes inside it. Now all we need is to decide whether each shortcut (p_i, p_j) $1 \leq i < j \leq n$ lies completely within that polygon $\mathcal{P}$ or not. For this we proprocess P into a data structure such that for any straight line query ray ρ originated from some point inside the $\mathcal{P}$, compute the first point on the boundary of $\mathcal{P}$ hit by ρ. We can use the data structure proposed by [8] of size $O(n+h^2)$ and construction time $O(n + h^2 \text{polylog} h)$ which answers a query in $O(\log n)$ query time. We have n^2 possible shortcuts (p_i, p_j) and need to examine whether each shortcut lies inside $\mathcal{P}$ or not in order to store them in the edge set of G_δ graph. We originate a ray at p_i and compute a hit point x on the boundary of $\mathcal{P}$ hit by the ray in $O(\log n)$ query time. All we need is to compare the length of the ray to the length of the shortcut. If $\|p_i - x\| \geq \|p_i - p_j\|$ then the shortcut lies inside $\mathcal{P}$, otherwise it does not. Once edge set of G_δ constructed, we compute the shortest path on it. As a result we have the following theorem.

▶ **Theorem 20.** *Let* $\delta > 0$ *and let* P *be a curve with* n *vertices. One can compute vertex-restricted min–# simplification under* $\overrightarrow{\delta_H^g}(P', P)$ *in* $O(n^2 \text{polylog} n)$ *time using* $O(n^2)$ *space.*

6 Discussion

In this paper we considered three different variants of the min–# curve simplification problem; vertex-restricted, curve-restricted and non-restricted. For vertex restricted under global Fréchet distance we gave $O(n^4)$-time dynamic programming algorithm that uses $O(n^3)$ space. We proved that curve-restricted min–# simplification under global directed Hausdorff from P' to P is NP-hard. We also gave a simple approximation algorithm for non-restricted case under global Fréchet distance which runs slightly faster than the one in [11] in any arbitrary dimension that guarantees an upper bound on the number of links in the simplified curve returned by the algorithm. We also presented a linear time greedy algorithm for the curve restricted version under global Fréchet in 1D. Solving this variant in dimensions higher than one seems to be hard, and we leave further consideration of this problem including a possible NP-hardness proof to future work.

References

1 P.K. Agarwal, S. Har-Peled, N. Mustafa, and Y. Wang. Near linear time approximation algorithm for curve simplification. *Algorithmica*, 42(3–4):203–219, 2005.

2 P.K. Agarwal and K.R. Varadarajan. Efficient algorithms for approximating polygonal chains. *Discrete & Computational Geometry*, 23(2):273–291, 2000.

3 H. Alt, A. Efrat, G. Rote, and C. Wenk. Matching planar maps. *Journal of Algorithms*, 49(2):262—283, 2003.

4 H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry and Applications*, 5(1–2):75–91, 1995.

5 A. Asano, T. Asano, L. Guibas, J. Hershberger, and H. Imai. Visibility of disjoint polygons. *Algorithmica*, 1(1–4):49–63, 1985.

6 S. Bereg, M. Jiang, W. Wang, B. Yang, and B. Zhu. Simplifying 3d polygonal chains under the discrete Fréchet distance. volume 4957, pages 630–641. LATIN, 2008.

7 S. Chan and F. Chin. Approximation of polygonal curves with minimum number of line segments or minimum error. *Intl. J. Computational Geoemtry and Applications*, 6(1):59–77, 1996.

8 D.Z. Chen and H. Wang. Visibility and ray shooting queries in polygonal domains. *Computational Geometry: Theory and Applications*, 48(2):31–41, 2015.

9 A. Driemel and S. Har-Peled. Jaywalking your dog: computing the Fréchet distance with shortcuts. *SIAM Journal on Computing*, 42:1830–1866, 2013.

10 M. Godau. A natural metric for curves — computing the distance for polygonal chains and approximation algorithms. volume 480, pages 127–136. STACS, 1991.

11 L. Guibas, J. Hershberger, J. Mitchell, and J. Snoeyink. Approximating polygons and subdivisions with minimum-link paths. *Intl. J. Computational Geometry and Applications*, 3(4):383–415, 1993.

12 J. Hershberger. Finding the upper envelope of n line segments in $O(n \log n)$ time. *Information Processing Letters*, 33(4):169–174, 1989.

13 H. Imai and M. Iri. An optimal algorithm for approximating a piecewise linear function. *Journal of Information Processing*, 9(3):159–162, 1986.

14 I. Kostitsyna, M. Löffler, V. Polishchuk, and F. Staals. On the complexity of minimum-link path problems. *Journal of Computational Geometry*, 8(2):80–108, 2017.

15 M. Kreveld, M. Löffler, and L. Wiratma. On optimal polyline simplification using the hausdorff and Fréchet distance. In *34th International Symposium on Computational Geometry (SoCG 2018)*, volume 56, pages 1–14, 2018.

A Definition of the subcurves of P

The parameter ζ is set to 0 in the normal proof. For the extension shown in Section 4.3, ζ is a real value such that $0 < \zeta \ll \gamma \ll \delta$.

The description for r^0 is different than for all r^i where $i \neq 0$.
$r^0 = \{r^0_0, r^0_1, r^0_2, r^0_3, r^0_4, r^0_5, r^0_6, r^0_7\}$
where $r^0_0 = (0, 0)$
$r^0_1 = (0.25\gamma, 0)$
$r^0_2 = (0.5\gamma, 0.5\gamma + \zeta)$
$r^0_3 = (0.75\gamma, 0)$
$r^0_4 = (0.5a_1 + 0.25\gamma, 0)$
$r^0_5 = (0.5a_1 + 0.5\gamma, 0.5\gamma + \zeta)$
$r^0_6 = (0.5a_1 + 0.75\gamma, 0)$
$r^0_7 = (\delta, 0)$

if $i \neq 0$, r^i also contains γ-spikes pointing upward. The polylines are defined by $r^i = \{r^i_0, r^i_1, r^i_2, r^i_3, r^i_4, r^i_5, r^i_6, r^i_7, r^i_8\}$
$r^i_0 = (-4i\delta - \gamma, 4i\delta + 2i\gamma)$
$r^i_1 = (-0.25\gamma, 4i\delta + 2i\gamma)$
$r^i_2 = (0, 4i\delta + [2i - 0.5]\gamma - \zeta)$
$r^i_3 = (0.5\gamma, 4i\delta + [2i + 0.5]\gamma + \zeta)$
$r^i_4 = (0.75\gamma, 4i\delta + 2i\gamma$
$r^i_5 = (0.5a_{i+1} + 0.25\gamma, 4i\delta + 2i\gamma)$
$r^i_6 = (0.5a_{i+1} + 0.5\gamma, 4i\delta + [2i + 0.5]\gamma + \zeta)$
$r^i_7 = (0.5a_{i+1} + 0.75\gamma, 4i\delta + 2i\gamma)$
$r^i_8 = ([4i + 1]\delta, 4i\delta + 2i\gamma)$

$m^i = \{m^i_0, m^i_1\}$
where $m^i_0 = ([4i + 1]\delta, [4i - 1]\delta + [2i - 0.5]\gamma)$
$m^i_1 = ([4i + 3]\delta + \gamma, [4i - 1]\delta + [2i - 0.5]\gamma)$

$l^i = \{l^i_0, l^i_1, l^i_2, l^i_3, l^i_4, l^i_5, l^i_6, l^i_7, l^i_8\}$
where $l^i_0 = ([4i - 1]\delta + \gamma, [4i - 2]\delta + [2i - 1]\gamma)$
$l^i_1 = (0.5a_i + 0.75\gamma, [4i - 2]\delta + [2i - 1]\gamma)$
$l^i_2 = (0.5a_i + 0.5\gamma, [4i - 2]\delta + [2i - 1.5]\gamma - \zeta)$
$l^i_3 = (0.5a_i + 0.25\gamma, [4i - 2]\delta + [2i - 1]\gamma)$
$l^i_4 = (0.75\gamma, [4i - 2]\delta + [2i - 1]\gamma)$
$l^i_5 = (0.5\gamma, [4i - 2]\delta + [2i - 1.5]\gamma - \zeta)$
$l^i_6 = (0, [4i - 2]\delta + [2i - 0.5]\gamma) + \zeta$
$l^i_7 = (-0.25\gamma, [4i - 2]\delta + [2i - 1]\gamma)$
$l^i_8 = ([-4i + 2]\delta, [4i - 2]\delta + [2i - 1]\gamma)$

$f^i = \{f^i_0, f^i_1\}$
where $f^i_0 = ([-4i + 2]\delta, [4i - 3]\delta + [2i - 1.5]\gamma)$
$f^i_1 = (-4i\delta - \gamma, [4i - 3]\delta + [2i - 1.5]\gamma)$

$t = \{t_0, t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9\}$ where
$t_0 = ([4n-1]\delta + \gamma, [4n-1]\delta + [2n-0.5]\gamma)$
$t_1 = ([-4n+4]\delta - \gamma, [4n-1]\delta + [2n-0.5]\gamma)$
$t_2 = ([-4n+4]\delta - \gamma, [4n-2]\delta + [2n-1]\gamma)$
$t_3 = (0.25\gamma, [4n-2]\delta + [2n-1]\gamma)$
$t_4 = (0.5\gamma, [4n-2]\delta + [2n-1.5]\gamma - \zeta)$
$t_5 = (0.75\gamma, [4n-2]\delta + [2n-1]\gamma)$
$t_6 = (0.5a_n + 0.25\gamma, [4n-2]\delta + [2n-1]\gamma)$
$t_7 = (0.5a_n + 0.5\gamma, [4n-2]\delta + [2n-1.5]\gamma - \zeta)$
$t_8 = (0.5a_n + 0.75\gamma, [4n-2]\delta + [2n-1]\gamma)$
$t_9 = (B + n\gamma, [4n-2]\delta + [2n-1]\gamma)$

B Computing the x-coordinates of skip vertices

If we want to skip vertical level l^i we need to place a skip vertex on r^{i-1} such that one of the holes between r^{i-1} and l^i, and the hole between l^i and r^i are aligned. We can construct lines between these holes and then intersect those lines with r^{i-1} to find the viable locations for skip vertices. We will show how to compute the vertex that is on the line containing the leftmost hole between r^{i-1} and l^i. How to compute the other location follows easily.
Each hole has the x-coordinate of the γ-spike that induces it and is exactly between two vertical levels. So the coordinates of the leftmost hole between r^{i-1} and l^i are $(0.5\gamma, [4i-3]\delta + [2i-1.5]\gamma)$. The coordinates of the hole between l^i and r^i are $(0, [4i-1]\delta + [2i-0.5]\gamma)$.
Using a simple linear equation the function for the line between the holes is
$\frac{2\delta+\gamma}{-0.5\gamma}x + [4i-1]\delta + [2i-0.5]\gamma = y$
The y-coordinate of all points on r^{i-1} (except γ-spikes) is $[4i-4]\delta + [2i-2]\gamma$, so to find the x-coordinate we need to solve
$\frac{2\delta+\gamma}{-0.5\gamma}x + [4i-1]\delta + [2i-0.5]\gamma = [4i-4]\delta + [2i-2]\gamma$
Getting rid of the intercept on the left side of the equation we get
$\frac{2\delta+\gamma}{-0.5\gamma}x = -3\delta - 1.5\gamma$
If we let $z = (\delta + 0.5\gamma)$ then our equation becomes
$\frac{2xz}{-0.5\gamma} = -3z$
Dividing by z we get
$\frac{2x}{-0.5\gamma} = -3$
This solves to $x = 0.75\gamma$.

Using the rightmost hole only changes the denominator of the slope of our line function to $-0.5\gamma - 0.5a_i$, making the final equation solve to $x = 0.75\gamma + 0.75a_i$

Computing the x-coordinate of the vertex needed to skip a level r^i is very similar. The coordinates of the involved holes are $(0, [4i-1]\delta + [2i-0.5]\gamma)$ and $(0.5\gamma, [4i+1]\delta + [2i+0.5]\gamma)$ (assuming we use the leftmost hole between r^i and l^{i+1}). l^i has y-coordinate $[4i-2]\delta + [2i-1]\gamma$, so the equation we are solving is
$\frac{2\delta+\gamma}{0.5\gamma}x + [4i-1]\delta + [2i-0.5]\gamma = [4i-2]\delta + [2i-1]\gamma$
using the same steps as above this becomes
$\frac{2\delta+\gamma}{0.5\gamma}x = -\delta - 0.5\gamma$
$\frac{2xz}{0.5\gamma} = -z$

$\frac{2x}{0.5\gamma} = -1$
$x = -0.25\gamma$
Using the rightmost hole instead makes our equation solve to $x = -0.25\gamma - 0.25a_{i+1}$.