Discrete Comput Geom 23:273–291 (2000)
DOI: 10.1007/s004549910018

Discrete & Computational
Geometry

Efficient Algorithms for Approximating Polygonal Chains*

P. K. Agarwal[1] and K. R. Varadarajan[2]

[1] Center for Geometric Computing,
Department of Computer Science, Duke University,
Durham, NC 27708-0129, USA
pankaj@cs.duke.edu

[2] DIMACS, Rutgers University,
Piscataway, NJ 08854, USA
krv@dimacs.rutgers.edu

Abstract. We consider the problem of approximating a polygonal chain C by another polygonal chain C' whose vertices are constrained to be a subset of the set of vertices of C. The goal is to minimize the number of vertices needed in the approximation C'. Based on a framework introduced by Imai and Iri [25], we define an error criterion for measuring the quality of an approximation. We consider two problems. (1) Given a polygonal chain C and a parameter $\varepsilon \geq 0$, compute an approximation of C, among all approximations whose error is at most ε, that has the smallest number of vertices. We present an $O(n^{4/3+\delta})$-time algorithm to solve this problem, for any $\delta > 0$; the constant of proportionality in the running time depends on δ. (2) Given a polygonal chain C and an integer k, compute an approximation of C with at most k vertices whose error is the smallest among all approximations with at most k vertices. We present a simple randomized algorithm, with expected running time $O(n^{4/3+\delta})$, to solve this problem.

1. Introduction

Let $C = \langle p_1, \ldots, p_n \rangle$ denote a polygonal chain whose vertices are $p_1, \ldots, p_n$ and whose edges are $p_1p_2, p_2p_3, \ldots, p_{n-1}p_n$. A polygonal chain $C' = \langle p_{i_1} = p_1, \ldots, p_{i_k} = p_n \rangle$ is an *approximation* of C if $i_1 < \cdots < i_k$ (Fig. 1).

* Work on this paper has been supported by an NSF Grant CCR-93-01259, an Army Research Office MURI Grant DAAH04-96-1-0013, an NYI award, and matching funds from the Xerox Corporation.

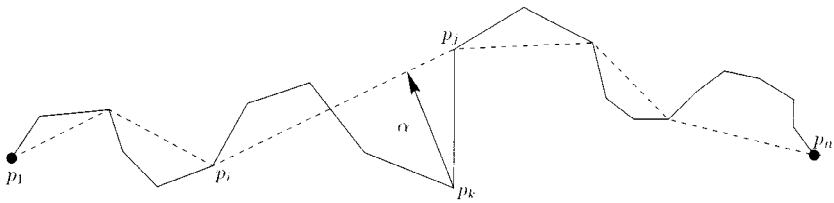

Fig. 1. A polygonal chain (in bold) and its approximation (dashed); the error, under the Euclidean metric, of segment $p_i p_j$ is α.

Let $d(\cdot, \cdot)$ be a distance function defined on points in the plane. We define the *error of a line segment* $p_i p_j$ connecting two points of C, where $i < j$, to be

$$\Delta(p_i p_j) = \max_{i \le k \le j} d(p_k, p_i p_j),$$

where the distance between a point p and a segment e is $d(p, e) = \min_{q \in e} d(p, q)$. The *error of an approximation* C' of C is

$$\Delta_C(C') = \max_{1 \le j < k} \Delta(p_{i_j} p_{i_{j+1}}),$$

the maximum of the errors of each of the edges of C'. We will omit the subscript for Δ when the chain C that is being approximated is clear from the context. We call C' an ε-approximation of C if $\Delta_C(C') \le \varepsilon$. We consider two problems:

1. **Min-# problem:** Given a polygonal chain C and a real number $\varepsilon \ge 0$, compute an ε-approximation of C that uses the the smallest number of vertices among all ε-approximations of C.
2. **Min-ε problem:** Given a polygonal chain C and an integer k, compute an approximation of C with at most k vertices that minimizes the error over all approximations of C that have at most k vertices.

1.1. *Motivation and Previous Results*

Polygonal curves are often used to represent boundaries of planar objects in cartography, computer graphics, pattern recognition, etc. The general problem of approximating a polygonal curve by a coarser one is of fundamental importance, and has been studied in disciplines such as geographic information systems [7], [11], [15], [21], [29], [31], digital image analysis [5], [24], [28], and computational geometry [8], [16], [18], [19], [25], [33], [35]. The problems considered fall in one of two categories, depending on whether the vertices of the approximating chain are required to be a subset of the vertices of the input chain. Within each category, min-# and min-ε problems have been studied under different error criteria.

We first mention some of the algorithms that compute approximations whose vertices are not restricted to being on the input chain. A well-studied problem in this category is to compute a piecewise-linear function approximating an n-point planar data set, using

the uniform metric to measure the error. This is in fact a special case of a whole class of problems in approximation theory that seek to fit a set of data using a spline function under some metric. (See texts discussing approximation theory such as [6], [9], [12], and [14].) Hakimi and Schmeichel [20] give an $O(n)$-time algorithm for the min-# version of the problem and an $O(n^2 \log n)$-time algorithm for the min-ε version. Goodrich [18] gives an improved $O(n \log n)$-time algorithm for the min-ε version. The min-# problems in this category closely resemble the problem of computing a minimum link path in a polygonal domain, which has been studied extensively; see [3], [19], [22], and [25] for a sample. Hershberger and Snoeyink [22] give an efficient algorithm for computing a minimum link path of a given homotopy. In this problem we are given two points s and t in a polygonal region R, which may have holes, and a representative path π between s and t in R, and we want to compute a minimum-link path in R between s and t that is *homotopic* to π. Homotopy considerations are important when polygonal approximation occurs in the context of other features; see the paper by de Berg et al. [13]. Sometimes, we also want the approximation of a given polygonal chain to be *simple*, that is, non-self-intersecting. This requirement seems to make the problems intractable. Guibas et al. [19] show that the problem of computing a minimum-link simple approximation of a simple polygon is NP-hard (for a particular error criterion). They also show that a version of the problem of approximating a polygonal subdivision (as opposed to just a single polygonal chain) is NP-hard.

We now turn to the methods for computing approximations whose vertices are restricted to being a subset of the vertices of the input chain. This is the class to which this paper belongs. The restricted case is important because in many applications, attributes such as color are associated with the vertices of the input chain. Moreover, in the unrestricted case, we have to deal with the issue of the precision needed to represent vertices in the approximation (see [26]). One of the oldest and most popular algorithms in cartography is the Douglas–Peucker algorithm [15], a heuristical approach that computes an approximation within a prespecified error. Hershberger and Snoeyink [23] showed that the Douglas–Peucker algorithm can be implemented in $O(n \log^* n)$ time, where $\log^*$ is the iterated logarithm function [10]. Imai and Iri [25] present a unified approach to polygonal approximation problems by formulating them in terms of graph theory, and present algorithms for approximating simple polygonal chains under a number of error criteria. Algorithms with the same flavor are also described by Chan and Chin [8], Melkman and O'Rourke [33], and Toussaint [36]. Most of these algorithms run in $\Omega(n^2)$ time. It appears as though the problems in this category become intractable if we require the approximation to be simple, or to be homotopic to the original chain; see [13] for a discussion.

1.2. *Our Results*

In this paper we study the min-# and min-ε problems under the L_1 and uniform (also known as Chebyshev) metric. Recall that the distance between two points $p = (p_x, p_y)$ and $q = (q_x, q_y)$ in the L_1 metric is

$$d(p, q) = |p_x - q_x| + |p_y - q_y|,$$

and the distance under the uniform metric is

$$d(p, q) = \begin{cases} |p_y - q_y| & \text{if} \quad p_x = q_x, \\ \infty & \text{otherwise.} \end{cases}$$

Note that the distance between a point p and a segment e under the uniform metric is the vertical distance between p and e if the x-projections of p and e intersect, and infinity otherwise. The two main results are the following:

- An $O(n^{4/3+\delta})$-time deterministic algorithm, for any $\delta > 0$, for the min-# problem under the L_1 metric.
- An $O(n^{4/3+\delta})$ expected-time randomized algorithm, for any $\delta > 0$, for the min-ε problem under the L_1 metric.

Our techniques also generalize to certain other metrics that approximate the Euclidean metric closely. Section 2 discusses preliminaries needed by our algorithms and the main idea used. In order to present the techniques clearly, we describe in Section 3 the min-# algorithm for an x-monotone chain under the uniform metric. In Section 4 we show how the techniques used in Section 3 can be generalized to solve the min-# problem for any polygonal chain under the L_1 metric. In Section 5 we describe a simple randomized algorithm for the min-ε problem under the L_1 metric. We offer our conclusions in Section 6.

2. Preliminaries

Imai and Iri's Algorithm. Since our algorithm is based on Imai and Iri's general framework for polygonal chain approximation, we first describe their approach. Let C be the polygonal chain $\langle p_1, \ldots, p_n \rangle$, and let $\varepsilon > 0$ be a given error bound. Assume that we have fixed a distance function on points in the plane. We define an unweighted, directed graph $G_\varepsilon(C) = (V, E_\varepsilon)$, where $V = \{p_1, \ldots, p_n\}$ and

$$E_\varepsilon = \{(p_i, p_j) \mid 1 \leq i < j \leq n, \text{ and } \Delta(p_i p_j) \leq \varepsilon\}.$$

Figure 2 illustrates this definition for an x-monotone chain under the uniform metric. We sometimes denote $G_\varepsilon(C)$ by G_ε when the underlying chain C is clear from the context.

An ε-approximation of C corresponds to a path from p_1 to p_n in G_ε, and an ε-approximation with the minimum number of vertices corresponds to a shortest path

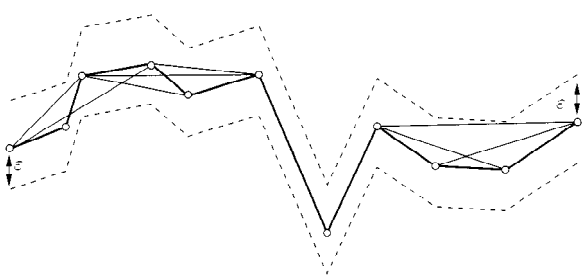

Fig. 2. The graph G_ε for an x-monotone chain under the uniform metric; solid segments are graph edges.

from p_1 to p_n in G_ε. The shortest path, which is the solution to the min-# problem, can be found by a breadth first search in G_ε using $O(|E_\varepsilon|) = O(n^2)$ time. The min-ε problem involves computing the smallest value of ε for which there is a path in G_ε from p_1 to p_n consisting of at most k vertices. Let ε^* denote this smallest value. Let $\varepsilon_1 \leq \varepsilon_2 \cdots$ be the values of $\Delta(p_i p_j)$ for $1 \leq i < j \leq n$. Note that G_ε remains the same for all $\varepsilon \in [\varepsilon_i, \varepsilon_{i+1})$. Therefore, the min-$\varepsilon$ problem reduces to finding the smallest ε_i for which G_{ε_i} contains a path from p_1 to p_n with at most k vertices. Since $E_{\varepsilon_i} \subseteq E_{\varepsilon_{i+1}}$, ε^* can be computed by a binary search on the ε_i's, using the min-# algorithm as the decision procedure.

Using Imai and Iri's approach naively, any algorithm for the min-# problem takes $\Omega(n^2)$ time, as it needs to construct the graph G_ε, which can have $\Theta(n^2)$ edges. Chan and Chin [8], Imai and Iri [25], Melkman and O'Rourke [33], and Toussaint [36] give quadratic or near-quadratic-time algorithms for constructing G_ε under various error criteria. The running time for the min-ε problems is governed by the time it takes to compute the errors $\Delta(p_i p_j)$ of the segments, and the time for $O(\log n)$ applications of the min-# decision procedure. Hence, these algorithms also take $\Omega(n^2)$ time.

Compact Representation of G_ε. If we are aiming for a subquadratic algorithm, we cannot compute the graph G_ε explicitly. Instead, we construct a compact representation of G_ε. Let $G = (V, E)$ be a directed graph, and let

$$\mathcal{G} = \{G_1 = (V_1, E_1), \ldots, G_l = (V_l, E_l)\}$$

be a family of subgraphs of G. We say that $\mathcal{G}$ is a *clique cover* of G if the following conditions hold:

1. Each G_i is a complete bipartite graph. If $V_{i_1}, V_{i_2} \subseteq V_i$ are the two vertex classes of G_i, then every edge of G_i must be directed from a vertex in V_{i_1} to a vertex in V_{i_2}.
2. $E = E_1 \cup \cdots \cup E_l$.
3. $E_i \cap E_j = \emptyset$ for $i \neq j$.

Since each G_i is a bipartite clique, we can represent it compactly by specifying its vertex classes V_{i_1} and V_{i_2}; this takes $O(|V_i|)$ space. The edges are now defined implicitly. We define the size of the clique cover $\mathcal{G}$, denoted by $|\mathcal{G}|$, to be $\sum_{i=1}^{l} |V_i|$; this is the space we require to represent G compactly. The notion of "compressing" graphs using clique covers is pursued by Feder and Motwani [17], who use clique covers of graphs to speed up a number of graph algorithms.

Proposition 2.1. *Let $\mathcal{G} = \{G_1, \ldots, G_l\}$ be a clique cover of a directed graph G. Given two vertices u and v in G, we can compute a shortest path between u and v in G in $O(|\mathcal{G}| + |V|)$ time.*

Proof. We define a new graph H whose vertex set is $V \cup \{g_1, \ldots, g_l\}$, where $g_1, \ldots, g_l$ are additional vertices. If (u, v) is an edge in G, and if (u, v) belongs to G_i, then we add the edges (u, g_i) and (g_i, v) in H. There are $O(|\mathcal{G}| + |V|)$ vertices and $O(|\mathcal{G}|)$ edges in H. We compute a shortest path between u and v in H. If the procedure returns the path $\langle u_1 = u, g_1, u_2, g_2, \ldots, g_k, u_{k+1} = v\rangle$ in H, then we return $\langle u_1, u_2, \ldots, u_{k+1}\rangle$ as the

shortest path between u and v in G. The correctness of the procedure is straightforward. This computation takes time proportional to $|\mathcal{G}| + |V|$. □

3. The Min-# Algorithm under the Uniform Metric

Let $C = \langle p_1, \ldots, p_n \rangle$ be an x-monotone polygonal chain, and let ε be a given error bound. In this section we present an algorithm that solves the min-# problem for C under the uniform metric. Our algorithm computes a shortest path between p_1 and p_n in the graph G_ε defined above. In order to do this efficiently, we first compute a clique cover of G_ε, and then use Proposition 2.1 to compute a shortest path from p_1 to p_n in G_ε.

In the remainder of this section we describe an $O(n^{4/3+\delta})$-time divide-and-conquer algorithm for constructing a clique cover $\mathcal{G}$ of $G_\varepsilon = G_\varepsilon(C)$. Let C_1 be the chain $\langle p_1, \ldots, p_{\lfloor n/2 \rfloor} \rangle$ and let C_2 be the chain $\langle p_{\lfloor n/2 \rfloor + 1}, \ldots, p_n \rangle$. Recall that $\Delta(p_i p_j) = \max_{i \le k \le j} d(p_k, p_i p_j)$, and that a pair $(p_i, p_j) \in E_\varepsilon$ if $\Delta(p_i p_j) \le \varepsilon$. Hence, if $(p_i, p_j) \in E_\varepsilon$ and $p_i, p_j \in C_1$ (resp. $p_i, p_j \in C_2$), then (p_i, p_j) is an edge in $G_\varepsilon(C_1)$ (resp. $G_\varepsilon(C_2)$). We recursively compute clique covers $\mathcal{G}_1, \mathcal{G}_2$ of $G_\varepsilon(C_1)$ and $G_\varepsilon(C_2)$, respectively. In the merge step we compute a clique cover $\mathcal{G}_{12}$ of the edges

$$E_{12} = \{(p_i, p_j) \in G_\varepsilon(C) \mid p_i \in C_1, p_j \in C_2\}.$$

That is, $\mathcal{G}_{12}$ is a clique cover of the edges of $G_\varepsilon(C)$, one of whose endpoints is in C_1 and the other in C_2. $\mathcal{G}_1 \cup \mathcal{G}_2 \cup \mathcal{G}_{12}$ is a clique cover of $G_\varepsilon(C)$. Before describing how to compute $\mathcal{G}_{12}$, we need a few definitions and preliminary lemmas.

If (x_i, y_i) denotes the coordinates of p_i, let p_i^- be the point $(x_i, y_i - \varepsilon)$, and let p_i^+ be the point $(x_i, y_i + \varepsilon)$. It is easy to see that (p_i, p_j) is an edge of $G_\varepsilon(C)$ if and only if for every k, $i \le k \le j$, the vertical segment $p_k^- p_k^+$ intersects the segment $p_i p_j$. For $p_i \in C_1$, we define $cone(p_i)$ to be the following subset of rightward directed rays emanating from p_i: a ray ρ belongs to $cone(p_i)$ if ρ intersects $p_k^- p_k^+$, for every $i \le k \le \lfloor n/2 \rfloor$. (Figure 3 illustrates the definition of $cone(p_i)$.) We regard $cone(p_i)$ not only as a set of rays, but as a set of points as well. Note that the interior of $cone(p_i)$ does not contain any p_k^-, p_k^+, for any $i \le k \le \lfloor n/2 \rfloor$, and for any ray $\rho \in cone(p_i)$, $d(p_k, \rho) \le \varepsilon$, for $i \le k \le \lfloor n/2 \rfloor$. Symmetrically, for a vertex p_j of C_2, we define $cone(p_j)$ to be the following collection of leftward directed rays emanating from p_j: a ray ρ belongs to $cone(p_j)$ if it intersects $p_k^- p_k^+$, for $\lfloor n/2 \rfloor + 1 \le k \le j$.

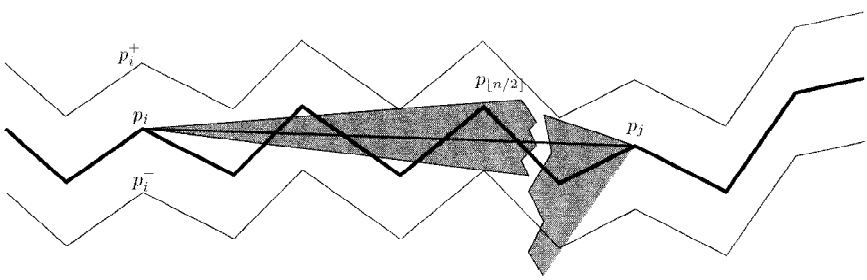

Fig. 3. $cone(p_i)$ and $cone(p_j)$; the edge (p_i, p_j) is in G_ε because $p_j \in cone(p_i)$ and $p_i \in cone(p_j)$

The boundary of a convex polygon Q can be divided into two x-monotone convex chains at its leftmost and rightmost vertices. The chain lying above (resp. below) the line through the x-extremal vertices is called the *upper* (resp. *lower*) boundary of Q. Let L_i denote the lower boundary of the convex hull of $\{p_i^+, \ldots, p_{\lfloor n/2 \rfloor}^+\}$ and U_i the upper boundary of the convex hull of $\{p_i^-, \ldots, p_{\lfloor n/2 \rfloor}^-\}$. By definition of $cone(p_i)$, its top ray is tangent to L_i and its lower ray is tangent to U_i. Note that $cone(p_i)$ is empty if the tangent to L_i is below the tangent to U_i.

Lemma 3.1. *We can compute $cone(p_k)$, for every $1 \leq k \leq n$, in $O(n \log n)$ time.*

Proof. We only describe the computation of $cone(p_k)$ for $1 \leq k \leq \lfloor n/2 \rfloor$; we can compute $cone(p_k)$ for $\lfloor n/2 \rfloor + 1 \leq k \leq n$ symmetrically. Suppose that we have inductively computed $cone(p_k)$ for $i \leq k \leq n/2$. Suppose also that we have computed L_i and U_i. We first compute L_{i-1} and U_{i-1} from L_i and U_i, respectively, in $O(\log n)$ time using standard techniques [32], [34]. We can then compute in $O(\log n)$ time the tangents to L_{i-1} and U_{i-1} from p_{i-1}, and thus $cone(p_{i-1})$. □

Using a standard duality transformation (that maps a point (a, b) to the line $y = ax + b$, and a line $y = mx + c$ to the point $(-m, c)$), we can map the line supporting any ray ρ to a point ρ^*. We refer to the point ρ^* as the *dual* of the ray ρ. Let γ_i denote the set of points dual to the rays in $cone(p_i)$. γ_i is a line segment in the dual plane. Let

$$\Gamma_1 = \{\gamma_i \mid 1 \leq i \leq \lfloor n/2 \rfloor\}$$

and

$$\Gamma_2 = \{\gamma_i \mid \lfloor n/2 \rfloor + 1 \leq i \leq n\}.$$

The following lemma is obvious.

Lemma 3.2. *Let p_i (resp. p_j) be a vertex of C_1 (resp. C_2). Then $p_j \in cone(p_i)$ and $p_i \in cone(p_j)$ if and only if γ_i and γ_j intersect.*

Lemma 3.3. *Let p_i (resp. p_j) be a vertex of C_1 (resp. C_2). Then $(p_i, p_j) \in G_\varepsilon$ if and only if γ_i and γ_j intersect.*

Proof. Suppose that γ_i and γ_j intersect. By Lemma 3.2, it follows that $p_j \in cone(p_i)$ and $p_i \in cone(p_j)$. Thus the ray $R(p_i, p_j)$ emanating from p_i and passing through p_j is in $cone(p_i)$. Therefore, this ray intersects the segment $p_k^- p_k^+$, for $i \leq k \leq \lfloor n/2 \rfloor$. Since the chain C is x-monotone, the line segment $p_i p_j$ intersects the segment $p_k^- p_k^+$, for $i \leq k \leq \lfloor n/2 \rfloor$. By a similar argument, we can conclude that the line segment $p_i p_j$ also intersects the segments $p_k^- p_k^+$, for $\lfloor n/2 \rfloor + 1 \leq k \leq j$. Thus, the line segment $p_i p_j$ intersects all the segments $p_k^- p_k^+$, for $i \leq k \leq j$, which means that $(p_i, p_j) \in G_\varepsilon$.

To prove the converse, assume that $(p_i, p_j) \in G_\varepsilon$. Then the line segment $p_i p_j$ intersects all the segments $p_k^- p_k^+$, for $i \leq k \leq j$. It follows that the ray $R(p_i, p_j)$ emanating from p_i and passing through p_j intersects all the segments $p_k^- p_k^+$, for $i \leq k \leq \lfloor n/2 \rfloor$. Thus the ray $R(p_i, p_j)$ is in $cone(p_i)$, and so $p_j \in cone(p_i)$. By a similar argument, we conclude that $p_i \in cone(p_j)$. Lemma 3.2 then implies that γ_i and γ_j intersect. □

By Lemma 3.3, the problem of computing $\mathcal{G}_{12}$ reduces to computing a family

$$\mathcal{F} = \{(\Gamma_{11}, \Gamma_{21}), \ldots, (\Gamma_{1u}, \Gamma_{2u})\}$$

so that the following conditions hold:

1. $\Gamma_{1i} \subseteq \Gamma_1$, $\Gamma_{2i} \subseteq \Gamma_2$;
2. each segment in Γ_{i1} intersects every segment in Γ_{i2}; and
3. for every pair of intersecting segments $\gamma_1 \in \Gamma_1$, $\gamma_2 \in \Gamma_2$ there is a unique i such that $\gamma_1 \in \Gamma_{1i}$ and $\gamma_2 \in \Gamma_{2i}$.

We now describe a procedure to compute such a family $\mathcal{F}$. We construct the segment-intersection-searching data structure [2] on the set Γ_2, which can report the set of segments of Γ_2 intersecting a query segment in the plane, as a union of few pairwise disjoint subsets of Γ_2. This segment-intersection-searching data structure is a multilevel partition tree, each of whose nodes is associated with a so-called *canonical subset* of Γ_2. The total size of all canonical subsets in the tree is $O(n^{4/3+\delta})$. For a query segment e, the query procedure selects $O(n^{1/3+\delta})$ pairwise disjoint canonical subsets whose union consists of exactly those segments Γ_2 that intersect e. Using this structure, we can construct the family $\mathcal{F}$ as follows. We query the data structure with all segments of Γ_1. For each canonical subset Γ_{2i} of Γ_2, let $\Gamma_{1i} \subseteq \Gamma_1$ be the set of segments whose output contained Γ_{2i}. If $\Gamma_{1i} \neq \emptyset$, we add the pair $(\Gamma_{1i}, \Gamma_{2i})$ to the family $\mathcal{F}$.

The size of the resulting clique cover $\mathcal{G}_{12}$ of E_{12} is $\sum_i (|\Gamma_{1i}| + |\Gamma_{2i}|)$, which is $O(n^{4/3+\delta})$. The running time for computing $\mathcal{G}_{12}$ is dominated by the time to compute the family $\mathcal{F}$, which is $O(n^{4/3+\delta})$.

Let $S(n)$ denote the size of the clique cover of $G_\varepsilon(C)$ computed by the entire algorithm. Then we have the recurrence

$$S(n) \leq 2S(n/2) + cn^{4/3+\delta},$$

for some constant c. This recurrence solves to $S(n) = O(n^{4/3+\delta})$. An identical argument shows that the running time of the algorithm is $O(n^{4/3+\delta})$.

Lemma 3.4. *For any $\delta > 0$, we can compute a clique cover of $G_\varepsilon(C)$ of size $O(n^{4/3+\delta})$ in $O(n^{4/3+\delta})$ time.*

From Proposition 2.1, it follows that we can compute a shortest path between p_1 and p_n in G_ε in $O(n^{4/3+\delta})$ time. Thus, we have the following result.

Theorem 3.5. *For any $\delta > 0$, we can solve the min-# problem under the uniform metric for an x-monotone chain with n vertices in $O(n^{4/3+\delta})$ time.*

4. The Min-# Algorithm under the L_1 Metric

Let $C = \langle p_1, \ldots, p_n \rangle$ be a polygonal chain (which is not necessarily x-monotone), and let ε be a given error bound. In this section we take the underlying metric to be the L_1 metric, and show how we can efficiently compute a clique cover of the graph G_ε defined

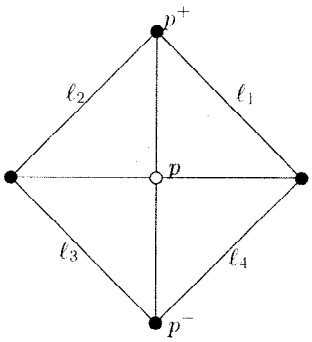

Fig. 4. The L_1 disk of radius ε centered at p. $\Phi(\ell_1) = \Phi(\ell_3) = -\pi/4$ and $\Phi(\ell_2) = \Phi(\ell_4) = \pi/4$. Any line ℓ, with $-\pi/4 \le \Phi(\ell) \le \pi/4$, that intersects the disk intersects the vertical segment $p^- p^+$.

on C. We then use Proposition 2.1 to compute a shortest path between p_1 and p_n in G_ε, thus solving the min-# problem. Throughout this section, $d(\cdot, \cdot)$ denotes the distance under the L_1 metric.

We first give some definitions. We define the *orientation* of a line ℓ, which we denote by $\Phi(\ell)$, to be the angle that ℓ makes with the positive x-axis in the range $[-\pi/2, \pi/2]$. We define the orientation of a line segment pq, which we denote by $\Phi(pq)$, to be the orientation of the line containing the line segment. Let $x(p)$ (resp. $y(p)$) denote the x-coordinate (resp. y-coordinate) of a point p. The following two lemmas state useful properties of the L_1 metric; see Figure 4.

Lemma 4.1. *Let ℓ be a line such that $-\pi/4 \le \Phi(\ell) \le \pi/4$. For any point p, $d(p, \ell) \le \varepsilon$ under the L_1-metric if and only if ℓ intersects the vertical segment $p^+ p^-$.*

Lemma 4.2. *Let qr be a line segment such that $-\pi/4 \le \Phi(qr) \le \pi/4$, and assume q lies to the left of r. For any point p, $d(p, qr) \le \varepsilon$ if and only if exactly one of the following three conditions hold*:

(i) $x(p) < x(q)$ *and* $d(p, q) \le \varepsilon$.
(ii) $x(p) > x(r)$ *and* $d(p, r) \le \varepsilon$.
(iii) $x(q) \le x(p) \le x(r)$ *and qr intersects the vertical segment $p^+ p^-$.*

As in Section 3, we use a divide-and-conquer approach to compute a clique cover of $G_\varepsilon(C)$. Let C_1 be the chain $\langle p_1, \ldots, p_{\lfloor n/2 \rfloor} \rangle$, and let C_2 be the chain $\langle p_{\lfloor n/2 \rfloor + 1}, \ldots, p_n \rangle$. We recursively compute the clique covers $\mathcal{G}_1$ and $\mathcal{G}_2$ of $G_\varepsilon(C_1)$ and $G_\varepsilon(C_2)$, respectively. The merge step computes a clique cover $\mathcal{G}_{12}$ of the set

$$E_{12} = \{(p_i, p_j) \in G_\varepsilon \mid p_i \in C_1,\ p_j \in C_2\}.$$

We will describe an algorithm for computing a clique cover of the set of edges

$$H_{12} = \{(p_i, p_j) \in E_{12} \mid x(p_i) \le x(p_j),\ -\pi/4 \le \Phi(p_i p_j) \le \pi/4\}.$$

By reversing the direction of the $(+x)$-axis and/or switching the role of the x- and y-axis, we can compute a clique cover of the remaining edges of E_{12}. Our approach will

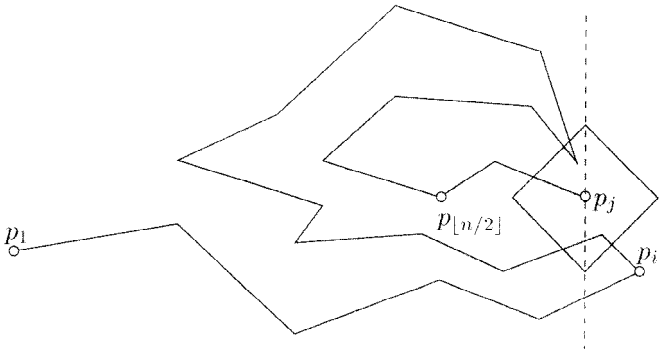

Fig. 5. $p_{i+1}, \ldots, p_{\lfloor n/2 \rfloor}$ are relevant for p_j; $p_1, \ldots, p_{\lfloor n/2 \rfloor}$ are not relevant for p_j.

be to do some "filtering" so that we can compute a clique cover of H_{12} by essentially the technique of Section 3. Let $\bar{H}_{12}$ denote the following collection of pairs:

$$\bar{H}_{12} = \{(p_i, p_j) \mid p_i \in C_1,\ p_j \in C_2,\ x(p_i) \le x(p_j),\ -\pi/4 \le \Phi(p_i p_j) \le \pi/4\}.$$

By definition, $H_{12} \subseteq \bar{H}_{12}$. We first compute a clique cover of $\bar{H}_{12}$, and then use that to compute a clique cover of H_{12}.

Definition 4.3. Let $p_i \in C_1$ and $p_j \in C_2$. We say that p_i *is relevant for* p_j, if for all p_k, $i \le k \le \lfloor n/2 \rfloor$, such that $x(p_k) > x(p_j)$, $d(p_k, p_j) \le \varepsilon$. Similarly, we say that p_j *is relevant for* p_i if for all p_k, $\lfloor n/2 \rfloor + 1 \le k \le j$, such that $x(p_k) < x(p_i)$, $d(p_k, p_i) \le \varepsilon$ (Fig. 5).

Definition 4.4. For $p_i \in C_1$, we define $cone(p_i)$ to be the following cone of rightward directed rays emanating from p_i: a ray ρ belongs to $cone(p_i)$ if $-\pi/4 \le \Phi(\rho) \le \pi/4$ and $d(p_k, \rho) \le \varepsilon$, for $i \le k \le \lfloor n/2 \rfloor$.

Symmetrically, we define $cone(p_j)$, for $p_j \in C_2$, to be the following cone of leftward directed rays emanating from p_j: a ray ρ belongs to $cone(p_j)$ if $-\pi/4 \le \Phi(\rho) \le \pi/4$ and $d(p_k, \rho) \le \varepsilon$, for $\lfloor n/2 \rfloor + 1 \le k \le j$.

By Lemma 4.2, a rightward directed ray ρ originating at p_i belongs to $cone(p_i)$ if and only if $-\pi/4 \le \Phi(\rho) \le \pi/4$, and, for every $i \le k \le \lfloor n/2 \rfloor$,

(i) if $x(p_k) < x(p_i)$, then $d(p_k, p_i) \le \varepsilon$, and
(ii) if $x(p_k) \ge x(p_i)$, then ρ intersects the vertical segment $p_k^- p_k^+$.

Similarly, a leftward directed ray ρ originating at p_j belongs to $cone(p_j)$ if and only if $-\pi/4 \le \Phi(\rho) \le \pi/4$, and, for every $\lfloor n/2 \rfloor + 1 \le k \le j$,

(i) if $x(p_k) > x(p_j)$, then $d(p_k, p_j) \le \varepsilon$, and
(ii) if $x(p_k) \le x(p_j)$, then ρ intersects the vertical segment $p_k^- p_k^+$.

The following lemma lays the foundation for our algorithm.

Lemma 4.5. *For any* $(p_i, p_j) \in \bar{H}_{12}$, $(p_i, p_j) \in H_{12}$ *if and only if*

(1) p_i *is relevant for* p_j *and* p_j *is relevant for* p_i, *and*
(2) $p_j \in cone(p_i)$ *and* $p_i \in cone(p_j)$.

Proof. Let $(p_i, p_j) \in \bar{H}_{12}$, and assume that conditions (1) and (2) hold. To show that $(p_i, p_j) \in H_{12}$, we will show that $(p_i, p_j) \in G_\varepsilon$ by proving that for any k, $i \le k \le j$, $d(p_k, p_i p_j) \le \varepsilon$. We consider three cases:

1. $x(p_k) < x(p_i)$. If $p_k \in C_1$, it must be the case that $d(p_k, p_i) \le \varepsilon$, for otherwise $cone(p_i)$ would be empty. If $p_k \in C_2$, it follows from the fact that p_j is relevant for p_i that $d(p_k, p_i) \le \varepsilon$. In either case, $d(p_k, p_i p_j) \le \varepsilon$.
2. $x(p_k) > x(p_j)$. This case is symmetric to the earlier one.
3. $x(p_i) \le x(p_k) \le x(p_j)$. Assume that $p_k \in C_1$; the case where $p_k \in C_2$ is symmetric. Since $p_j \in cone(p_i)$, the ray $R(p_i, p_j)$ emanating from p_i and passing through p_j is in $cone(p_i)$. By the definition of $cone(p_i)$, it must be the case that $d(p_k, R(p_i, p_j)) \le \varepsilon$. Since $x(p_k) \ge x(p_i)$, it follows from Lemma 4.2 that $R(p_i, p_j)$ intersects the vertical segment $p_k^- p_k^+$. Moreover, since $x(p_k) \le x(p_j)$, we can conclude that the segment $p_i p_j$ intersects the vertical segment $p_k^- p_k^+$. Thus, $d(p_k, p_i p_j) \le \varepsilon$.

Now we establish the other half of the lemma. Assume that $(p_i, p_j) \in H_{12}$. We argue that (1) p_j is relevant for p_i, and (2) the ray $R(p_i, p_j)$ lies in $cone(p_i)$. By symmetrical arguments, we can also show that p_i is relevant for p_j and the ray $R(p_j, p_i)$ lies in $cone(p_j)$.

Consider any p_k for $\lfloor n/2 \rfloor + 1 \le k \le j$ such that $x(p_k) < x(p_i)$. Since $(p_i, p_j) \in G_\varepsilon$, $d(p_k, p_i p_j) \le \varepsilon$. Lemma 4.2 tells us that this can only happen if $d(p_k, p_i) \le \varepsilon$. (Here we are using the fact that $(p_i, p_j) \in \bar{H}_{12}$, and so $-\pi/4 \le \Phi(p_i p_j) \le \pi/4$.) We conclude that p_j is relevant for p_i.

Consider any p_k such that $i \le k \le \lfloor n/2 \rfloor$. Since $(p_i, p_j) \in G_\varepsilon$, it follows that $d(p_k, p_i p_j) \le \varepsilon$, and hence $d(p_k, R(p_i, p_j)) \le \varepsilon$. On the other hand, $(p_i, p_j) \in \bar{H}_{12}$, which imples that $R(p_i, p_j)$ is rightward directed and $-\pi/4 \le \Phi(R(p_i, p_j)) \le \pi/4$. We conclude that $R(p_i, p_j) \in cone(p_i)$. This completes the proof of the second half of the lemma. □

In view of Lemma 4.5, we compute a clique cover for H_{12} in four steps.

Step 1. We compute a clique cover of the edges in $\bar{H}_{12}$.

Step 2. For each bipartite clique (A_1, A_2) computed in Step 2, we compute a clique cover of the pairs $(p_i, p_j) \in A_1 \times A_2$ such that p_i is relevant for p_j and p_j is relevant for p_i.

Step 3. For each $p_i \in C_1$, we compute $cone(p_i)$; for each $p_j \in C_2$, we compute $cone(p_j)$.

Step 4. For each bipartite clique (B_1, B_2) computed in Step 3, we compute a clique cover of the pairs $(p_i, p_j) \in B_1 \times B_2$ such that $p_j \in cone(p_i)$ and $p_i \in cone(p_j)$. This is done by taking the segments dual to the cones and proceeding exactly as in Section 3.

The algorithms for Steps 1, 2, and 3 are described in Lemmas 4.7, 4.8, and 4.9, respectively. As described in Section 3, the procedure in Step 4 for a single bipartite clique (B_1, B_2) can be implemented in $O((|B_1|+|B_2|)^{4/3+\delta})$. We conclude that we can compute a clique cover of H_{12} of size $O(n^{4/3+\delta})$ in $O(n^{4/3+\delta})$ time. Putting everything together, we get an algorithm that computes a clique cover of G_ε whose size is $O(n^{4/3+\delta})$. The running time of the algorithm is also $O(n^{4/3+\delta})$. We then use Proposition 2.1 to compute a shortest path between p_1 and p_n in $G_\varepsilon(C)$, obtaining the main result of this section.

Theorem 4.6. *For any $\delta > 0$, we can solve the min-# problem under the L_1 metric for a (possibly) nonmonotone polygonal chain in $O(n^{4/3+\delta})$ time.*

In the rest of this section we describe the procedures for Steps 1, 2, and 3 in detail.

Lemma 4.7. *We can compute, in $O(n \log^2 n)$ time, a clique cover of $\bar{H}_{12}$ of size $O(n \log^2 n)$.*

Proof. For a point p in the plane, let

$$\mathrm{Wed}(p) = \{q \in \mathbb{R}^2 \mid |y(p) - y(q)| \le x(q) - x(p)\},$$

that is, $\mathrm{Wed}(p)$ is the wedge that is bounded by the two rightward directed rays emanating from p with orientations $+\pi/4$ and $-\pi/4$. By definition of $\bar{H}_{12}$, $(p_i, p_j) \in \bar{H}_{12}$ if and only if $p_j \in \mathrm{Wed}(p_i)$. We preprocess C_2 into a data structure to answer queries of the following form efficiently: Given a point p, report all points $q \in C_2$ that lie in $\mathrm{Wed}(p)$.

The data structure we construct is a range-tree [34] on C_2, which can report the points in C_2 that lie in a query wedge $\mathrm{Wed}(p)$ as a union of $O(\log^2 n)$ disjoint canonical subsets of C_2. The total size of all the canonical subsets in the range tree is $O(n \log n)$. With this data structure, we proceed as in Section 3 to construct a clique cover of $\bar{H}_{12}$. That is, we query the data structure with all points $p \in C_1$. For each canonical subset B_i in the range tree, let $A_i \subseteq C_1$ be the set of query points whose output contains B_i. If $A_i \neq \emptyset$, we include (A_i, B_i) as a bipartite clique of the clique cover. The size of the clique cover is $O(n \log^2 n)$, and the running time is also $O(n \log^2 n)$. □

Lemma 4.8. *Let $A \subseteq C_1$ and $B \subseteq C_2$, and let $|A|+|B| = m$. Let $\mathrm{Rel}(A, B) \subseteq A \times B$ denote the collection of ordered pairs (p_i, p_j) such that p_i is relevant for p_j and p_j is relevant for p_i. We can compute a clique cover of $\mathrm{Rel}(A, B)$ of size $O(m \log^2 m)$ in $O(m \log^6 m)$ time.*

Proof. We proceed in two stages. In the first stage we compute a clique cover $\mathcal{G}'$ for the ordered pairs $(p_i, p_j) \in A \times B$ such that p_i is relevant for p_j. We assume that A is given as a sequence ordered according to C_1; that is, $p_i \in A$ occurs before $p_k \in A$ if $i < k$. For $p_j \in B$, let $\alpha(p_j)$ be the last point in the sequence A such that $x(\alpha(p_j)) > x(p_j)$ and $d(\alpha(p_j), p_j) > \varepsilon$. Observe that the points in the sequence A *after* $\alpha(p_j)$ are the ones that are relevent for p_j. Thus the set of points in A that are relevant for a given $p_j \in B$ forms a suffix of the sequence A. Exploiting this observation, we can compute,

in a straightforward fashion, a collection of canonical subsets of A, whose total size is $O(m \log m)$, so that, for any $p_j \in B$, the set of points in A that are relevant for p_j can be returned as a disjoint union of $O(\log m)$ canonical sets. With this data structure, we proceed as in the proof of Lemma 4.7 to construct a clique cover $\mathcal{G}'$ whose size is $O(m \log m)$. The time for computing the clique cover is governed by the time needed to compute $\alpha(p_j)$ for each $p_j \in B$; combining binary search with a data structure based on range trees [34], this can be done in a total of $O(m \log^5 m)$ time.

In the second stage we take each bipartite clique (A', B') of $\mathcal{G}_1$, and compute a clique cover of the set of ordered pairs $(p_i, p_j) \in A' \times B'$ such that p_j is relevant for p_i. We do this in a manner completely similar and symmetric to the first stage. If $|A'| + |B'| = m'$, the second stage applied to (A', B') produces a clique cover of size $O(m' \log m')$.

We return the union of all the clique covers computed after the second stage as the clique cover of $\mathrm{Rel}(A, B)$. The size of the clique cover is $O(m \log^2 m)$, and the overall running time is $O(m \log^6 m)$. □

Lemma 4.9. *We can compute cone(p_i), for $1 \le i \le n$, in $O(n \log^3 n)$ time.*

Proof. We only describe the computation of *cone*(p_i) for $p_i \in C_1$. Let left(p_i) (resp. right(p_i)) be the set of points $p_k, i \le k \le \lfloor n/2 \rfloor$, such that $x(p_k) < x(p_i)$ (resp. $x(p_k) \ge x(p_i)$). To compute *cone*(p_i), we first test if $d(p_i, p) \le \varepsilon$, for every $p \in$ left(p_i). If, for any $p \in$ left(p_i), $d(p_i, p) > \varepsilon$, we stop and declare *cone*(p_i) to be empty. Otherwise, we compute the cone of rightward directed rays emanating from p_i consisting of all rays ρ such that ρ intersects the vertical segment $p^- p^+$, for every $p \in$ right(p_i). We clip this cone so that all rays of this cone have orientation between $-\pi/4$ and $\pi/4$, and return the clipped cone as *cone*(p_i).

In order to do all this efficiently, we preprocess C_1 into a data structure that returns left(p_i) (or right(p_i)), for any $p_i \in C_1$ as a union of $O(\log^2 n)$ canonical subsets. This data structure is simply the two-dimensional range tree in which the points are ordered in one dimension according to their x-coordinates, and in the other dimension according to their occurrence in the chain C_1 (that is, $p_1, \ldots, p_{\lfloor n/2 \rfloor}$). With each canonical subset S of the data structure, we store the region $reg(S) = \{q \mid d(q, p) \le \varepsilon, \text{ for all } p \in S\}$. It is easy to see that $reg(S)$ is a rectangle. We also store the upper boundary $U(S)$ of the convex hull of the points $\{p^- | p \in S\}$ and the lower boundary $L(S)$ of the points $\{p^+ | p \in S\}$. The data structure can be built in $O(n \log^3 n)$ time.

With these data structures set up, we can compute *cone*(p_i), for any $p_i \in C_1$, as follows. We query the data structure and find $O(\log^2 n)$ canonical subsets $\{S_1, \ldots, S_l\}$ whose union is left(p_i). For each S_k, we check whether $p_i \in reg(S_k)$; if, for any S_k, $p_i \notin S_k$, we stop and declare *cone*(p_i) to be empty. Otherwise, we query the data structure to find $O(\log^2 n)$ canonical $\{S_1, \ldots, S_l\}$ subsets whose union is right(p_i). For each S_k, we compute the cone of rightward directed rays emanating from p_i that is bounded on the top by the ray from p_i tangent to $L(S_k)$, and on the bottom by the ray from p_i tangent to $U(S_k)$. We compute the intersection of the cones computed for each S_k. Finally, we clip this cone to one whose rays have orientation between $-\pi/4$ and $\pi/4$.

Using this procedure, we can compute *cone*(p_i) for any $p_i \in C_1$ in $O(\log^3 n)$ time. Hence, the overall algorithm for computing all the cones runs in $O(n \log^3 n)$ time. □

5. The Min-ε Algorithm under the L_1 Metric

In the min-ε problem we are given a polygonal chain $C = \langle p_1 \cdots p_n \rangle$ and an integer $k \leq n$, and we want to find an approximation of C that minimizes the error over all approximations that use at most k vertices. In this section we present a randomized algorithm to solve the min-ε problem for a polygonal chain C under the L_1 metric. As mentioned in Section 2, the min-ε problem reduces to finding the smallest value of ε, for which there is a path in G_ε between p_1 and p_n consisting of at most k vertices; we let ε^* denote this smallest value. Recall that $\varepsilon^* = \Delta(p_i p_j)$, the error of some segment p_i, p_j. Given an ε, we can use the min-# algorithm as a *decision procedure* to determine whether $\varepsilon^* \leq \varepsilon$ or $\varepsilon^* > \varepsilon$. Hence, we can use the decision procedure to binary search the errors corresponding to each of the $\Theta(n^2)$ segments $p_i p_j$. However, we cannot explicitly enumerate these errors if we are aiming for a subquadratic algorithm. Instead, we use a variant of the *random halving* technique [30] to do the search. Our algorithm can be made deterministic using the expander based approach by Katz and Sharir [27]. We first describe some primitives that our algorithm uses.

Lemma 5.1. *For any given $\varepsilon \geq 0$, we can preprocess the polygonal chain C into a data structure in $O(n^{4/3+\delta})$ time, for any $\delta > 0$, so that we can select with uniform probability a random segment $p_i p_j$ whose error is at most ε. The selection procedure takes $O(\log n)$ time.*

Proof. For simplicity, we assume that we have access to a random number generator that can generate a random number in the range $(0, 1]$ with uniform probability. We first compute a clique cover $\mathcal{G} = \{G_1 = (V_1, E_1), \ldots, G_l = (V_l, E_l)\}$ of the graph G_ε in $O(n^{4/3+\delta})$ time. Let A_i and B_i denote the vertex classes of V_i. Let

$$w = \sum_{1 \leq i \leq l} |A_i| * |B_i|$$

denote the total number of segments with error at most ε. We first describe how we can select a bipartite clique G_i with probability $w_i = |A_i||B_i|/w$. We divide the interval $(0, 1]$ into intervals $I_1, \ldots, I_l$, where

$$I_j = \left(\sum_{1 \leq k \leq j-1} w_k, \sum_{1 \leq k \leq j} w_k \right].$$

To generate a random bipartite clique, we first generate a random number $r \in (0, 1]$. We do a binary search on the intervals $I_1, \ldots, I_l$ to locate the interval I_j containing r, and return G_j.

For each A_i (resp. B_i), we build a similar structure that will allow us to generate an element $a \in A_i$ (resp. $b \in B_i$) with probability $1/|A_i|$ (resp. $1/|B_i|$). To pick a random segment $p_i p_j$ with error at most ε, we first pick a random bipartite clique as above; if G_k is picked, we pick random elements $a \in A_k$ and $b \in B_k$, and return (a, b).

The preprocessing time of this scheme is $O(n^{4/3+\delta})$, and the time for generating a random segment is $O(\log n)$. If $\Delta(p_i p_j) \leq \varepsilon$, and $(p_i, p_j) \in A_k \times B_k$, the probability of picking (p_i, p_j) is

$$w_k * 1/|A_k| * 1/|B_k| = 1/w. \qquad \square$$

Using clique covers, the following lemma is easily established.

Lemma 5.2. *For any given $\varepsilon \geq 0$, we can* count, *in $O(n^{4/3+\delta})$ time, for any $\delta > 0$, the number of segments $p_i p_j$ whose error is at most ε. As a corollary, we can count in $O(n^{4/3+\delta})$ time the number of segments $p_i p_j$ whose error lies in a given range $(\varepsilon_1, \varepsilon_2]$.*

Proof. We compute a clique cover $\mathcal{G} = \{G_1 = (V_1, E_1), \ldots, G_l = (V_l, E_l)\}$ of the graph G_ε. Let A_i and B_i denote the vertex classes of V_i. Then the total number of segments with error at most ε is

$$\sum_{1 \leq i \leq l} |A_i| * |B_i|.$$

The number of segments whose error lies in the range $(\varepsilon_1, \varepsilon_2]$ is the number of segments whose error is at most ε_2 minus the number of segments whose error is at most ε_1. □

We will also need the following lemma.

Lemma 5.3. *We can preprocess the polygonal chain C in $O(n \log^2 n)$ time so that given a query segment $p_i p_j$, its error can be computed in $O(\log^3 n)$ time.*

Proof. We only describe the data structure for query segments $p_i p_j$ that belong to

$$\bar{H} = \{p_i p_j | x(p_i) \leq x(p_j), -\pi/4 \leq \Phi(p_i p_j) \leq \pi/4\}.$$

Let

$$\begin{aligned} \text{left}(p_i, p_j) &= \{p_k \mid i \leq k \leq j,\ x(p_k) < x(p_i)\}, \\ \text{right}(p_i, p_j) &= \{p_k \mid i \leq k \leq j,\ x(p_k) > x(p_j)\}, \quad \text{and} \\ \text{between}(p_i, p_j) &= \{p_k \mid i \leq k \leq j,\ x(p_i) \leq x(p_k) \leq x(p_j)\}. \end{aligned}$$

We preprocess the points $\{p_1, \ldots, p_n\}$ in $O(n \log n)$ time into a data structure so that given any (p_i, p_j), left(p_i, p_j), right(p_i, p_j), and between(p_i, p_j) can be returned as a union of $O(\log^2 n)$ canonical subsets. This data structure is simply the two-dimensional range tree in which the points are ordered in one dimension according to their x-coordinates, and in the other dimension according to their occurrence in the chain C (that is, $p_1, \ldots, p_n$). For each canonical set $S \subseteq C$ of the data structure, we store conv(S), the convex hull of S. We also store extr(S), the four "L_1-extremal" points of S; for any point p in the plane, the point in S that maximizes the L_1 distance to p is one of these four points.

This completes the description of our data structure for query segments in $\bar{H}$. Recall that $\Delta(p_i p_j) = \max_{i \leq k \leq j} d(p_k, p_i p_j)$. For $p_i p_j \in \bar{H}$, $d(p_k, p_i p_j)$ equals $d(p_k, p_i)$ (resp. $d(p_k, p_j)$) for $p_k \in \text{left}(p_i, p_j)$ (resp. for $p_k \in \text{right}(p_i, p_j)$), and equals the vertical distance between p_k and the line ℓ through p_i and p_j for $p_k \in \text{between}(p_i, p_j)$ because $-\pi/4 \leq \Phi(p_i p_j) \leq \pi/4$. It follows that $\Delta(p_i p_j)$ is determined by either the point q_1 in left(p_i, p_j) that maximizes the distance to p_i, or the point q_2 in right(p_i, p_j)

that maximizes the distance to p_j, or the point in q_3 between(p_i, p_j) that maximizes the vertical distance to the line ℓ.

Our query procedure for computing $\Delta(p_i p_j)$ computes q_1, q_2, and q_3 as follows. We first query the above data structure to find the $O(\log^2 n)$ canonical sets whose union is left(p_i, p_j). For each canonical set S, we find the point in S maximizing the distance to p_i by looking at the four L_1-extremal points of S. Thus, we can determine q_1 in $O(\log^2 n)$ time. By a symmetric scheme, we can also determine q_2 in $O(\log^2 n)$ time. To determine q_3, we query the above data structure to find $O(\log^2 n)$ canonical sets whose union is between(p_i, p_j). For each canonical set S, we find the point in S maximizing the vertical distance to ℓ by doing a binary search over the convex hull conv(S) of S. The binary search finds the two lines ℓ' and ℓ'' that are parallel to ℓ and tangent to conv(S). If $p' \in S$ (resp. $p'' \in S$) is the point through which ℓ' (resp. ℓ'') passes, then either p' or p'' maximizes the vertical distance from S to ℓ. Thus, we can determine q_3 in $O(\log^3 n)$ time. The time for the overall query procedure is also $O(\log^3 n)$. □

The Algorithm. The min-ε algorithm maintains a working interval $I = (\varepsilon_1, \varepsilon_2]$ containing ε^*. We refer to the segments (p_i, p_j) whose errors lie in the working interval as the *candidate segments*, and the corresponding errors as the *candidate values*. Our algorithm operates in two phases. In the first phase we repeatedly shrink the working interval containing ε^* until it contains at most $t = \lfloor n^{2/3} \rfloor$ candidate values. In the second phase we expicitly enumerate all the segments whose errors lie in the working interval, and binary search the errors to find ε^*. We can afford to do the explicit enumeration because the working interval does not contain too many candidate values. We now describe the phases in detail.

The First Phase. The first phase works in stages. Suppose that at the beginning of the ith stage, we have a working interval $I^{i-1} = (\varepsilon_1, \varepsilon_2]$ that contains ε^*. (Before beginning the first stage, we check whether $\varepsilon^* = 0$ using the decision procedure. If $\varepsilon^* > 0$, we set $I^0 = (0, \infty]$.)

1. Let N^{i-1} denote the set of the candidate values contained in I^{i-1}. We check whether $|N^{i-1}| \leq t$ using the algorithm of Lemma 5.2. If $|N^{i-1}| \leq t$, the first phase ends and we proceed to the second phase. Otherwise, we shrink the working interval, as described below, to an interval I^i such that $|N^i| \leq |N^{i-1}|/3$.
2. We run the preprocessing algorithm of Lemma 5.1 that will allow us to generate a random segment $p_i p_j$ whose error lies in the range $(0, \varepsilon_2]$.
3. We generate, in $O(\log n)$ time, a random segment $p_i p_j$ whose error lies in the range $(0, \varepsilon_2]$. We compute its error $\varepsilon' = \Delta(p_i p_j)$ in $O(\log^3 n)$ time using Lemma 5.3. If ε' does not lie in the working interval I_{i-1} (i.e., $\varepsilon' < \varepsilon_1$), we repeat this step.
4. Otherwise ($\varepsilon' \in I_{i-1}$), we check if ε' lies in the middle third of the values in N^{i-1}. We do this in $O(n^{4/3+\delta})$ time by using the algorithm of Lemma 5.2 to count the number of candidate values in the intervals $(\varepsilon_1, \varepsilon']$ and $(\varepsilon', \varepsilon_2]$. If ε' does not lie in the middle third, we go back to Step 3.
5. We use the min-# algorithm to decide if $\varepsilon^* > \varepsilon'$, or $\varepsilon^* \leq \varepsilon'$. If $\varepsilon^* > \varepsilon'$, we let the working interval for the next stage be $I^i = (\varepsilon', \varepsilon_2]$; otherwise we let $I^i = (\varepsilon_1, \varepsilon']$.

Clearly, a random candidate value in the interval I_{i-1} lies in the middle third of the values in N_{i-1} with probability at least $1/3$. Since N_{i-1} contains at least t values, a random segment $p_i p_j$ generated in Step 3 lies in I_{i-1} with probability at least $t/n^2 \approx 1/n^{4/3}$. Using these observations, it is easy to see that with high probability (probability at least $1 - 1/n^c$, for some constant c), there are $O(n^{4/3} \log n)$ iterations of Step 3 and $O(\log n)$ iterations of Step 4 in the ith stage. We can conclude that the time taken to execute the ith stage is $O(n^{4/3+\delta})$, with high probability. Since, during each stage, the number of candidate values in the working interval decreases by a constant fraction, there are only a logarithmic number of stages. Therefore, the overall running time of the first phase is $O(n^{4/3+\delta})$, with high probability.

The Second Phase. Assume that we enter the second phase with the working interval $I = (\varepsilon_1, \varepsilon_2]$. We first compute a superset of such segments and then discard those whose values do not lie in I. We compute the candidate segments and values that lie in I as follows. We compute the clique covers of G_{ε_1} and G_{ε_2}, and use them to find the in-degree and out-degree of each vertex in G_{ε_1} and G_{ε_2}; this takes $O(n^{4/3+\delta})$ time. We then find the set S_1 (resp. S_2) of all vertices p_i (resp. p_j) of C for which there is some p_j (resp. p_i), with $i < j$, such that the error of segment $p_i p_j$ lies in I. A vertex p_i belongs to S_1 (resp. S_2) if its out-degree (resp. in-degree) in G_{ε_1} is strictly smaller than its out-degree (resp. in-degree) in G_{ε_2}. Hence, we can find S_1 (resp. S_2) by comparing the out-degrees (resp. in-degrees) of each vertex in G_{ε_1} and G_{ε_2}. Clearly, S_1 and S_2 contain at most t vertices. Using Lemma 5.3, we compute the error of each $(p_i, p_j) \in S_1 \times S_2$, and discard the segments whose errors do not lie in I. We are left with the segments whose errors lie in I. Finally, we perform a binary search over the $O(n^{2/3})$ candidate values in I, using the min-# algorithm as a decision procedure, to compute ε^*. The overall running time of the second phase is also bounded by $O(n^{4/3+\delta})$. Hence, we can conclude:

Theorem 5.4. *For any $\delta > 0$, we can solve the min-ε problem under the L_1 metric using a randomized algorithm with expected running time $O(n^{4/3+\delta})$.*

6. Conclusions

We have presented efficient algorithms that exploit the structure of the graph of short-cuts G_ε. This brings us to an extremely interesting question: What happens when other error criteria are used? (See [25].) An interesting case is when the Euclidean distance is used to define the error. We have not been able to extend the techniques used in this paper to compute a compact representation of the graph G_ε in this case.

Another interesting problem is to find near-linear-time algorithms for the problems solved in this paper.

References

1. P. K. Agarwal, N. Alon, B. Aronov, and S. Suri. Can visibility graphs be represented compactly? *Discrete Comput. Geom.*, 12:347–365, 1994.
2. P. K. Agarwal and M. Sharir. Applications of a new space-partitioning technique. *Discrete Comput. Geom.*, 9:11–38, 1993.

3. A. Aggarwal, H. Booth, J. O'Rourke, S. Suri, and C. K. Yap. Finding minimal convex nested polygons. *Inform. Comput.*, 83(1):98–110, Oct. 1989.
4. N. Alon and J. Spencer. *The Probabilistic Method*. Wiley, New York, 1993.
5. T. Asano and N. Katoh. Number theory helps line detection in digital images. In *Proc. 4th Annual International Symposium on Algorithms and Computing*, volume 762 of Lecture Notes in Computer Science, pages 313–322. Springer-Verlag, Berlin, 1993.
6. R. E. Bellman and R. S. Roth. *Methods in Approximation*: *Techniques for Mathematical Modelling*. Reidel, Boston, MA, 1986.
7. B. Buttenfield. Treatment of the cartographic line. *Cartographica*, 22:1–26, 1985.
8. W. S. Chan and F. Chin. Approximation of polygonal curves with minimum number of line segments. In *Proc. 3rd Annual International Symposium on Algorithms and Computing*, volume 650 of Lecture Notes in Computer Science, pages 378–387. Springer-Verlag, Berlin, 1992.
9. S. D. Conte and C. de Boor. *Elementary Numerical Analysis*: *An Algorithmic Approach*, 3rd edn. McGraw-Hill, New York, 1980.
10. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990.
11. R. G. Cromley. A vertex substitution approach to numerical line simplification. In *Proc. 3rd International Symposium Spatial Data Handling*, pages 57–64, 1988.
12. P. J. Davis. *Interpolation and Approximation*. Blaisdell, New York, 1963.
13. M. de Berg, M. van Kreveld, and S. Schirra. A new approach to subdivision simplification. In *Proc. 12th International Symposium on Computer-Assisted Cartography*, pages 79–88, 1995.
14. P. Dierckx. *Curve and Surface Fitting with Splines*. Clarendon Press, New York, 1993.
15. D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Canad. Cartog.*, 10(2):112–122, Dec. 1973.
16. D. Eu and G. T. Toussaint. On approximating polygonal curves in two and three dimensions. *CVGIP*: *Graph. Models Image Process.*, 56(3):231–246, May 1994.
17. T. Feder and R. Motwani. Clique partitions, graph compression, and speeding up algoithms. In *Proc. 27th Annual ACM Symposium on Theory of Computing*, pages 123–133, 1991.
18. M. T. Goodrich. Efficient piecewise-linear function approximation using the uniform metric. *Discrete Comput. Geom.*, 14:445–462, 1995.
19. L. J. Guibas, J. E. Hershberger, J. S. B. Mitchell, and J. S. Snoeyink. Approximating polygons and subdivisions with minimum link paths. *Internat. J. Comput. Geom. Appl.*, 3(4):383–415, Dec. 1993.
20. S. L. Hakimi and E. F. Schmeichel. Fitting polygonal functions to a set of points in the plane. *CVGIP*: *Graph. Models Image Process.*, 53(2):132–136, 1991.
21. J. Hershberger and J. Snoeyink. Speeding up the Douglas–Peucker line simplification algorithm. In *Proc. 5th International Symposium on Spatial Data Handling*, pages 134–143, 1992.
22. J. Hershberger and J. Snoeyink. Computing minimum length paths of a given homotopy class. *Comput. Geom. Theory Appl.*, 4:63–98, 1994.
23. J. Hershberger and J. Snoeyink. Cartographic line simplification and polygon CSG formulae in $O(n \log^* n)$ time. In *Proc. 5th International Workshop on Algorithms and Data Structures*, volume 1272 of Lecture Notes in Computer Science, pages 93–103. Springer-Verlag, Berlin, 1997.
24. J. D. Hobby. Polygonal approximations that minimize the number of inflections. In *Proc. 4th ACM–SIAM Symposium on Discrete Algorithms*, pages 93–102, 1993.
25. H. Imai and M. Iri. Polygonal approximations of a curve-formulations and algorithms. In G. T. Toussaint, editor, *Computational Morphology*, pages 71–86. North-Holland, Amsterdam, 1988.
26. S. Kahan and J. Snoeyink. On the bit complexity of minimum link paths: superquadratic algorithms for problems solvable in linear time. In *Proc. 12th Annual ACM Symposium on Computational Geometry*, pages 151–158, 1996.
27. M. J. Katz and M. Sharir. An expander-based approach to geometric optimization. *SIAM J. Comput.*, 26:1384–1408, 1997.
28. Y. Kurozumi and W. A. Davis. Polygonal approximation by the minimax method. *Comput. Graph. Image Process.*, 19:248–264, 1982.
29. Z. Li and S. Openshaw. Algorithms for automated line generalization based on a natural principle of objective generalization. *Internat. J. Geogr. Inform. Systems*, 6:373–389, 1992.
30. J. Matoušek. Randomized optimal algorithm for slope selection. *Inform. Process. Lett.*, 39:183–187, 1991.

31. R. B. McMaster. Automated line generation. *Cartographica*, 24(2):74–111, 1987.
32. A. Melkman. On-line construction of the convex hull of a simple polyline. *Inform. Process. Lett.*, 25:11–12, 1987.
33. A. Melkman and J. O'Rourke. On polygonal chain approximation. In G. T. Toussaint, editor, *Computational Morphology*, pages 87–95. North-Holland, Amsterdam, 1988.
34. F. P. Preparata and M. I. Shamos. *Computational Geometry*: *An Introduction*. Springer-Verlag, New York, 1985.
35. S. Suri. A linear time algorithm for minimum link paths inside a simple polygon. *Comput. Vision Graph. Image Process.*, 35:99–110, 1986.
36. G. T. Toussaint. On the complexity of approximating polygonal curves in the plane. In *Proc. IASTED, International Symposium on Robotics and Automation*, 1985.

Received September 17, 1998, *and in revised form July* 8, 1999.