Polyline Simplification under the Local Fréchet Distance has Subcubic Complexity in 2D

Sabine Storandt
University of Konstanz, Konstanz, Germany

Johannes Zink
University of Würzburg, Würzburg, Germany

Abstract

Given a polyline on \( n \) vertices, the polyline simplification problem asks for a minimum size subsequence of these vertices defining a new polyline whose distance to the original polyline is at most a given threshold under some distance measure. In this paper, we improve the long-standing running time bound for the simplification of polylines under the local Fréchet distance. The best algorithm known so far is by Imai and Iri and has a cubic running time in \( n \). We present an algorithm with a running time of \( O(n^2) \) under the \( L_1 \) and \( L_\infty \) norm, and \( O(n^2 \log n) \) under \( L_{p \in (1, \infty)} \) (including the Euclidean norm \( L_2 \)). Our approach is based on the ideas of Chan and Chin, who showed that under the local Hausdorff distance, the Imai-Iri algorithm can be improved to run in quadratic time for \( L_1, L_2, \) and \( L_\infty \). However, the Hausdorff distance does not take the order of the points along the polyline into account. The Fréchet distance, which is sensitive to the course of the polylines, is hence often deemed the superior distance measure for polyline similarity but it also more intricate to compute. So far, the significantly faster simplification algorithms for the Hausdorff distance made them preferable for practical application. But our new algorithm for simplification under the Fréchet distance matches the running time bounds for the Hausdorff distance up to logarithmic factors and thus allows the usage of this more suitable distance measure.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Polyline simplification, Local Fréchet distance, Exact algorithm, \( p \)-norm

1 Introduction

Polyline simplification is an extensively studied topic due to its relevance to a variety of applications, such as trajectory and shape analysis, data compression or map visualization. The goal of polyline simplification is to replace a given polyline with \( n \) vertices with a simpler one, while ensuring that the input and the output polyline are sufficiently similar. The similarity is governed by a given distance threshold \( \varepsilon \). There are different types of simplification, depending on where the vertices of the output polyline are allowed to be and how similarity is measured. We focus on the case where the output polyline has to consist of a subsequence of the input polyline vertices (always including the first and the last vertex). Line segments between these vertices are then called shortcuts. To determine the similarity of the input and output polyline, the Hausdorff and the Fréchet distance are the most commonly used measures. Both can be applied either globally or locally. In the global version, the distance between the entire input and output polyline is measured and must not exceed \( \varepsilon \). In the local version, the distance between each shortcut and the part of the input polyline it bridges must not exceed \( \varepsilon \). For many applications, local similarity is more sensible and intuitive. Simplification with global similarity has only been studied recently. For the global Hausdorff distance, computing a simplified polyline with the smallest number of shortcuts yields an NP-hard problem \cite{11}. For the global Fréchet distance, a \( O(n^3) \) algorithms was designed by Bringmann and Chaudhury \cite{5}. For the more extensively studied problem of simplification with local Hausdorff or Fréchet distance, the Imai-Iri algorithm \cite{10}, published in 1988, guarantees a running time of \( O(n^3) \) by reducing the simplification problem...
Polyline Simplification under the Local Fréchet Distance has Subcubic Complexity in 2D

to a graph problem. Basically, they construct the graph where the polyline vertices are the nodes and there is an edge between a pair of vertices if a shortcut is possible. For the local Hausdorff distance, Chan and Chin showed in 1996 that the running time of the Imai-Iri algorithm can be improved to \(O(n^2)\) by making the graph construction phase more efficient \([7]\). For the local Fréchet distance, though, the algorithm by Chan and Chin cannot be applied and also no other improvements for optimal polyline simplification under this distance measure were made so far. Recent papers \([1, 5, 11]\) still report the cubic running time of the Imai-Iri algorithm under the local Fréchet distance.

The Fréchet distance is often considered to be the superior distance measure, as it takes the order of the vertices along the polyline into account while the Hausdorff distance ignores this aspect. For long input polylines, though, a cubic running time is prohibitive for practical use. In this paper, we close the gap between running times for polyline simplification under local Hausdorff and local Fréchet distance up to a logarithmic factor. We achieve this by proposing an algorithm that builds on the method of Chan and Chin, but additionally maintains a so called wave front. For a polyline vertex, the wave front encodes the region in which the vertices must lie to which valid shortcuts can be created that do not exceed a local Fréchet distance of \(\varepsilon\). We describe in detail how to construct and update the wave front efficiently and prove that it enables the computation of optimal polyline simplifications under the local Fréchet distance in subcubic time. Detailed results are listed in Table 1.

Related Work

The most famous polyline simplification algorithm is due to Imai and Iri \([10]\). It computes an optimal simplification in time \(O(n^3)\) for local distance measures for which the distance of a polyline of length \(n\) to a single segment is computable in \(O(n)\) time. This applies to the local Hausdorff and the local Fréchet distance. For the global Fréchet distance, a dynamic program (DP) with a running time of \(O(n^3)\) was presented by Bringmann and Chaudhury \([5]\). The Imai-Iri and the DP algorithm can also be generalized to work in \(\mathbb{R}^{d \geq 2}\) with the running time only increasing by a polynomial factor in \(d\). For the local and global Fréchet distance, no subcubic simplifications algorithms were known so far. For the local Hausdorff distance, Chan and Chin \([7]\) showed that the Imai-Iri algorithm can be improved to run in \(O(n^2)\) for \(L_1, L_2\) and \(L_\infty\) (the concept also can be applied to any \(L_{p \in (1, \infty)}\) up to possible numerical issues that are further discussed in Section 4). Furthermore Barequet et al. \([3]\) proposed an \(O(n^2 \log n)\) algorithm for the local Hausdorff distance under the \(L_2\) norm which works in \(\mathbb{R}^3\).

Bringmann and Chaudhury \([5]\) have also proven a conditional lower bound for simplification in \(\mathbb{R}^d\) under the local Hausdorff distance as well as under the local and global Fréchet distance. More precisely, for \(L_p\) with \(p \in [1, \infty), p \neq 2\), algorithms with a running time subcubic in \(n\) and polynomial in \(d\) were ruled out (unless the \(\forall \forall 3\)-OV hypothesis fails). For large dimensions \(d\), the above discussed algorithmic upper bounds of \(O(n^3 \cdot \text{poly}(d))\) for polyline simplification are hence tight. However, the lower bound still allows the existence of simplification algorithms with a running time in \(O(n^k \cdot \exp(d))\) with \(k < 3\). Hence, for small values of \(d\) (which are of high practical relevance), faster algorithms are possible, as evidenced by a \(2^{O(d)} n^2\) algorithm for the local Hausdorff distance under the \(L_1\) norm \([5]\). For \(p = 2\) and \(p = \infty\), the best currently known conditional lower bound for the three similarity measures, local Hausdorff distance, local Fréchet distance and global Fréchet distance, was proven by Buchin et al. \([6]\). It rules out algorithms with a subquadratic running time in \(n\) and polynomial running time in \(d\) (unless SETH fails). Here again, better running times for simplification problems in a low-dimensional space with \(d \in o(\log n)\) are still possible. Table 1 provides an overview of known lower and upper bounds for optimal polyline simplification.
We re-use parts of their notation regarding the cone-shaped wedge and extend it by our notation regarding the wave front, which we introduce for our algorithm. In Section 4, we analyze the structure of the wave front, which we later use in Section 5 where we describe our algorithm for computing polyline simplifications in two dimensions in subcubic time. After we have described and proven the general functionality of our algorithm, we address the specifics in the L_1 and L_∞, L_2, and L_p norm in Sections 4.1, 4.2, 4.3, respectively. At the end, we discuss our findings and possible future research questions in Section 5.

Outline of the Paper

This paper is structured as follows. In Section 2, we define the problem of polyline simplification, the relevant distance measures, and the L_p norm. Moreover, we recap the algorithms by Imai and Iri [10] and by Chan and Chin [7], upon which our new techniques heavily build. We re-use parts of their notation regarding the cone-shaped wedge and extend it by our notation regarding the wave front, which we introduce for our algorithm. In Section 4, we analyze the structure of the wave front, which we later use in Section 5 where we describe our algorithm for computing polyline simplifications in two dimensions in subcubic time. After we have described and proven the general functionality of our algorithm, we address the specifics in the L_1 and L_∞, L_2, and L_p norm in Sections 4.1, 4.2, 4.3, respectively. At the end, we discuss our findings and possible future research questions in Section 5.
Preliminaries

A polyline is a series of line segments that are defined by a sequence of \( d \)-dimensional points \( \mathcal{L} = (p_1, p_2, \ldots, p_n) \), which we call vertices. There, \( n \) is the length of a polyline. The continuous curve induced by these vertices is denoted as \( c_\mathcal{L} : [1, n] \to \mathbb{R}^d \) with \( c_\mathcal{L} : x \mapsto ([x] + 1 - x)p_{[x]} + ([x])p_{[x]} \). The polyline simplification problem is defined as follows.

**Definition 1** (Polyline Simplification). Given a polyline \( \mathcal{L} = (p_1, p_2, \ldots, p_n) \), a distance measure \( d_X \) for determining the distance between two polylines, and a distance threshold parameter \( \varepsilon \), the objective is to obtain a minimum size subsequence \( S \) of \( \mathcal{L} \) such that \( p_1, p_n \in S \), and \( d_X(\mathcal{L}, S) \leq \varepsilon \). We refer to \( S \) as a simplification of the (original) polyline \( \mathcal{L} \).

**Distance measures**

Next, we discuss typical candidates for such a distance measure \( d_X \), namely the Hausdorff and the Fréchet distance in their local and their global variant.

**Definition 2** (Global Hausdorff Distance). Given two polylines \( \mathcal{L}_1 = (p_1, p_2, \ldots, p_n) \) and \( \mathcal{L}_2 = (q_1, q_2, \ldots, q_m) \), the global (undirected) Hausdorff distance \( d_{gH}(\mathcal{L}_1, \mathcal{L}_2) \) is defined as

\[
d_{gH}(\mathcal{L}_1, \mathcal{L}_2) := \max_{p \in \mathcal{L}_1} \sup_{q \in \mathcal{L}_2} \inf d(p, q), \sup_{q \in \mathcal{L}_2} \inf d(p, q) \,
\]

where \( \sup \) is the supremum, \( \inf \) is the infimum and \( d(p, q) \) is the distance between the points \( p \) and \( q \) under some norm (typically the Euclidean distance).

An often raised criticism concerning the use of the Hausdorff distance is that it does not reflect the similarity of the courses of two polylines. In contrast, the Fréchet distance measures the maximum distance between two polylines while traversing them in parallel and is therefore often regarded as the better suited measurement for polyline similarity.

**Definition 3** (Global Fréchet Distance). Given two polylines \( \mathcal{L}_1 = (p_1, p_2, \ldots, p_n) \) and \( \mathcal{L}_2 = (q_1, q_2, \ldots, q_m) \), the global Fréchet distance \( d_{gF}(\mathcal{L}_1, \mathcal{L}_2) \) is defined as

\[
d_{gF}(\mathcal{L}_1, \mathcal{L}_2) := \inf_{\alpha, \beta} \max_{t \in [0, 1]} d(c_\mathcal{L}_1(\alpha(t)), c_\mathcal{L}_2(\beta(t)))
\]

where \( \alpha : [0, 1] \to [1, n] \) and \( \beta : [0, 1] \to [1, m] \) are continuous and non-decreasing functions with \( \alpha(0) = \beta(0) = 1 \), \( \alpha(1) = n \), \( \beta(1) = m \).

Traditionally, in the context of polyline simplification, the local Hausdorff and local Fréchet distance is used, which only measures the Hausdorff or Fréchet distance between a line segment of the simplification and its corresponding subpolyline in the original polyline. We define only the local Fréchet distance – the local Hausdorff distance is defined analogously.

**Definition 4** (Local Fréchet Distance). Given a polyline \( \mathcal{L} = (p_1, p_2, \ldots, p_n) \) and a simplification \( S = (p_1 = p_{s_1}, p_{s_2}, \ldots, p_{s_{|S|}} = p_n) \) of \( \mathcal{L} \), the local Fréchet distance

\[
d_{IF}(S, \mathcal{L}) := \max_{i < 1 : \ldots : |S| - 1} d_{gF}(\{p_{s_i}, p_{s_{i+1}}\}, \mathcal{L}[p_{s_i}, p_{s_{i+1}}])
\]

where \( \{p_{s_i}, p_{s_{i+1}}\} \) is the polyline of length two (i.e., the line segment) from \( p_{s_i} \) to \( p_{s_{i+1}} \) and \( \mathcal{L}[p_{s_i}, p_{s_{i+1}}] \) is the (sub)polyline we obtained by taking the substring from \( p_{s_i} \) to \( p_{s_{i+1}} \) of \( \mathcal{L} \).
Figure 1 Unit circles in $L_p$ norms for selected values of $p$.

When using the local Fréchet distance, we can tell for each pair of vertices $p_i, p_j$ (for $1 \leq i < j \leq n$) in the original polyline independently whether a simplification may contain the line segment $(p_i, p_j)$ or not by only computing the Fréchet distance between the line segment $(p_i, p_j)$ and its corresponding subpolyline. When considering such a pair $(p_i, p_j)$ as a line segment for a simplification, we call it a shortcut. If the Hausdorff/Fréchet distance between a shortcut and its corresponding subpolyline does not exceed the distance threshold $\varepsilon$, we call it a valid shortcut. Note that trivially $(p_i, p_{i+1})$ is always a valid shortcut.

$L_p$ Norm

In the definition of the Hausdorff as well as the Fréchet distance, we can choose how the distance between two $d$-dimensional points, or vectors, is determined. Typically, a vector norm is used for this purpose. For $p \in (0, \infty)$, the $L_p$ norm of a vector $x \in \mathbb{R}^d$ is defined as $\|x\|_p := \left(\sum_{i=1}^d |x_i|^p\right)^{1/p}$. For $p = 1$, it is called the Manhattan norm, for $p = 2$, we have the Euclidean norm. For $p \to \infty$, $L_\infty$ is called the maximum norm and it is defined as $\max_{i=1,\ldots,n} |x_i|$. The unit sphere $S^d_p$ is the set of points in $\mathbb{R}^d$ within unit distance to the origin. While this unit is conventionally set to 1, we will use $\varepsilon$ here instead, as this allows for easier integration with our goal of polyline simplification with error bound $\varepsilon$. We hence define $S^d_p := \{x \in \mathbb{R}^d \mid \|x\|_p \leq \varepsilon\}$. For $d = 2$, $S^2_p$ is also called the unit circle in $L_p$. For $L_1$ and $L_\infty$, it actually forms a square with side length $\sqrt{2}\varepsilon$ and $2\varepsilon$, respectively. For $L_2$, it is indeed a circle with radius $\varepsilon$. For $p$ between 2 and $\infty$, it forms a supercircle which for larger $p$ resembles more and more a square; see Figure 1. We refer to a contiguous subset of the boundary of a unit circle in $L_p$ as arc. This concept will be frequently used in our algorithms to describe the regions in which feasible shortcut endpoints may be located.

Imai-Iri Algorithm and Chan-Chin Algorithm

Next we recap the polyline simplification algorithms due to Imai and Iri [10] and their optimization due to Chan and Chin [7].

The polyline simplification algorithm by Imai and Iri [10] proceeds in two phases. In the first phase, it determines for each pair of vertices whether there is a valid shortcut to
contribute the shortcut graph. The shortcut graph is a directed acyclic graph (dag) that has a node for each vertex of the polyline \( L \) and it has a directed edge between two vertices if and only if there is a valid shortcut between these two vertices. The edge direction is towards the vertex coming later in the input polyline. In the second phase, it computes a directed shortest path from the first vertex \( p_1 \) to the last vertex \( p_n \) within the shortcut graph, which can easily be done in \( \mathcal{O}(n^2) \) time in a dag with \( n \) nodes. Clearly, this shortest path \( S \) is a minimum-length simplification of the original polyline \( L \). The total running time, however, is dominated by the first phase. For each pair of vertices, the algorithm checks in \( \mathcal{O}(n) \) time whether the (local) Hausdorff distance would be exceeded by this pair or not. This can be extended to the local Fréchet distance since we can also check in \( \mathcal{O}(n) \) time whether the Fréchet distance between a line segment and a polyline having \( \mathcal{O}(n) \) vertices exceeds \( \varepsilon \). This results in a total running time of \( \mathcal{O}(n^3) \).

Chan and Chin \cite{Chan2006} (and also we) work in exactly the same framework introduced by Imai and Iri, but Chan and Chin introduce a faster technique to compute the shortcut graph for the local Hausdorff distance. Starting once at each vertex \( p_i \) for \( i \in \{1, \ldots, n\} \), they traverse the rest of the polyline vertex by vertex in total linear time to determine all valid shortcuts originating at \( p_i \). To this end, they maintain a cone-shaped region called wedge which is the area in which all valid shortcuts are required to lie. More precisely, the wedge is an angular region having its origin at \( p_i \) and being the intersection of all angular regions (which we call local wedges) that define the areas where valid shortcuts may lie for each intermediate vertex. When traversing the polyline, the wedge iteratively becomes narrower. When starting at \( p_i \) and encountering \( p_j \) during the traversal, we denote by \( D_{ij} \) the local wedge of \( p_i \) and \( p_j \), which is the area between the two tangential rays of the unit circle around \( p_j \) emanating at \( p_i \); see Fig. 2b. Moreover, we denote by \( W_{ij} \) the (global) wedge where \( W_{ij} := \bigcup_{k \in \{i+1, i+2, \ldots, j\}} D_{ik} \). Clearly, to be a valid shortcut, the line segment \((p_i, p_j)\) needs to lie within \( W_{i(j-1)} \). Since we can check containment in \( W_{i(j-1)} \) in constant time and we can update the wedge from \( W_{i(j-1)} \) to \( W_{ij} \) in constant time, the running time of this phase is \( \mathcal{O}(n) \) per starting vertex \( p_i \) and \( \mathcal{O}(n^2) \) in total. Note that this procedure may produce false positives, i.e., some shortcuts are added to the shortcut graph \( G_1 \) though not being valid. This problem is encountered by repeating this whole process in reverse direction of the polyline. This way a second (false positive) shortcut graph \( G_2 \) is produced and the (real) shortcut graph \( G \) is the intersection of both, i.e., it has an edge if and only if this edge appears in both \( G_1 \) and \( G_2 \). Still, the running time of \( \mathcal{O}(n^2) \) for the first phase matches the running time of the second phase, resulting in a total running time of \( \mathcal{O}(n^3) \).

**Wedges, Wave Fronts, and Valid Regions**

In our algorithm, we also use the notion of \( D_{ij} \) and \( W_{ij} \) and generalize it to all \( L_p \) norms for \( p \in [1, \infty] \). Namely, \( D_{ij} \) is defined as the area between the two outermost rays intersecting or touching the unit circle around \( p_i \) and emanating at \( p_j \). \( W_{ij} \) is defined inductively as the intersection of a narrower version \( W'_{i(j-1)} \) of \( W_{i(j-1)} \) and \( D_{ij} \). We define \( W_{ii} \) and all \( D_{ij} \) where \( d(p_i, p_j) \leq \varepsilon \) to be the whole plane. From now on assume without loss of generality that \( p_i \) is the bottommost point of any \( W_{ij} \). To accommodate a structure using the Fréchet distance, we introduce to each wedge \( W_{ij} \) a wave front (defined below). Intuitively, a wave front is a sequence of arcs of unit circles of the considered \( L_p \) norm such that the leftmost arc starts at

---

1 We describe in Section 4 how and why \( W_{i(j-1)} \) is made narrower first in our algorithm. There, we also use \( W'_{ij} \) to denote an intermediate stage of the wedge when updating from \( W_{i(j-1)} \) to \( W_{ij} \).
(a) Under the $L_1$ norm, the unit circles are squares of side length $\sqrt{2}\varepsilon$ whose boundary is rotated by 45 degrees relative to the coordinate axes. The wave front consists of one or two line segments.

(b) Under the $L_2$ norm, the unit circles are circles of radius $\varepsilon$. The wave front consists of at most a linear number of circular arcs.

(c) Under the $L_\infty$ norm, the unit circles are squares of side length $2\varepsilon$ whose boundary is parallel to the coordinate axes. The wave front consists of one or two line segments.

Figure 2 Iterative construction of the wedge in the $L_1$, $L_2$, and $L_\infty$ norm: From left to right, the local wedges $D_{12}$, $D_{13}$, and $D_{14}$ are visualized in pink. Here, the intersection of local wedges defines the wedges $W_{12}$, $W_{13}$, and $W_{14}$, respectively. This is as in the algorithm by Chan and Chin [7]. Additionally, our algorithm uses the wave front, which is a sequence of unit circle arcs – here depicted in blue. Within the wedge and above the wave front, there is the valid region (depicted in green). This is the area, where a subsequent vertex $p_j$ needs to lie if there is a valid shortcut $(p_1, p_j)$. For example, $(p_1, p_5)$ is a valid shortcut in the $L_\infty$ norm, whereas in the $L_1$ and the $L_2$ norm it is not.
the left boundary of $W_{ij}$ and the rightmost arc ends at the right boundary of $W_{ij}$. We call the region within $W_{ij}$ and above and on the wave front the valid region of $W_{ij}$ (the valid region of $W_{ij}$ is the whole plane). Similar to what has been the wedge in the algorithm by Chan and Chin is the valid region in our algorithm – the region where the vertex encountered next is required to lie if it has a valid shortcut from $p_i$.

The wave front of $W_{ij}$ is defined inductively. Let $c_j$ be the unit circle of radius ε around $p_j$ and let $l_j$ ($r_j$) be the left (right) tangential point of $c_j$ and $D_{ij}$. Between $l_j$ and $r_j$, there are two arcs of $c_j$ – the bottom arc and the top arc. Clearly, any ray inside $D_{ij}$ emanating at $p_i$ intersects the bottom and the top arc one time each. We call the bottom arc of $c_j$ between $l_j$ and $r_j$ the wave of $D_{ij}$. We call the region within $D_{ij}$ and above its wave the local valid region of $D_{ij}$. The wave front of $W_{ij}$ (for $j > i$) is the boundary of the intersection of the valid region of $W_{i(j-1)}$ and the local valid region of $D_{ij}$ within $W_{ij}$ and excluding the boundary of $W_{ij}$. Intuitively, it is the wave front of $W_{i(j-1)}$ within $W_{ij}$ where we cut out the (bottom) boundary of $c_j$. See Figure 2 for an example of a polyline that shall be simplified in the $L_1$, $L_2$, and $L_\infty$ norm, and see Figure 3 for an example of a wave front in the $L_2$ norm.

3 Properties of the Wave Front

The main goal of this section is to prove the following three properties of the wave front, which we use later to build our polyline simplification algorithm under the local Fréchet distance. (i) any ray emanating at our base vertex $p_i$ intersects the wave front at most once, (ii) any unit circle intersects the wave front at most twice, and (iii) the size of the wave front is constant in the $L_1$ and $L_\infty$ norm and linear in the other $L_p$ norms ($p \in (1, \infty)$). This is critical to obtain the runtimes of our algorithm depending on the used norm. From now on we assume that we never compare two identical unit circles (and also our polylines consist of $n$ distinct points). Note, however, that when implementing our polyline simplification algorithm in practice, one should also catch this special case.

First, we show inductively that any ray intersects the wave front at most once. The key insight is that the wave front uses only bottom arcs.

Lemma 5. Any ray emanating at $p_i$ intersects the wave front at most once.

Proof. We prove this statement inductively. As $W_{i(i+1)} = D_{i(i+1)}$, consider the wave of $D_{i(i+1)}$. Since the unit circle in any $L_p$ norm for $p \in [1, \infty]$ is convex, any ray emanating at $p_i$ intersects a unit circle at most twice. Observe that within $D_{i(i+1)}$, the first intersection is with the bottom circular arc of the unit circle around $p_{i+1}$ and the second intersection is with the top circular arc of the same unit circle. As the wave of $D_{i(i+1)}$ is defined as the bottom circular arc within $D_{i(i+1)}$, any ray emanating at $p_i$ intersects the wave of $D_{i(i+1)}$ at most once.

It remains to show the induction hypothesis for all $j > i + 1$. We know that any ray emanating at $p_i$ intersects the wave front of $W_{i(j-1)}$ at most once. The wave front of $W_{ij}$ is the boundary of the intersection of the valid region of $W_{i(j-1)}$ and the local valid region of $D_{ij}$. Consider a ray $R$ originating at $p_i$. The ray $R$ enters the valid region of $W_{i(j-1)}$ at most at one point $p$ when it intersects the wave front of $W_{i(j-1)}$ and it enters the local valid region of $D_{ij}$ at most at one point $p'$ when it intersects the wave of $D_{ij}$. Hence, $R$ enters the intersection of the valid region of $W_{i(j-1)}$ and the local valid region of $D_{ij}$ at most at one point – namely either at $p$ or at $p'$ (or $p = p'$). This is by definition the only point of the wave front of $W_{ij}$ that is shared with $R$.

The second property, which show in Lemma 8 is that each unit circle intersects the wave
Figure 3 Example of a wave front (blue) inside the wedge (dark gray) made by arcs of unit circles (orange) in the $L_2$ norm. The dashed blue line segments indicate the places where the arcs change.

Figure 4 Illustration of the situation described in the proof of Lemma 6.

front at most twice. By the following two lemmas we first gain more structural insights about the interplay of unit circles and the wave front, which we also re-use in proofs of Section 4.

Lemma 6. Given two unit circles and a point $p$, if the two bottom arcs (with respect to $p$) intersect, then the second intersection point is between their top arcs.

Proof. For an illustration of this proof see Figure 4. Let $c$ and $c'$ be the two unit circles with the center of $c$ being left of the center of $c'$ w.r.t. $p$. Now the cone between the right tangential from $p$ on $c$ and the left tangential from $p$ on $c'$ contains all of the intersection area of $c$ and $c'$, and hence also both intersection points. We call the tangential points $r_c$ and $l_{c'}$, respectively. Note that $r_c = l_{c'}$ is excluded as then $c$ and $c'$ would only have a single intersection point. For the intersection point $s$ between the bottom arcs of $c$ and $c'$, we know that the line segment $ps$ does not intersect the inner part of any of the two circles by definition of the bottom arc. Hence the ray elongating this segment has to go through the intersection area of $c$ and $c'$ above $s$. Therefore, the partial bottom arc of $c$ from $s$ to $r_c$ and the partial bottom arc of $c'$ from $s$ to $l_{c'}$ are both on the boundary of the intersection area. As the intersection area is convex, it means that the line segment $l_{c'}r_c$ is fully contained in the intersection area, and the intersection points have to be on opposite sides of the line through $l_{c'}$ and $r_c$. Accordingly, the second intersection point $s'$ of $c$ and $c'$ then has to lie above $l_{c'}r_c$ and is therefore on the respective top arcs of $c$ and $c'$.

Lemma 7. If a unit circle $c$ intersects the wave front more than once, then on the left side of the leftmost intersection point $s_1$ (relative to rays originating in $p_i$) and on the right side of the rightmost intersection point $s_2$, $c$ is below the wave front. In other words, the intersection pattern depicted in Figure 5a cannot occur.

Proof. Clearly, if at $s_1$ the top arc of $c$ intersects the wave front, then on the left side of $s_1$, $c$ is below the wave front. Symmetrically, the same holds for $s_2$.

Now assume that at $s_1$ and at $s_2$, the bottom arc of $c$ intersects the arcs $a_j$ and $a_k$ of the wave front, respectively. We denote their unit circles by $c_j$ and $c_k$. W.l.o.g. let $c$ on the left side of $s_1$ be above the wave front. $c_j$ contains the rest of the wave front (we will show this later in Lemma 11 without depending on this lemma) including all of $a_k$. This means, that $c$ intersects $c_j$ at $s_3$ in between $s_1$ and $s_2$ (potentially $s_2 = s_3$ if $c_j = c_k$); see Figure 5b.
Because the intersection of $c$ at $s_2$ is with the bottom arc of $c$, the intersection of $c$ and $c_j$ at $s_3$ is also with the bottom arc of $c$. This contradicts Lemma 8.

Finally, assume w.l.o.g. that at $s_1$ the top arc of $c$ intersects the arc $a_j$ of the wave front and at $s_2$ the bottom arc of $c$ intersects the arc $a_k$ of the wave front; see Figure 5. Again by Lemma 11, the unit circle $c_j$ of $a_j$ contains the wave front including the whole arc $a_k$. Hence, there is an intersection point $s_3$ (where $s_2 \neq s_3$ and $c_j \neq c_k$ as otherwise $c$ and $c_j$ would have an intersection between their bottom arcs and between a bottom and a top arc). At $s_3$ there is the bottom arc of $c$ (since later at $s_2$, there is also the bottom arc of $c$ involved). If $c_j$ also would have its bottom arc at $s_3$, it would contradict Lemma 5. Therefore, at $s_3$ is the top arc of $c_j$. This however means that $s_3$ is outside $D_{ij}$ — a contradiction. ▢

Lemma 8. Any unit circle of radius $\varepsilon$ intersects the wave front at most twice.

Proof. Again, we prove this statement inductively. Say $p_i$ is our start vertex and we consider the wave front of $W_{i(i+1)}$, which is the same as the wave of $D_{i(i+1)}$, which is part of the boundary of a unit circle. Since each pair of unit circles in the $L_p$ norm for $p \in [1, \infty]$ intersects at most twice, we know that any unit circle intersects the wave front of $W_{i(i+1)}$ at most twice.

It remains to show the induction hypothesis for all $j > i + 1$. Assume for contradiction that a unit circle $c$ intersects the wave front of $W_{ij}$ more than twice. Observe that the wave front of $W_{ij}$ is a subset of the wave front of $W_{(i-1)}$ and the wave of $D_{ij}$. Say $c$ intersects the wave front of $W_{(i-1)}$ at $q_1$ and $q_2$ and $c$ intersects the wave of $D_{ij}$ at $q_3$ and $q_4$. (Maybe one of these points does not exist.) Next, we argue topologically that at most two points of $\{q_1, q_2, q_3, q_4\}$ lie on the wave front of $W_{ij}$, which is a contradiction.

The wave front of $W_{(i-1)}$ and the wave of $D_{ij}$ intersect at most twice. Let these intersection points from left to right be $s_1$ and $s_2$; see Figure 6. Let the subdivisions of the wave front of $W_{(i-1)}$ and the wave of $D_{ij}$ induced by $s_1$ and $s_2$ be $A_{W1}, A_{W2}, A_{W3}$ and $a_{D1}, a_{D2}, a_{D3}$, respectively. Some of them may be empty. Clearly, the wave front of $W_{ij}$ is either $A_{W1} - a_{D2} - A_{W3}$ or $a_{D1} - A_{W2} - a_{D3}$. By 7 we know that it cannot be $a_{D1} - A_{W2} - a_{D3}$, therefore, it is $A_{W1} - a_{D2} - A_{W3}$.

Next, we analyze the intersection points $q_3$ and $q_4$ (maybe $q_4$ does not exist). Either one or two of them lies on $a_{D2}$ as otherwise there are no more than two intersection points of $c$ with the new wave front.
Case A: The intersection points \( q_3 \) and \( q_4 \) lie on \( a_{D_2} \); see Figure 6a. As both intersection points are between the unit circle \( c \) and the wave of \( D_{ij} \), i.e., a bottom arc of another unit circle, we know by Lemma 5 that at \( q_3 \) and \( q_4 \) (and thus also in between), there is the top arc of \( c \). There is no ray \( R \) to the left of \( q_3 \) or to the right of \( q_4 \), originating at \( p_i \) and intersecting the arc of \( c \) between \( q_3 \) and \( q_4 \) as otherwise \( R \) would intersect the top arc of \( c \) twice. Therefore, the arc of \( c \) between \( q_3 \) and \( q_4 \) lies in the valid region (dashed orange in Figure 6a) without reaching \( AW_1 \) or \( AW_3 \). When \( c \) passes through \( q_3 \) and \( q_4 \), it reaches the region between \( a_{D_2} \) and \( AW_2 \). If there are intersections between \( W_{i(i+1)} \), they both lie on \( AW_2 \).

Case B: Only one intersection point, let it be \( q_3 \), lies on \( a_{D_2} \); Figure 6b. If it is a touching point, then \( c \) lies in the region between \( a_{D_2} \) and \( AW_2 \) before and after reaching \( q_3 \) (because both circles are non-identical unit circles). If it is an intersection point, then \( c \) passes through \( q_3 \) into the region between \( a_{D_2} \) and \( AW_2 \) (dashed orange in Figure 6b). To leave this region, \( q_1 \) (or \( q_2 \)) lies on \( AW_2 \). Hence, there are at most two points of \( \{q_1, q_2, q_3, q_4\} \) on the new wave front.

In the remainder of this section, we consider the size of wave front in our algorithm. By size we mean the number of arcs that constitute the wave front at a point in time. For \( L_1 \) and \( L_\infty \) we observe that any local wave consists of one or two orthogonal line segments.

Then show inductively that cutting away the wave front by parallel line segments results again in at most two (orthogonal) line segments. For the other \( L_p \) norms, we use Lemma 8 to argue that in each step, the size of the wave front increases at most by 2.

**Lemma 9.** In the \( L_1 \) norm and the \( L_\infty \) norm, the wave front consists of either one or two (orthogonal) straight line segments. These straight line segments are horizontal or vertical in the \( L_\infty \) norm and rotated by 45 degrees in the \( L_1 \) norm.

**Proof.** We show this claim inductively. For \( W_{i(i+1)} \) = \( D_{i(i+1)} \) it is just the bottom arc of a square (the unit circle in \( L_1 \) or \( L_\infty \)). Clearly, this is either one or two orthogonal line segments – horizontal or vertical line segments in the \( L_\infty \) norm and line segments rotated by 45 degrees in the \( L_1 \) norm.

When we compute the wave front of \( W_{ij} \), we compute the intersection of the valid region of \( W_{i(i-1)} \) (which is bounded by one or two orthogonal line segments by the induction hypothesis) and the local valid region of \( D_{ij} \) (which is bounded by one or two line segments parallel to the ones of \( W_{i(i-1)} \)). Computing the boundary of this intersection in the \( L_\infty \) norm can be done by computing the intersection of two axis-parallel rectangles. The intersection of two axis-parallel rectangles is again an axis-parallel rectangle. In the \( L_1 \) norm, the situation is the same but rotated by 45 degrees.

**Lemma 10.** In the \( L_p \) norm \( (p \in (1, \infty)) \) the wave front consists of at most \( \mathcal{O}(n) \) arcs.
Polyline Simplification under the Local Fréchet Distance has Subcubic Complexity in 2D

Proof. According to the inductive definition, we start with a wave front consisting of one arc. Now in each step we extend the wave front, we consider the intersection between the current valid region and a local valid region – one is defined by the current wave front, the other is defined by a single wave, i.e., a single arc $c$. In other word, this is the intersection between the current wave front and the unit circle of radius $\varepsilon$ on which $c$ lies. By Lemma 8, we know that there are at most two intersection points. This means, the number of circular arcs on the wave front increases by at most two. In the worst case, we start at vertex $p_i$ and adjust the wave front $n - 1$ times until we have created the wave front of $W_{1n}$. Therefore, any wave front consists of at most $1 + 2(n - 1) \in O(n)$ arcs. □

4 Algorithm

In this section we present our algorithm for the Imai-Iri framework to compute a simplification of a polyline in the $L_p$ norm (for $p \in [0,1]$) and two dimensions using the local Fréchet distance in $O(n^2)$ time (for the $L_1$ and $L_\infty$ norm) and in $O(n^2 \log n)$ time for the other $L_p$ norms including the most interesting and relevant case – the $L_2$ norm.

The high-level idea is as follows. As in the algorithm by Chan and Chin [7], we traverse the given polyline $n$ times – starting once from each vertex and determining all shortcuts starting at that vertex. During this process, we build the shortcut graph of this polyline. As in the classical algorithm by Imai and Iri [10], we then find a shortest path in the shortcut graph in $O(n^2)$ time, which represents an optimal simplification under the local Fréchet distance. Unlike the algorithm by Chan and Chin [7], we do not need to traverse the polyline forwards and backwards because all shortcuts we find in one direction are valid shortcuts.

Let us describe the procedure how to determine, for each vertex $p_i$ of the polyline individually, the set of subsequent vertices to which $p_i$ can have a valid shortcut starting at $p_i$. Say we are traversing the polyline from $p_i$ and we next process $p_j$. We add the edge $(p_i, p_j)$ to the shortcut graph if $p_j$ lies in the valid region of $W_{i(j-1)}$. Independently of whether $(p_i, p_j)$ is a valid shortcut or not, we update the wedge and the wave front since vertices succeeding $p_j$ could still have a valid shortcut starting at $p_i$. To this end, we first tighten the wedge to $W'_{ij}$ by computing the intersection of $W'_{i(j-1)}$ and $D_{ij}$. If $W'_{ij}$ is empty then we can abort since no vertex on the polyline after $p_j$ can have a valid shortcut starting at $p_i$. This assures that the Hausdorff distance threshold is not exceeded. To also not exceed the Fréchet distance threshold, we introduce the wave front. The shape of the wave front depends on the shape of the unit circle in the $L_p$ norm.

Say we are traversing the polyline from $p_i$ and we next process $p_j$. We add the edge $(p_i, p_j)$ to the shortcut graph if $p_j$ lies in the valid region of $W_{i(j-1)}$. Independently of whether $(p_i, p_j)$ is a valid shortcut or not, we update the wedge and the wave front since vertices succeeding $p_j$ could still have a valid shortcut starting at $p_i$. To this end, we first tighten the wedge to $W'_{ij}$ by computing the intersection of $W'_{i(j-1)}$ and $D_{ij}$. If $W'_{ij}$ is empty then we can abort since no vertex on the polyline after $p_j$ can have a valid shortcut starting at $p_i$. This assures that the Hausdorff distance threshold is not exceeded. To also not exceed the Fréchet distance threshold, we determine the region $I$ enclosed by the intersection of the unit circle around $p_j$, which we denote by $c_j$, and the valid region of $W_{i(j-1)}$; see Fig. 7a. If this region is empty, we also can abort the traversal from $p_i$ as again no vertex on the polyline after $p_j$ can have a valid shortcut starting at $p_i$. Otherwise, we update the wedge $W'_{ij}$ to $W_{ij}$ by the following operation. We determine the leftmost ray $R_l$ and the rightmost ray $R_r$ emanating at $p_i$ and intersecting $I$. We define $W_{ij}$ to be the wedge bounded by $R_l$ and $R_r$. It remains to update the wave front of $W_{i(j-1)}$ to the wave front of $W_{ij}$ regardless of whether $p_j$ lies inside the valid region or not. For an example, see Fig. 7b. First, we cut off the areas outside the bounding rays of $W_{ij}$. Second, if the local wave of $D_j$ (i.e., the bottom side of $c_j$) lies partially above the wave front of $W_{i(j-1)}$, we update the arcs constituting the wave front. To this end, we compute the up to two intersection points (see Lemma 8) of the local wave of $D_j$ and the wave front of $W_{i(j-1)}$ that lie inside $W_{ij}$. Say one such intersection...
point is $s$ and $s$ lies on the $t$-th arc $w'_t$ of the wave front of $W_{i(j-1)}$. For the wave front of $W_{ij}$, we subdivide $w'_t$ and keep the part that lies inside the local valid region of $D_j$ together with the other arcs from the wave front of $W_{i(j-1)}$ inside the local valid region of $D_j$. On the other side of $s$, we replace the previous arcs of the wave front by the arc from the boundary of $c_j$ connecting $s$ with the boundary of $W_{ij}$ or the other intersection point. In Fig. 7b we subdivide the arc $w'_t$ of the wave front of $W_{i(j-1)} - w'_t$’s left half becomes $w_t$ in the wave front of $W_{ij}$ and $w'_t$’s right half together with $w'_{t+1}$ is replaced by an arc $w_{t+1}$ of $c_j$.

Correctness

To show that our algorithm works correctly, we have to prove two things: that all shortcuts our algorithm finds are valid (Lemma 14) and that our algorithm finds all valid shortcuts (Lemma 15). As it is more difficult to show this directly, we first state three helpful lemmas.

Lemma 11. Consider the wave front of $W_{ik}$. For every point $p_j$ ($i < j \leq k$) whose unit circle $c_j$ contributes an arc of the wave front of $W_{ik}$, the wave front of $W_{ik}$ lies inside $c_j$.

Proof. All arcs $a_j'$ on the wave front belonging to a point $p_j'$ with $j' < j$ are inside $c_j$ because when the wave front of $W_{ij}$ has been constructed, the wave front of $W_{ij}$ consisted of arcs of the wave of $D_{ij}$, i.e., arcs of $c_j$, and it consisted of arcs of the wave front of $W_{i(j-1)}$ lying inside $I$, i.e., the intersection between $c_j$ and the valid region of $W_{i(j-1)}$.

All arcs $a_{k'}$ on the wave front belonging to a point $p_{k'}$ with $j < k' \leq k$ are completely inside $c_j$ because if they were not, there would be an $a_{k'}$ (which is part of the bottom arc of the unit circle $c_{k'}$) that intersects the top arc of $c_j$ at $s_1$; see Figure 8. The intersection $s_1$ is with the top arc of $c_j$ as otherwise $a_{k'}$ would be (partially) outside the local valid region of $D_{ij}$. Still for $a_j$ to be in the local valid region of $D_{ik'}$, $c_j$ and $c_{k'}$ intersect a second time. We consider two possible cases for a second intersection and denote them by $s_2$ and $s'_2$. First assume that the intersection $s_2$ is between the bottom arc of $c_{k'}$ and the bottom arc of $c_j$. This however contradicts Lemma 6 because in $s_1$, there was already the bottom arc of $c_{k'}$,
To this end, we consider the wedge $W_{ij}$ and any $L^p$ norm with $p \in [1, \infty]$.

**Proof.** Assume for contradiction that $d(p_i, q_j) > d(p_i, q_k)$. Then, $q_k$ is below the wave front of $W_{ij}$ and, hence, $q_k$ does not lie in the valid region of $W_{ij}$ but in the valid region of $W_{ik}$. However, the valid region of $W_{ik}$ is the intersection of the local valid region of $D_{ik}$ and all previous valid regions including $W_{ij}$ and, thus, the valid region of $W_{ik}$ is a subset of $W_{ij}$.

**Lemma 14.** Any shortcut found by our algorithm is valid under the local Fréchet distance.

**Proof.** Let $(p_i, p_k)$ be a shortcut found by our algorithm. We show that there is a mapping of the vertices $p_{i+1}, p_{i+2}, \ldots, p_{k-1}$ onto points $m_{i+1}, m_{i+2}, \ldots, m_{k-1} \in \overline{p_i p_k}$ such that $d(p_j, m_j) \leq \varepsilon$ for any $j \in \{i+1, \ldots, k-1\}$ and $m_j$ precedes or equals $m_{j+1}$ for any $j \in \{i+1, \ldots, k-2\}$ when traversing $\overline{p_i p_k}$ from $p_i$ to $p_k$. Clearly, this implies that also the Fréchet distance between each pair of line segments $\overline{p_i p_{i+1}}$ and $\overline{m_j m_{j+1}}$ is at most $\varepsilon$ and, hence, $(p_i, p_k)$ is a valid shortcut.

In the remainder of this proof, we describe how to obtain $m_{i+1}, m_{i+2}, \ldots, m_{k-1} \in \overline{p_i p_k}$. To this end, we consider the wedge $W_{ij}$ and the corresponding wave front for each $j \in \{i+1, \ldots, k-1\}$, i.e., for each intermediate step when executing our algorithm. Without loss of generality, we assume that $p_i$ is the bottommost point of the wedge $W_{ij}$. By construction, $\overline{p_i p_k}$ lies inside the wedge $W_{ij}$ and $p_k$ lies above its wave front (since $p_k$ lies in the valid...
region of $W_{i(k-1)}$ and, by Lemma 13, the wave front has moved only upwards. Let $m_j$ be the intersection point of $p_kp_k$ and the wave front of $W_{ij}$. By Lemma 12, $d(m_j) \leq \varepsilon$. Moreover, by Lemma 13, $m_j$ precedes or equals $m_{j+1}$ for any $j \in \{i+1, \ldots k-2\}$ when traversing $p_kp_k$ from $p_i$ to $p_k$.

\begin{lemma}
All valid shortcuts under the local Fréchet distance and any $L_p$ norm with $p \in [1, \infty]$ are found by our algorithm.
\end{lemma}

\textbf{Proof.} Suppose for the sake of a contradiction that there is a valid shortcut $(p_i, p_k)$ that was not found by our algorithm.

If $p_k$ lay outside $W_{i(k-1)}$, then there would be some $p_j$ with $i < j < k$ such that $d(p_j, p_k) > \varepsilon$. So, as in the algorithm by Chan and Chin [7], already the Hausdorff distance requirement would be violated and $(p_i, p_k)$ would be no valid shortcut. Hence, $p_k$ lies inside $W_{i(k-1)}$.

Presume for now that $p_k$ is not in the valid region of $W_{i(k-1)}$, i.e., $p_k$ lies below the wave front of $W_{i(k-1)}$ (we assume w.l.o.g. that $p_i$ is the bottommost point of $W_{i(k-1)}$). Since $p_k$ is below the wave front, the line segment $p_kp_k$ does not intersect the wave front (otherwise we would violate Lemma 13). Now consider the ray we obtain by extending $p_kp_k$ at $p_k$. This ray intersects the wave front of $W_{i(k-1)}$ once at a point $w$. The point $w$ lies on a geometric object of the wave front, i.e., a line segment for the $L_1$ and $L_\infty$ norm or an arc for the other $L_p$ norms (in particular a circular arc for $L_2$). This geometric object is the bottom boundary of a unit circle $c_j$ (in the $L_p$ norm) of radius $\varepsilon$ that has some $p_j$ with $i < j < k$ as center point. As it is the bottom boundary, $p_k$ lies outside $c_j$ and $d(p_j, p_k) > \varepsilon$.

Therefore, $p_k$ lies inside $W_{i(k-1)}$ and above or on the wave front of $W_{i(k-1)}$, i.e., in the valid region of $W_{i(k-1)}$. However, points that fulfill these conditions, are precisely the ones for which our algorithm adds a shortcut.

\end{lemma}

\subsection{\textbf{$L_1$ and $L_\infty$ Norm}}

Under the Fréchet distance, simplifying a polyline in the $L_1$ norm (Manhattan metric) or the $L_\infty$ norm (maximum metric) is the simplest case. This can be done in $O(n^2)$ time and, hence, matching the running time of the algorithm by Chan and Chin [7], which uses the Hausdorff distance, when adjusted for the $L_1$ and the $L_\infty$ norm. This is due to the fact that the wave front consists of just up to two orthogonal line segments, which we have shown in Lemma 13.

Let us describe how to check containment in the valid region and how we can update the wedge and the wave front. Say we are computing the shortcuts originating at $p_i$ and we have reached $p_j$. We first check in constant time if $p_j$ is in the wedge $W_{i(j-1)}$ by comparing the radial angle of $p_j$ relative to $p_i$ with the radial angles of the boundaries of the wedge $W_{i(j-1)}$. Then, we check in constant time that $p_j$ is beyond both of the underlying lines of the up to two line segments of the wave front.

We update the wave front by the following operations. The unit circle around $p_j$ is bounded by four line segments that are horizontal or vertical (in the $L_\infty$ norm) or rotated by 45 degrees (in the $L_1$ norm). Therefore, they are parallel to the up to two line segments of the wave front of $W_{i(j-1)}$ (see Lemma 13) and, therefore, we can update the wedge and the wave front in constant time just by computing the intersection points of these up to six line segments.

\begin{theorem}
An $n$-vertex polyline can be simplified optimally under the local Fréchet distance in the $L_1$ and $L_\infty$ norm in $O(n^2)$ time.
\end{theorem}
Proof. According to Lemmas 14 and 15, our algorithm finds precisely the set of valid shortcuts in the polyline. For computing these shortcuts, we consider each vertex of the polyline and decide, for each subsequent vertex, in constant whether there exists a shortcut. Thus, we construct the shortcut graph in $O(n^2)$ time. In the resulting shortcut graph, we can find an optimal polyline simplification by finding a shortest path in again $O(n^2)$ time. ◀

4.2 $L_2$ Norm

Now, we consider the most natural norm that is usually used when talking about polyline simplification – the $L_2$ norm, also known as Euclidean norm. The main difference to the $L_1$ and $L_\infty$ norm is that our wave front is more complex than just two line segments.

To maintain the wave front, we use a balanced binary search tree, e.g., a red-black tree. As objects in the tree, we use the circular arcs that constitute the wave front. The high-level reason to use a balanced binary search tree is because we want to do binary search on the arcs of the wave front to find intersections with rays and unit circles and we want to update the wave front in logarithmic time. The keys according to which the circular arcs are arranged in the search tree are angles around $p_i$. More precisely, consider an arc $a$ of the wave front. The line segment between the (left) starting point of $a$ and $p_i$ has some angle $\alpha$ at $p_i$. We use this $\alpha$ as key for $a$. These angles cover a range of less than $\pi$, hence, we may rotate the drawing when computing the wave fronts for start vertex $p_i$ to avoid “jumps” from $2\pi$ to 0.

We perform the following operations in logarithmic time relative to the size of the tree (which is in $O(n)$ due to Lemma 10). One is updating the wave front by inserting or removing circular arcs with the corresponding angles as keys. The other is computing the intersection point(s) between the wave front and either a ray emanating at $p_i$ or a unit circle. The case for a ray is straightforward since a ray is described completely by its radial angle and we use a standard query on balanced binary search trees in logarithmic time. We find the arc of the wave front intersecting this ray in logarithmic time and then compute the (bottom) intersection point between the corresponding unit circle and this ray in constant time.

Finding the up to two (see Lemma 8) intersection points between the wave front and a unit circle $c$ in logarithmic time is a bit more nuanced. Hence, we describe a recursive algorithm using a case distinction in more detail in the next section and prove its correctness there. The high-level idea is as follows.

We conclude the following statement, which we prove in Section 4.2.1.

Lemma 17. Given a unit circle $c$, we can determine the up to two intersection points between $c$ and the wave front in $O(\log n)$ time in the $L_2$ norm.

Now we have all ingredients to prove our main theorem.

Theorem 18. An $n$-vertex polyline can be simplified optimally under the local Fréchet distance in the $L_2$ norm in $O(n^2 \log n)$ time.
Proof. According to Lemmas 14 and 15 our algorithm finds all valid shortcuts. It remains to analyze the runtime. We consider each of the \( n \) polyline vertices as potential shortcut starting point \( p_i \). When we encounter a vertex \( p_j > i \), we determine in logarithmic time whether it is in the valid region. We do this by computing the ray emanating at \( p_i \) and going through \( p_j \) and querying the arc it intersects in the wave front. Then, by Lemma 17, we compute the intersection point(s) of \( c_j \) with the wave front in logarithmic time (or certify that none exist). Based thereupon, we update the wave front and the wedge. As for each polyline vertex, its corresponding arc may only be inserted and removed from the wave front at most once for each \( p_i \) (at logarithmic cost each), the total time for updating the wave fronts \( W_{ti} \) to \( W_{tn} \) is in \( O(n \log n) \). Overall, we hence need \( O(n \log n) \) per polyline vertex. Accordingly, we construct the shortcut graph in \( O(n^2 \log n) \) time. In the resulting shortcut graph, we can find an optimal polyline simplification by finding a shortest path in \( O(n^2) \) time. ◀

4.2.1 Finding the Intersection Points between a Unit Circle and the Wave Front in the \( L_2 \) Norm

We now describe how to find the intersection points between the wave front and a unit circle \( c \) in logarithmic time by a recursive algorithm build upon a case distinction. By Lemma 8 the wave front and \( c \) intersect at most twice. We determine the up to two intersection points by binary search. We proceed in two steps. First, we determine the left intersection point between the wave front and \( c \) (if any), and second we determine the right intersection point (if any). Here, we describe only the procedure to determine the left intersection point. Finding the right intersection point is symmetric.

Initially, we determine the leftmost point of the wave front (i.e., the intersection point of the wedge and the leftmost arc of the wave front) and the rightmost point of the wave front and denote them by \( l_{wf} \) and \( r_{wf} \). It suffices to do this in logarithmic time, which can be done by querying the first and the last element in the search tree.

Now we traverse the balanced binary search tree recursively. In each step, we consider an arc \( a_j \) of the wave front; see Figure 9a for the base situation. Initially, we set \( a_j \) to be the root. Let the unit circle on which \( a_j \) lies be \( c_j \). Let the left and right endpoints of this arc be \( l_{a_j} \) and \( r_{a_j} \), respectively. Moreover, we determine the left and the right intersection point of the bottom arc of \( c_j \) with the wedge and denote the intersection points by \( l_j \) and \( r_j \). Next we compute the up to two intersection points of \( c \) and \( c_j \). Here, we distinguish three main cases and some subcases. In one recursion step, we only consider the first (sub)case that is true.

**Case 0:** There is no intersection point; see Figure 9b.
We can abort because if \( c \) would intersect a different arc \( a_k \) of the wave front, this would contradict Lemma 11.

**Case 1:** There is precisely one touching point \( q \); see Figure 9c.
Then, \( c \) intersects (to be more precise: touches) the wave front exactly once at \( q \) if \( q \) is in between \( l_{a_j} \) and \( r_{a_j} \) when traversing \( c \) counterclockwise. Otherwise, \( c \) does not intersect the wave front because the wave front is inside \( c_j \) by Lemma 11 (unless it touches the leftmost or rightmost arc of the search tree, which we can find in at most logarithmic time.) Again, we can abort.

**Case 2:** There are two intersection points \( q_1 \) and \( q_2 \), where \( q_1 \) is left to \( q_2 \).
First we compute the rays emanating at \( p_i \) and going through the left tangential point \( l_c \) at \( c \), through the right tangential point \( r_c \) at \( c \), through \( l_j \) and through \( r_j \). We denote these rays by \( R_{l_c}, R_{r_c}, R_{l_j}, \) and \( R_{r_j} \), respectively. Now we sort these four rays by their angle relative to \( p_i \).
Figure 9 Cases in our recursive approach to compute the left intersection point of a unit circle $c$ (green; here not always sketched in the correct size and shape) with the wave front (blue). Left intersection points of $c$ with the wave front are depicted as red crosses.
**Case 2.1:** The sorted order of rays is $R_l, R_r, R_l, R_r$; see Figure 9d.
In this case, we recurse at the left child of $a_j$ in the search tree because clearly if there are any intersection of $c$ with the wave front, they are in between $R_l, R_r$.

**Case 2.2:** The sorted order of rays is $R_l, R_r, R_l, R_r$.
Symmetrically, we recurse at the right child of $a_j$ in the search tree.

Next we check if the intersection points of $c$ with the wave front lie on $a_j$.

**Correctness**
Clearly, this recursive algorithm runs in logarithmic time and returns an intersection point or reports that there is no intersection point. We have already argued, why there is no intersection point once we have reached Case 0 or Case 1. The behavior of the algorithm in the Cases 2.1, 2.2, and 2.3 is trivially correct. Obviously, if leftentry has an odd number of intersections with $c$, then there is exactly one intersection between the wave front and $c$ to the left of $a_j$ because for each time $c$ enters the region between leftentry and the wave front, it also has to leave it. (This holds symmetrically for rightentry.) Therefore, the behavior in the Cases 2.4 and 2.5 is also correct.
Polyline Simplification under the Local Fréchet Distance has Subcubic Complexity in 2D

It remains to prove that in Cases 2.6, 2.7, and 2.8 the algorithm behaves correctly. First, we show that if there is no intersection in leftentry, there also cannot be an intersection point of c and the wave front to the left of a_j. This shows that Case 2.8 is correct.

**Lemma 19.** Consider an arc a_j of the wave front and a unit circle c. If there is an intersection point of c with the wave front to the left of a_j, then leftentry contains an intersection point with c or the sorted order of rays is R_l c, R_r c, R_l j, R_r j.

**Proof.** Assume for contradiction that there is an intersection point s_1 of c with the wave front to the left of a_j, but leftentry contains no intersection point with c and the sorted order of rays is different from R_l c, R_r c, R_l j, R_r j. For this situation, see Figure 10a.

Of course, there is a second intersection point s_2 (to the right of s_1) between c and the wave front left of a_j as otherwise c would enter, but never leave the region between leftentry and the wave front. Since R_l always precedes R_r and R_l j always precedes R_r j, in any permutation different from R_l c, R_r c, R_l j, R_r j, R_l j precedes R_r c. Thus, for c to have any intersection point with the wave front, the bottom arc of c needs to intersect the wave front at least once – this intersection is at s_2. This, however, contradicts Lemma 7 because at the right side of s_2, c is above the wave front.

Second, we show that if there are two intersection points of c with the wave front on one side of a_j, then on the other side there is no intersection point of c with leftentry/rightentry. This shows the correctness of the Cases 2.6 and 2.7.

**Lemma 20.** Consider an arc a_j of the wave front and a unit circle c. If there are two intersection points of c with the wave front to the right of a_j, then leftentry contains no intersection point with c.

**Proof.** Assume for contradiction that there are two intersection points of c with the wave front to the right of a_j, but leftentry contains intersection points with c. For this situation, see Figure 10b. Clearly, leftentry contains precisely two intersection points with c because if it had an odd number of intersection points, there would be an intersection with the wave front to the left of a_j. There are four possible intersections of c with leftentry, namely, q_3 and q_4 on the boundary of the wedge (q_3 with c’s top arc, q_4 with c’s bottom arc), and q_1 and q_2 on c_j. However, it cannot be both q_1 and q_2 because all of leftentry is bounded by the bottom arc of c_j and c needs to intersect c_j also at least once with its bottom arc, which contradicts Lemma 6. Also, it cannot be both q_1 and q_2 because when traversing...
c counterclockwise, c leaves the region between leftentry and the wave front at q3 and then immediately afterwards re-enters this region at q4 without any chance of leaving it again. Hence, there are the intersection points q1 and q4.

By Lemma [19] we know that there are also two intersection points of c with rightentry. They cannot be both with the right boundary of the wedge because then c would need a third intersection with the wave front to connect its bottom arc to q1.

This immediately yields our main statement of this section, which we have stated before.

Lemma 17. Given a unit circle c, we can determine the up to two intersection points between c and the wave front in O(log n) time in the L2 norm.

4.3 Lp Norm

We can use all of our data structures for the L2 norm also for the Lp norm for p ∈ (1, ∞). However, we should take this with a grain of salt as computing the intersection points between lines and unit circles and between pairs of unit circles in the Lp norm for p ∈ (1, ∞) \ {2} may involve solving equations of degree p, which may raise numerical questions for the required precision. To avoid this, we assume here that we can determine intersection points between unit circles and lines (or another unit circle) in all Lp norms in constant time.

Corollary 21. An n-vertex polyline can be simplified optimally under the local Fréchet distance in the Lp norm for p ∈ (1, ∞) in O(n2 log n) time under the assumption that we can compute the intersection point of a line and a unit circle in the Lp norm, as well as the intersection point of two unit circles in the Lp norm in constant time.

5 Conclusion and Open Problems

We have presented an algorithm that computes optimal polyline simplifications under the local Fréchet distance in subcubic time, based on the introduced concept of a wave front. Our theoretical analysis shows that the wave front has constant complexity for L1 and L∞, but might have up to linear complexity for Lp ∈ (1, ∞), which for the latter results in an additional logarithmic factor in the respective running time. However, for a large wave front to arise, the polyline vertices need to form a specific pattern. Since it is unlikely that such patterns occur naturally, we actually expect the wave front to have constant size in practice, resulting in O(n2) time also for L2. It would be interesting to validate this claim empirically. Furthermore, the investigation of lower bounds could shed light on the question whether our upper bounds are tight. Existing lower bounds only apply to simplification of polylines in high dimension. For the practically most relevant use case of two dimensions no (conditional) lower bounds are known, though. Another direction for future work would be to generalize our algorithm to work in higher dimensions, which requires dealing with a more complex wave front. Finally, one could also consider further distance measures, as e.g. the Fréchet distance under the Lp norm for p ∈ (0, 1). But there, the respective unit circles are not convex anymore which could make updating the wave front data structure more expensive.

References

1 Pankaj K. Agarwal, Sariel Har-Peled, Nabil H. Mustafa, and Yusu Wang. Near-linear time approximation algorithms for curve simplification. Algorithmica, 42(3):203–219, 2005. doi:10.1007/s00453-005-1165-y
Polyline Simplification under the Local Fréchet Distance has Subcubic Complexity in 2D

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

3 Gill Barequet, Danny Z Chen, Ovidiu Daescu, Michael T Goodrich, and Jack Snoeyink. Efficiently approximating polygonal paths in three and higher dimensions. *Algorithmica*, 33(2):150–167, 2002. doi:10.1007/s00453-001-0096-5

4 Sergey Bereg, Minghui Jiang, Wencheng Wang, Boting Yang, and Binhai Zhu. Simplifying 3D polygonal chains under the discrete Fréchet distance. In *Proc. 8th Latin American Symposium on Theoretical Informatics (LATIN’08)*, pages 630–641. Springer, 2008. doi:10.1007/978-3-540-78773-0_54

5 Karl Bringmann and Bhaskar Ray Chaudhury. Polyline simplification has cubic complexity. In *Proc. 35th International Symposium on Computational Geometry (SoCG’19)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.SoCG.2019.18

6 Kevin Buchin, Maike Buchin, Maximilian Konzack, Wolfgang Mulzer, and André Schulz. Fine-grained analysis of problems on curves. In *Proc. 32nd European Workshop on Computational Geometry (EuroCG’16)*, 2016. URL: https://www.eurocg2016.usi.ch/sites/default/files/paper_68.pdf

7 W. S. Chan and F. Chin. Approximation of polygonal curves with minimum number of line segments or minimum error. *International Journal of Computational Geometry and Applications*, 6(1):59–77, 1996. doi:10.1142/S0218195996000058

8 David H. Douglas and Thomas K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica*, 10(2):112–122, 1973. doi:10.1002/9780470669488.ch2

9 Michael Godau. A natural metric for curves – computing the distance for polygonal chains and approximation algorithms. In *Proc. 8th Annual Symposium on Theoretical Aspects of Computer Science (STACS’91)*, pages 127–136, 1991. doi:10.1007/BFb0020793

10 Hiroshi Imai and Masao Iri. Polygonal approximations of a curve – formulations and algorithms. In Godfried T. Toussaint, editor, *Computational Morphology*, volume 6 of *Machine Intelligence and Pattern Recognition*, pages 71–86. North-Holland, 1988. doi:10.1016/B978-0-444-70467-2.50011-4

11 Marc J. van Kreveld, Maarten Löffler, and Lionov Wiratma. On optimal polyline simplification using the Hausdorff and Fréchet distance. *Journal of Computational Geometry*, 11(1):1–25, 2020. doi:10.20382/jocg.v11i1a1