

Light Orthogonal Networks with Constant Geometric Dilation*

Adrian Dumitrescu[†]

Csaba D. Tóth[§]

July 15, 2008

Abstract

An orthogonal spanner network for a given set of n points in the plane is a plane straight line graph with axis-aligned edges that connects all input points. We show that for any set of n points in the plane, there is an orthogonal spanner network that (i) is *short* having a total edge length at most a constant times the length of a Euclidean minimum spanning tree for the point set; (ii) is *small* having $O(n)$ vertices and edges; and (iii) has *constant geometric dilation*, which means that for any two points u and v in the network, the shortest path in the network between u and v is at most a constant times longer than the Euclidean distance between u and v . Such a network can be constructed in $O(n \log n)$ time.

1 Introduction

A typical problem in the theory of metric embeddings asks for a mapping from one metric space to another that distorts the distances between point pairs as little as possible. In this paper, we address the following problem about geometric dilation: Given a finite set S of points in the plane, find a small plane graph G containing S so that the distortion between the L_2 distance and the Euclidean shortest path distance in G between any two points (on edges or at vertices) of G is bounded by an absolute constant.

A restricted variant of this problem, where the distortion is measured only for pairs of points in S , called *stretch factor* or *vertex dilation*, received increased attention in the late 80s and early 90s [9, 11, 24, 26]; see [17] for a survey. From the previous results, perhaps the most similar to ours is that of Bose *et al.* [8], which states that for any set S of n points in the plane, one can construct in $O(n \log n)$ time a plane graph H with four properties: (i) the vertex set of H is S , (ii) H has maximum degree $O(1)$, (iii) the total length of the edges of H is $O(W)$, where $W = W(S)$ denotes the length of a Euclidean minimum spanning tree for S , and (iv) for any two vertices $u, v \in S$, the shortest path along H is at most $O(1)$ times longer than the distance between u and v . The last property is also referred to as constant *vertex dilation*. Note that the graph H is sparse and the bound on the total length is best possible, since H has to be connected. Intuitively, the graph H corresponds to a road network that ensures that the *detour* between any two of n given cities is bounded by a constant (see a precise definition below). However, there may be pairs of points along the roads (halfway between cities) with arbitrarily large detour.

Good transportation networks should have small detour values for any pair of points in the network (graph): since most new sites (facilities) that are being deployed naturally lie on the existing roads, their detour to any other point in the network is therefore small from the start. Other applications include directional wireless networks where interference is undesirable (see the connection between constant geometric dilation and so-called narrow channels, discussed in the next section).

*A preliminary version of this paper appeared in the *Proceedings of the 24th Symposium on Theoretical Aspects of Computer Science (Aachen, 2007)*, vol. 4393 of LNCS, Springer, pp. 175-187.

[†]Department of Computer Science, University of Wisconsin–Milwaukee, USA. E-mail: ad@cs.uwm.edu

[‡]Supported in part by NSF CAREER grant CCF-0444188.

[§]Department of Mathematics, University of Calgary, Alberta, Canada. E-mail: cdtotoh@ucalgary.ca

Let us recall the formal definition of geometric dilation (see [13, 15]). Let G be a plane graph whose edges are curves. If there is no danger of confusion, G also denotes the set of points in the plane covered by the edges and vertices of the plane graph G . The *detour* between two points $u, v \in G$ (on edges or vertices of G) is the ratio between the length $d_G(u, v)$ of a Euclidean shortest path connecting u and v in G and their Euclidean distance $|uv|$. The supremum value of detours over all pairs of points, denoted $\delta(G)$, is called the *geometric dilation* of G :

$$\delta(G) := \sup_{u, v \in G} \frac{d_G(u, v)}{|uv|}.$$

In contrast, the *vertex dilation* (also known as *stretch factor*) is $\max_{u, v \in V(G)} d_G(u, v)/|uv|$, where $V(G)$ is the vertex set of G . For instance, if G consists of the four vertices and edges of a rectangle of aspect ratio t (that is, a longer edge of the rectangle is $t \geq 1$ times longer than a shorter edge), then the geometric dilation of G is $t + 1$, while its vertex dilation is only $(t + 1)/\sqrt{t^2 + 1}$. For a set S of n points in the plane, a *spanner network* G is a connected plane straight line graph (for short PSLG) whose vertex set contains S .

In the current paper, we further extend the results in [8] and construct a graph G spanning the points in S such that G is not only a *plane* graph with $O(1)$ maximum degree, $O(W)$ weight, and $O(1)$ stretch factor, but G also has constant *geometric dilation*, that is, the detour between *any* two points of the graph (not just between vertices) is bounded from above by a constant. In addition, G is an *orthogonal* network, having axis-parallel edges, hence its maximum degree is at most 4. Our construction uses $O(n)$ Steiner points, which is best possible in general, since there are n -element point sets in the plane for which every orthogonal spanner network requires $\Omega(n)$ Steiner points.

Theorem 1 *For every set S of n points in the plane, there is an orthogonal spanner network G such that (i) its geometric dilation is $O(1)$; (ii) it has $O(n)$ vertices; and (iii) its length is $O(W)$. Such a network can be computed in $O(n \log n)$ time.*

The choice of parameters in our construction allows trade-offs among the geometric dilation, the number of vertices, and the length of the network. In this work, we focused on finding a simple and efficient way to construct a network with properties (i)–(iii). We made no attempt to optimize the constant coefficients hidden in the asymptotic notation of our bounds. In the analyses of our algorithms, we preferred simplicity to sharpness, and some of the constants are currently too large for practical applications.

Related previous results

Geometric spanners and vertex dilation. Planar straight line graphs with constant vertex dilation were thoroughly studied in the context of geometric spanners, motivated by VLSI design problems [17, 27]. Chew [10] proved that the vertex dilation of the rectilinear Delaunay triangulation of n points in the plane is at most $\sqrt{10}$. He also conjectured that the vertex dilation of the Euclidean Delaunay triangulation is at most $\pi/2 \approx 1.57$, which would be best possible; Dobkin *et al.* [12] gave an upper bound of about 5.08, which was later improved by Keil and Gutwin [24] to $4\pi/(3\sqrt{3}) \approx 2.42$. Das and Joseph [11] found a large class of geometric graphs with constant vertex dilation, characterized by a certain *diamond property*.

A lot of work has been done on constructing “good” spanners: sparse and light graphs with constant vertex dilation. For a set S of n points in the plane, a greedy algorithm [2, 26] on the Delaunay triangulation computes a plane spanner graph G with vertex set S that has $O(1)$ vertex dilation and $O(W)$ length. Here the vertex dilation can be made smaller than $2.42 \cdot t$, for any $t > 1$, while the length of the graph increases to at most $\frac{t+1}{t-1}W$. Bose *et al.* [8] were able to combine planarity with constant maximum degree while guaranteeing constant vertex dilation. However, none of these results provides any upper bound for the *geometric* dilation of the resulting networks.

If the network is not required to be plane, Narasimhan and Smid [28] proved that a greedy algorithm on the *complete* graph can produce a spanner network whose vertex dilation is t , for any $t > 1$, while having $O((t-1)^{-4}W)$ length and vertex degree bounded by $O((t-1)^{-2} \log(t-1)^{-1})$. Recently, Aronov *et al.* [3] gave a tight worst-case bound on the vertex dilation in terms of the number of edges of the spanning network of n points. For many other related results, see the recent book of Narasimhan and Smid [28] and a survey paper of Gudmundsson and Knauer [20] on geometric spanners.

Geometric dilation of planar point sets. The problem of embedding a given planar point set in a network of small geometric dilation, as well as the problem of computing or estimating the dilation of planar networks have only recently received attention. First attempts were made in designing efficient algorithms for computing the geometric dilation of a polygonal curve [1, 16]. Dumitrescu *et al.* [13] showed that some point sets require geometric dilation strictly greater than $\pi/2 \approx 1.5707$: at least $(1 + 10^{-11})\pi/2$, to be precise. Ebberts-Baumann *et al.* [15] proved that every finite point set can be embedded in a plane graph (with curved edges) of geometric dilation at most 1.678: This network, however, may use an exponential number of Steiner points, and may be much heavier than W . Here we show how to construct a plane spanner network for a point set which *simultaneously* has few Steiner vertices, small weight, and constant geometric dilation.

Related problems. A somewhat related problem is the *Manhattan network* problem. A plane graph whose vertex dilation is 1 under the L_1 metric is called a Manhattan network [5, 21]. For our purpose such networks might be too expensive. Take, for instance, n equidistant points on the boundary of an axis-aligned unit square. The minimum Manhattan network of such a graph has weight $\Omega(nW)$ and it contains $\Omega(n^2)$ Steiner vertices; while the unit square itself has weight $O(W)$, n vertices, and geometric dilation 2.

2 Reduction to axis-aligned polygons

Notation on plane straight line graphs and polygons. A *plane straight line graph* (PSLG) is a finite graph together with an embedding into the plane \mathbb{R}^2 , where the vertices are mapped to distinct points and the edges are mapped to straight line segments, any two of which are either disjoint or meet only at a common endpoint. The complement $\mathbb{R}^2 \setminus G$ of a PSLG G may have several components. Since G is finite, exactly one component of $\mathbb{R}^2 \setminus G$ is unbounded, and all others are bounded. The portion of G lying on the boundary of a bounded component of $\mathbb{R}^2 \setminus G$ is called a face. The *interior* of a face f is denoted by $\text{int}(f)$.

A *simple polygon* is a connected 2-regular PSLG. A *weakly simple polygon* is a PSLG that can be covered with a closed polygonal chain $(p_1, p_2, \dots, p_k, p_{k+1} = p_1)$ with possible repetitions such that the sides $p_i p_{i+1}$ are pairwise noncrossing segments and each point p_i can be perturbed by at most an arbitrarily small $\epsilon > 0$ to a position p_i' so that the closed polygonal chain $P' = (p_1', p_2', \dots, p_k', p_{k+1}' = p_1')$ is a simple polygon (Fig. 1(a-b)). In particular, every bounded face of a PSLG G is a *weakly simple polygon with simply connected interior*, which is for short called *polygon* in this paper. The union of a polygon P and its interior $\text{int}(P)$ is the *polygonal domain* $\text{dom}(P) \subset \mathbb{R}^2$. A *subdivision* of a polygon P is a PSLG G with $P \subset G \subset \text{dom}(P)$.

The *length* of a PSLG G , denoted $|G|$, is the total length of the edges of G . We use the terms *length* and *weight* interchangeably. For two PSLGs, G_1 and G_2 , with no two crossing edges, we use $G_1 \cup G_2$ (resp., $G_1 \setminus G_2$) to denote the PSLG which is the union (resp., set difference) of the two drawings G_1 and G_2 (edges may be subdivided in this process). The cardinality of a set X is denoted by $\#X$.

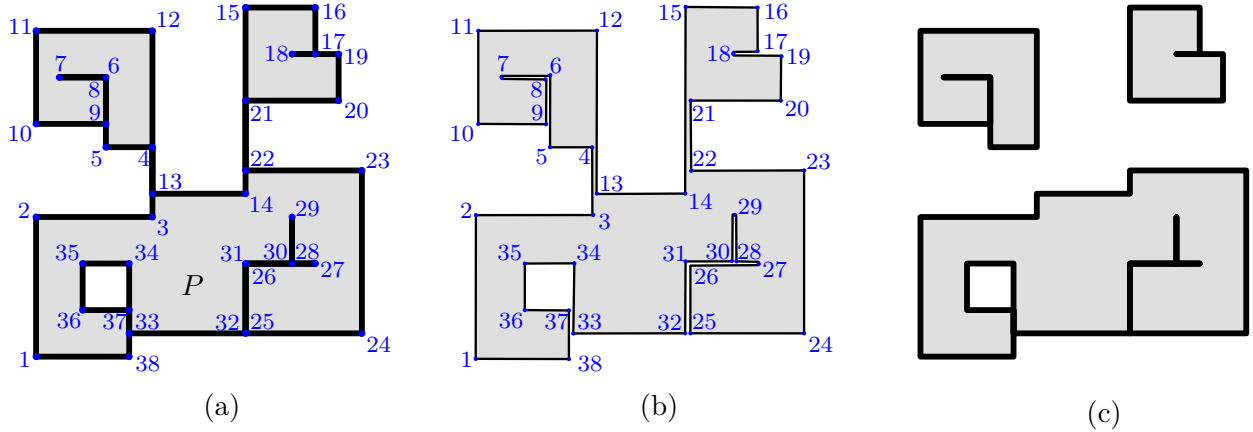


Figure 1: (a) A weakly simple polygon P covered with a closed polygonal chain $(p_1, p_1, \dots, p_{38}, p_1)$ with 38 vertices. (b) Polygon P is perturbed into a simple polygon. (c) The bounded faces of P are weakly simple polygons with simply connected interiors, for short, *polygons* in this paper.

2.1 Our algorithm in a nutshell

We construct an orthogonal spanner network for a given set S of n points in the plane (Fig. 2(a)). First, we reduce the problem to a polygon subdivision problem. We construct a constant factor approximation $T_A(S)$ of a rectilinear Steiner minimum tree (RSMT) of S . The tree $T_A(S)$ retains a key property of RSMT, which we call *spaciousness*; it is similar, in some sense, to the diamond property from [11]. Intuitively, an empty rectangle of large aspect ratio is called a “narrow channel” if its two longer sides are contained in two parallel edges of the graph. A PSLG G is “spacious” if it has no narrow channels. A narrow channel is undesirable because the detour between the midpoints of the longer sides of the channel is large. We enclose $T_A(S)$ in an appropriate orthogonal bounding square B , add a segment connecting $T_A(S)$ and B and thus obtain a *spacious* weakly simple polygon P (Fig. 2(b)). It suffices to subdivide P into polygonal faces of constant geometric dilation such that the total length and the number of vertices increase by at most constant factors.

We augment P with new edges and vertices in two phases. The first phase (Section 3) decomposes a spacious orthogonal polygon into spacious *pocketed mountain* polygons; see Def. 5 and Fig. 2(c). The advantage of mountain polygons is that it is easy to approximate their geometric dilation in terms of the detours corresponding to horizontal and vertical point pairs (Lemma 4). In the second phase (Section 4), we greedily decompose each pocketed mountain polygon into polygons of constant geometric dilation in a top-down plane sweep algorithm: Whenever the portion of a mountain above the sweep line has “critical” vertical or horizontal dilation, we insert new edges that separate this area and an adjacent buffer zone from the rest of the mountain (Fig. 2(d)). The buffer zones make sure that the detour is bounded by a constant for points lying on the newly inserted edges.

2.2 Reduction to an axis-aligned subdivision

Ebberts-Baumann *et al.* [15] proved that the geometric dilation of a plane graph G is attained for the endpoints of a *visibility segment* uv (where $u, v \in G$ but the relative interior $\text{relint}(uv)$ of the segment uv is disjoint from G). In our final graph G , any pair of visible points lie on the boundary of a bounded (polygonal)

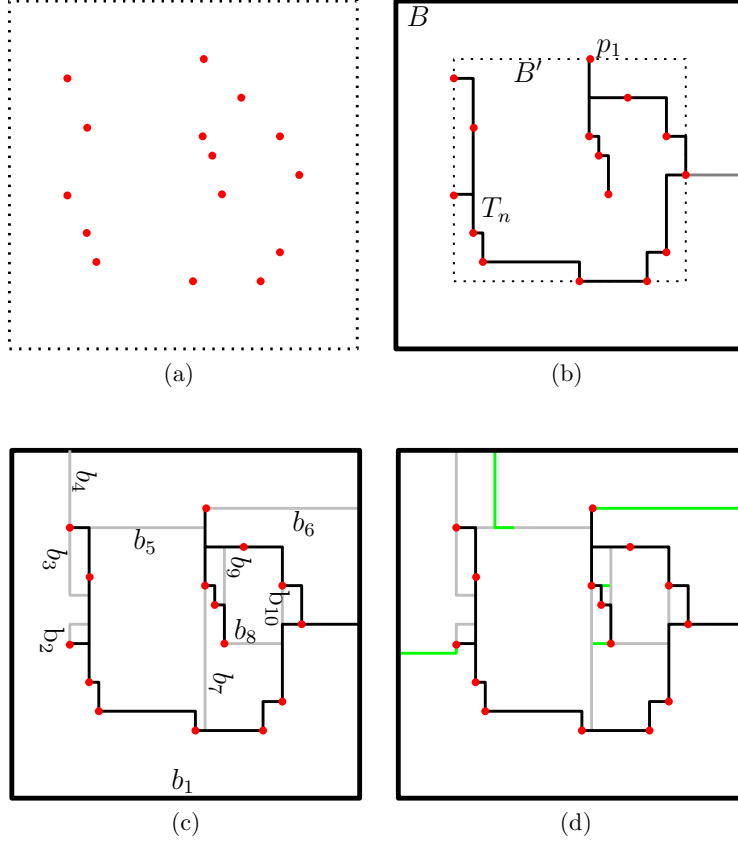


Figure 2: The three main steps of our algorithm. (a) A point set S ; (b) a rectilinear Steiner tree $T_A(S)$ and a bounding square B jointly forming a (weakly simple) polygon; (c) a pocketed mountain subdivision; and (d) a further refined subdivision into polygons of constant geometric dilation.

face of G . The *internal dilation* of a polygon P is

$$\delta_{\text{int}}(P) = \sup \left\{ \frac{d_P(u, v)}{|uv|} : u, v \in P \text{ and } \text{relint}(uv) \subset \text{int}(P) \right\}.$$

Thus if the internal dilation of every bounded face is at most a constant c , then the dilation of G is bounded by the same constant c .

Theorem 2 *For every set S of n points in the plane, there is an orthogonal subdivision G of a bounding square of S such that (i) the internal dilation of every bounded face of G is $O(1)$; (ii) G has $O(n)$ vertices; and (iii) the length of the subdivision G is $O(W)$. Such a subdivision can be computed in $O(n \log n)$ time.*

It is easy to see that Theorem 1 follows from Theorem 2; in particular, conditions (ii) and (iii) are the same.

2.3 Reduction to spacious orthogonal polygons

Given a set S of n points in the plane, we first construct a Steiner spanning tree $T_A(S)$ whose vertex set contains S . Ideally, $T_A(S)$ is the *rectilinear Steiner minimum tree* (RSMT) of S , which has at most $2n - 1$ vertices and whose length is at most $\sqrt{2}W$. One crucial property of the RSMT is that it is *1-spacious*, as defined below.

Definition 1 Given an orthogonal PSLG G and a parameter $\kappa \geq 1$, a κ -narrow channel is a rectangle r of aspect ratio greater than κ such that (refer to Fig. 3(a))

- the two longer sides of r are each contained in an edge of G ;
- the interior of r is disjoint from G .

Definition 2 An orthogonal PSLG G is κ -spacious, if it has no κ -narrow channel. An orthogonal polygon P is internally κ -spacious if it has no κ -narrow channel r with $r \subset \text{dom}(P)$.

Remark. For an orthogonal κ -spacious polygon P , κ -narrow channels can neither be present inside nor outside the polygon.

By definition, if a PSLG G is κ_1 -spacious, then it is also κ_2 -spacious for every $\kappa_2 \geq \kappa_1$. It is easy to see that every RSMT T for a point set is 1-spacious: If an RSMT T of a point set contained a 1-narrow channel r , then one could construct a shorter RSMT by replacing a portion of T along a longer side of r with a shorter side of r (see Fig. 3(b-c)).

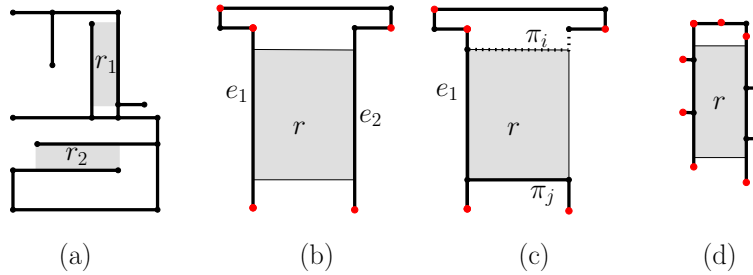


Figure 3: (a) A PSLG with two 3-narrow channels (highlighted) but no 4-narrow channel. (b-c) If an orthogonal Steiner tree T is not 1-spacious, then it is not minimal. (d) This argument does not work if one of the longer sides of r passes through any (input or Steiner) vertex of G .

Although, it is NP-hard to construct an RSMT for a given point set [18], the problem admits a PTAS [4]. Instead of an RSMT, we construct in $O(n \log n)$ time a tree $T_A(S)$ satisfying property (\star) defined below.

Definition 3 An orthogonal spanner network G for a set S of n points has property (\star) if

1. it is 4-spacious,
2. it has $O(W)$ length, and
3. it has $O(n)$ vertices.

Constructing a Steiner spanning tree T_A with property (\star) . It is not difficult to construct a rectilinear Steiner spanning tree satisfying property (\star) for a set S of n points in the plane. Thomborson (see [7]) suggested the following Kruskal-type algorithm: Initially, let the input points form a forest G_0 with n singleton components. In each step, find two components whose L_1 -distance is minimal and connect them with an axis-aligned straight line segment or with an axis-aligned L-shape between a vertex of one component and a point (at a vertex or on an edge) of the other. The L-shaped path is always oriented so that the horizontal edge is at the bottom. The algorithm computes a rectilinear Steiner spanning tree T_A in $n - 1$ steps. Each step creates at most one Steiner vertex, so the output has no more than $2n - 1$ vertices. By Kruskal's result [25], the resulting rectilinear Steiner tree cannot be heavier than the minimum rectilinear spanning tree (which has no Steiner points but the edge length is measured in L_1 norm); which in turn has weight at most $\sqrt{2}W$.

The above tree T_A is also 1-spacious: Assume that the two longer sides of a 1-narrow channel r lie along two parallel edges e_1 and e_2 of T_A . Refer to Fig. 3(b-c). We may assume that e_1 was created prior to e_2 ; and at the step when e_2 was created, it was part of a shortest orthogonal path π connecting components C_i and C_j of the current forest. Observe that edge e_1 cannot belong to both C_i and C_j . By removing $e_2 \cap r$ and adding the two shortest sides of r , we obtain two orthogonal paths π_i and π_j that connect some points in e_1 to C_i and C_j . Note that both π_i and π_j are strictly shorter than π (since the aspect ratio of r is greater than 1). Thus the algorithm would not make the assumed connection π through e_2 ; a contradiction. Hence, the tree T_A does not have any 1-narrow channel.

Wee *et al.* [30, Section 5] implemented the above algorithm in $O(n \log n)$ space and $O(n \log^2 n)$ time, which we show can be reduced to $O(n \log n)$. The bottleneck of their implementation is a semi-dynamic (insert only) data structure that stores the n input points and $O(n)$ Steiner points (inserted one-by-one) and, for a query point (a, b) , can report a point in the two-sided range $\{(x, y) : x \leq a \text{ and } y \leq b\}$ with maximum weight $cx + dy$ for some fixed $c, d \in \mathbb{R}$. Note also that every Steiner point in Thomborson's construction has the same x - and y -coordinate as some input points, so all points in the data structure lie in the $n \times n$ grid generated by the n input points. Wee *et al.* used the *fully dynamic* range tree data structure of Bentley [6] and Willard [31], which takes $O(n \log n)$ preprocessing time and space, and $O(\log^2 n)$ update and query time. Hence, $O(n)$ insertions and queries are done in $O(n \log^2 n)$ total time and $O(n \log n)$ space. Even the best currently known *fully dynamic* data structure for orthogonal range searching, by Nekrich [29], would only improve the time bound to $O(n(\log n / \log \log n)^2)$. However, we can use the *semi-dynamic* data structure of Imai and Asano [23, Section 3.5], which takes $O(n \log n)$ preprocessing time and space, and $O(\log n)$ amortized insertion and query time. With this data structure, the total runtime of Thomborson's construction of the Steiner tree T_A is only $O(n \log n)$.

Reduction. Let B' be the minimum axis-aligned bounding box of S , and let B be a square of side length $2W$ containing B' which extends B' by at least $W/2$ in each direction. Let now $P = P(B)$ be the PSLG formed by the union of B , a tree T_A with property (\star) , and an axis-parallel segment connecting a vertex of T_A lying on B' to the closest point on B ; see Fig. 2(b).

Proposition 1 P is an orthogonal polygon with property (\star) .

Proof: First we show that P has property (\star) . P has at most $O(n) + 4 + 1 = O(n)$ vertices since T_A has $O(n)$ vertices and there are 5 more vertices on the bounding box B . Since T_A is 1-spacious, P is also 1-spacious, therefore also 4-spacious. Its length is $|P| \leq (4 \cdot 2 + 1)W + |T_A| = O(W)$. Note also that P has exactly one bounded face, which is simply connected and lies inside B . Hence, P is a *polygon* in our terminology. \square

The following theorem immediately implies Theorem 2.

Theorem 3 Every 4-spacious orthogonal polygon P with n vertices has an orthogonal subdivision G such that (i) the internal dilation of every face of G is $O(1)$; (ii) G has $O(n)$ vertices; and (iii) the length of the subdivision G is $O(|P|)$. Such a subdivision can be computed in $O(n \log n)$ time.

Overview. In Sections 3 and 4, we prove Theorem 3 and present an algorithm that constructs an orthogonal subdivision G for a given 4-spacious orthogonal polygon P with n vertices. This algorithm has two phases. In the first phase, we decompose P into 4-spacious pocketed mountains (Section 3). In the second phase, we decompose each 4-spacious pocketed mountain into orthogonal polygons of constant internal dilation (Section 4).

In both phases, we augment P with new edges and vertices. We charge every new vertex to old vertices such that each vertex of P is charged at most a constant number of times. Similarly, we charge the length of

every new edge to portions of edges of P of the same length such that each point of P is charged at most a constant number of times.

3 Subdividing spacious orthogonal polygons

In this section, we subdivide a 4-spacious orthogonal polygon into certain 4-spacious polygons of a special structure, namely 4-spacious pocketed mountains. Specifically, we prove the following lemma at the end of this section.

Lemma 1 *Every 4-spacious orthogonal polygon P with n vertices has an orthogonal subdivision G such that every bounded face is a 4-spacious pocketed mountain polygon (defined below); (ii) G has $O(n)$ vertices; and (iii) the length of the subdivision G is $O(|P|)$. Such a subdivision can be computed in $O(n \log n)$ time.*

We start with defining mountain polygons, pockets, and pocketed mountains.

Definition 4 (Refer to Fig. 4, left.) A vertical mountain (alternatively, histogram) is an orthogonal polygon P that has a special horizontal edge b , called base, such that for every point $u \in \text{dom}(P)$ there is a vertical segment $uv \subset \text{dom}(P)$ that connects u to a point $v \in b$. A horizontal mountain is defined analogously (with a vertical base and horizontal segments uv).

Ideally, we would like to subdivide a 4-spacious polygon into 4-spacious mountain polygons. However, our algorithm sometimes creates 4-narrow channels in intermediate steps. The concept of a “pocket” allows us to efficiently dispose of any such 4-narrow channel. Intuitively, we augment the polygonal domain $\text{dom}(P)$ of a polygon P with a rectangle (called “pocket”) adjacent to P . This deformation of P can be thought of as a perturbation that replaces a portion of an edge of P by a polyline in the proximity of the original edge.

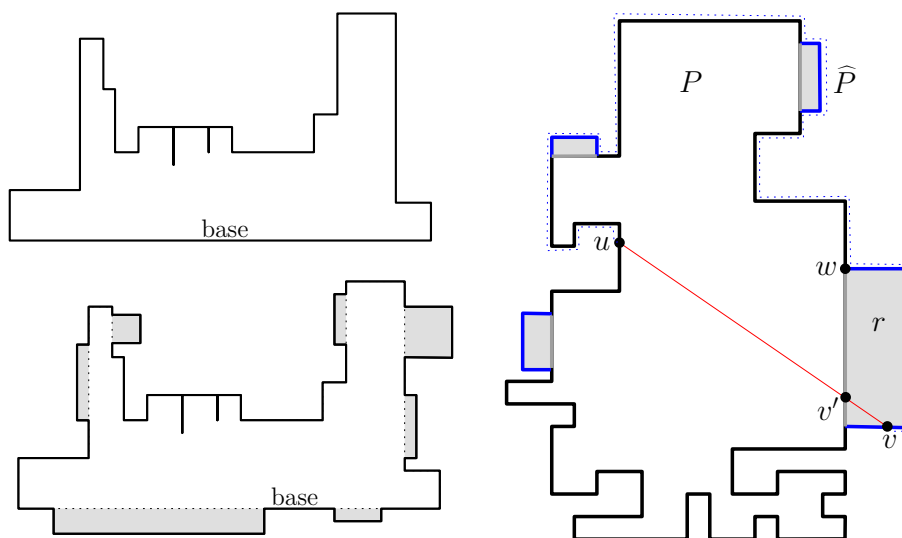


Figure 4: Upper left: a vertical mountain polygon. Lower left: a vertical pocketed mountain polygon with 7 pockets. Right: a pocketed polygon \hat{P} composed of an orthogonal polygon P and 4 pockets.

Definition 5 A pocketed polygon \widehat{P} is an orthogonal polygon with the following representation¹: an orthogonal polygon P and a finite set of rectangles, called pockets, such that (refer to Fig. 4)

- the rectangles are pairwise disjoint and also disjoint from the interior of P ;
- one of the longer sides s_r of each rectangle r is contained in an edge of P .

The polygon \widehat{P} is obtained by replacing each segment s_r by the polygonal path $r \setminus s_r$ composed of three segments. In particular, if M is a mountain polygon, then \widehat{M} is a pocketed mountain polygon.

We first show that a pocketed polygon \widehat{P} cannot have much larger internal dilation than P .

Lemma 2 Every pocketed polygon \widehat{P} satisfies $\delta_{\text{int}}(\widehat{P}) \leq 3\delta_{\text{int}}(P)$.

Proof: Consider two points $u, v \in \widehat{P}$ for which the internal dilation of \widehat{P} is attained. We distinguish three cases:

Case 1: $u, v \in P$. Since every segment $s_r, r \in R$, was replaced by a path of length at most $3|s_r|$, we have $d_{\widehat{P}}(u, v) \leq 3d_P(u, v) \leq 3\delta_{\text{int}}(P)|uv|$.

Case 2: $u \in P$, and $v \in \widehat{P} \setminus P$, that is, v lies on the boundary of a pocket r . (Refer to Fig. 4, right.) Let v' denote the intersection points of uv with $P \cap \text{int}(\widehat{P})$, and assume that the shortest path $d_P(u, v')$ passes through vertex w of $r \cap P$. We have $d_{\widehat{P}}(u, v) \leq d_{\widehat{P}}(u, w) + d_{\widehat{P}}(w, v) \leq 3d_P(u, w) + 3(|wv'| + |v'v|) \leq 3d_P(uv') + 3|v'v|$. Hence the detour between u and v in \widehat{P} can be bounded by

$$\frac{d_{\widehat{P}}(u, v)}{|uv|} \leq \frac{3d_P(u, v') + 3|vv'|}{|uv'| + |vv'|} \leq \max \left\{ \frac{3d_P(u, v')}{|uv'|}, \frac{3|vv'|}{|vv'|} \right\} \leq 3\delta_{\text{int}}(P).$$

Case 3: $u, v \in \widehat{P} \setminus P$ (both u and v lie on the boundary of pockets). If u and v are in two distinct pockets, then we can argue analogously to the previous case. If u and v lie in the same pocket, then their detour is at most $3 \leq 3\delta_{\text{int}}(P)$.

In all three cases, we have $d_{\widehat{P}}(u, v) \leq 3\delta_{\text{int}}(P)|uv|$ for $u, v \in \widehat{P}$, hence $\delta_{\text{int}}(\widehat{P}) \leq 3\delta_{\text{int}}(P)$. \square

A subroutine for eliminating 4-narrow channels. We say that a κ -narrow channel r of a graph G is adjacent to a face f of G , if r lies in a bounded face of G adjacent to f and one of the longer sides of r is contained in an edge of f . The subroutine `AddPockets` modifies G by attaching pockets to f to “fill” all the adjacent 4-narrow channels. The pockets attached to f are slightly shorter than the adjacent narrow channels. This gap guarantees that the edges created (on the boundary of pockets) are sufficiently far from other edges and thus do not form new 4-narrow channels.

Algorithm 1 `AddPockets`(G, f)

Input: PSLG G and a face f of G . (Refer to Fig. 5(b-c), where $f = M_b$.)

Output: a PSLG \widehat{G} and its face \widehat{f} .

Find the set R of all maximal 4-narrow channels adjacent to f in G . For each $r \in R$, let r' be the rectangle obtained from r by removing two squares adjacent to the two shorter sides of r . Let \widehat{f} be the pocketed polygon determined by f and the rectangles $r', r \in R$. Let $\widehat{G} = (G \setminus f) \cup \widehat{f}$. Return \widehat{G} and \widehat{f} .

¹This representation is not necessarily unique.

Lemma 3 *Let G be an orthogonal PSLG and let f be one of its faces. If every 4-narrow channel of G that lies outside of f is adjacent to f , then the output of $\text{AddPockets}(G, f)$ has no 4-narrow channel outside of \hat{f} .*

Proof: One side of every 4-narrow channel of G outside of f must be contained in an edge of f . The procedure $\text{AddPockets}(G, f)$ removes a large portion of this side from G so that remaining portions of the edges do not form a 4-narrow channel anymore. Now consider a newly created edge e (one of the shorter edges of a rectangle r' , $r \in R$). If the graph produced by $\text{AddPockets}(G, f)$ has a 4-narrow channel adjacent to e , the channel would lie in a square along e , which is empty by construction. So e cannot participate in any 4-narrow channel, either. \square

Proposition 2 *For any orthogonal PSLG G and face f , subroutine $\text{AddPockets}(G, f)$ does not increase the length of G .*

Proof: Let R be as defined in subroutine $\text{AddPockets}(G, f)$. For every rectangle $r \in R$, subroutine $\text{AddPockets}(G, f)$ removes the edge of r' adjacent to f and augments G with the two shorter edges of r' . Since the aspect ratio of r' is at least 2, the length of G does not increase. \square

For the proof of Lemma 1, we present a recursive algorithm, $\text{BuildMountains}(\hat{P}, b)$, whose input is a 4-spacious pocketed polygon \hat{P} and an edge b , called *base*, of the corresponding polygon P ; it recursively builds pocketed mountains induced by a base (Definition 6); its output is a subdivision of \hat{P} into 4-spacious pocketed mountains. Initially, we choose an arbitrary side of P as the base b . Refer to Fig. 5(a-f).

Definition 6 *Given a pocketed polygon \hat{P} (represented by a polygon P and some pockets) and an edge b on the boundary of P , let the pocketed mountain polygon induced by b with respect to \hat{P} be the boundary of the region formed by the union of all maximal segments that lie in $\text{dom}(\hat{P})$, are orthogonal to b , and have an endpoint in b .*

Algorithm 2 $\text{BuildMountains}(\hat{P}, b)$.

Input: a 4-spacious pocketed polygon \hat{P} (represented as P and some pockets) and an edge b of P .

Output: a subdivision G of the polygon \hat{P} into 4-spacious pocketed mountain polygons.

1. Construct the pocketed mountain M_b induced by b with respect to \hat{P} , and let $G = \hat{P} \cup M_b$ (Fig. 5(a-b)).
2. Call $\text{AddPockets}(G, M_b)$ (Fig. 5(b-c)). [This modifies G and creates \widehat{M}_b .]
3. Let \mathcal{P}_b denote the set of all faces of G with the exception of \widehat{M}_b . For every face $f \in \mathcal{P}_b$,
 - (a) let the base edge $b(f)$ be the unique common edge of f and $\widehat{M}_b \setminus \hat{P}$ orthogonal to b (Fig. 5(d));
 - (b) call $\text{AddPockets}(G, f)$ (Fig. 5(e-f)) [this modifies G and creates \hat{f} .];
 - (c) let $G = G \cup \text{BuildMountains}(f, b(f))$.
4. Return G .

Proof of Lemma 1: We analyze Algorithm 2. We may assume w.l.o.g. that b is horizontal and P lies above b in the invocation we consider. Step 1 creates a pocketed vertical mountain polygon M_b (where all pockets, if any, lie below b). Then $\text{AddPockets}(G, M_b)$ creates \widehat{M}_b by attaching pockets to the vertical edges of M_b . Finally, subroutines $\text{AddPockets}(G, f)$, $f \in \mathcal{P}_b$, may shrink \widehat{M}_b to a weakly simple polygon, denoted \widehat{N}_b , whose interior may be disconnected. Let \mathcal{N}_b denote the set of bounded faces of \widehat{N}_b . A recursive call subdivides P into the set of polygons $\mathcal{N}_b \cup \widehat{\mathcal{P}}_b$, where $\widehat{\mathcal{P}}_b = \{\hat{f} : f \in \mathcal{P}_b\}$ (Fig. 5(f)). The polygons in

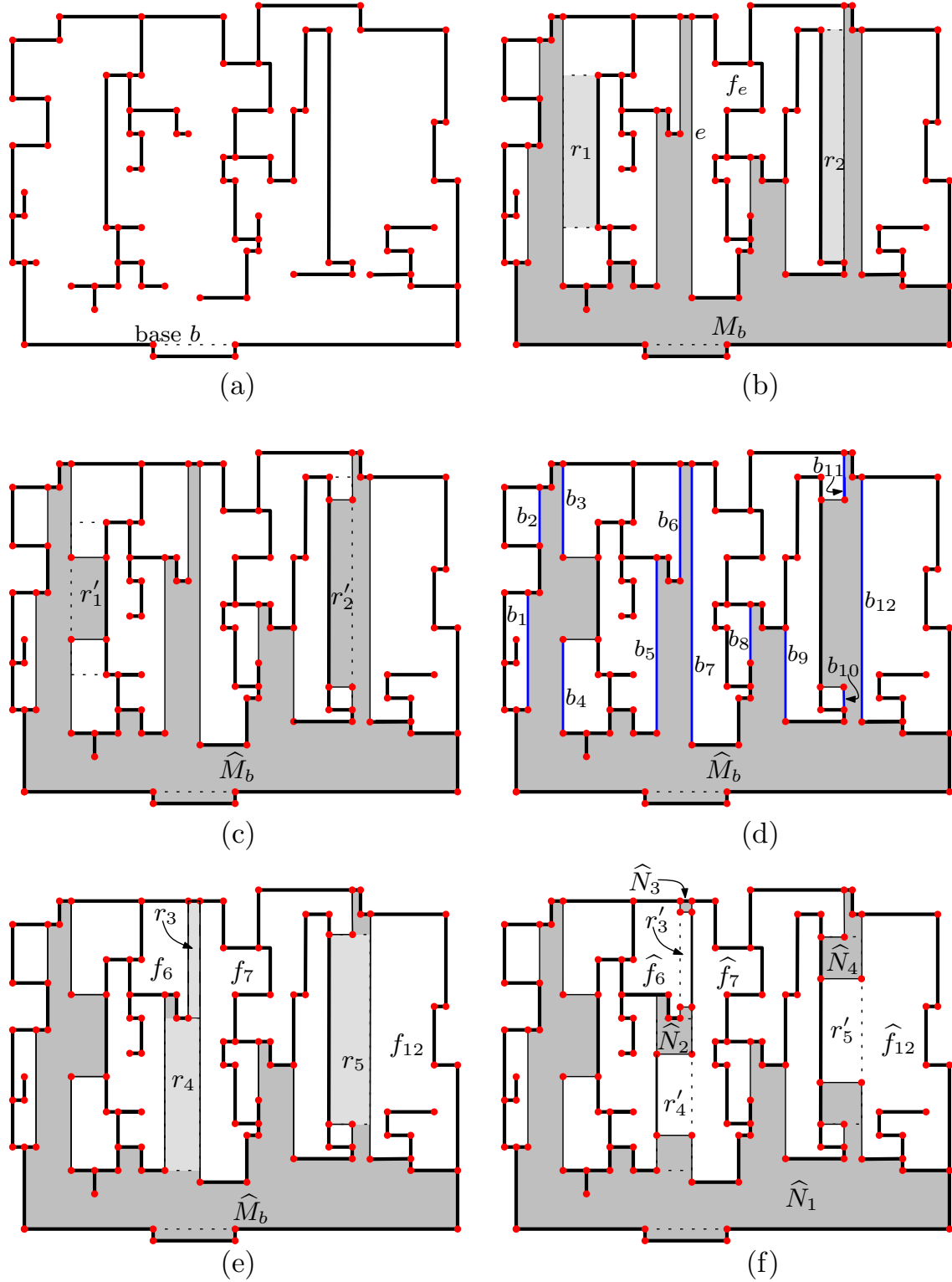


Figure 5: (a) A polygon \widehat{P} with a (pocketed) base b . (b) The pocketed mountain polygon M_b induced by the base b w.r.t. \widehat{P} is adjacent to two 4-narrow channels r_1 and r_2 . An edge $e \subset M_b \setminus \widehat{P}$ is adjacent to M_b and to a face f_e of the current subdivision. (c) Subroutine $\text{AddPockets}(G, M_b)$ extends M_b to a polygon \widehat{M}_b . (d) The base of every face in \mathcal{P}_b is the vertical edge along $\widehat{M}_b \setminus \widehat{P}$. (e) There are three 4-narrow channels r_3 , r_4 , and r_5 in $\text{dom}(\widehat{M}_b)$. (f) Subroutines $\text{AddPockets}(G, f_6)$, $\text{AddPockets}(G, f_7)$, and $\text{AddPockets}(G, f_{12})$ split \widehat{M}_b into a set of polygons $\mathcal{N}_b = \{\widehat{N}_1, \widehat{N}_2, \widehat{N}_3, \widehat{N}_4\}$.

\mathcal{N}_b are removed from further consideration in the remainder of Algorithm 2, and we recurse on the polygons in $\widehat{\mathcal{P}}_b$.

4-Spaciousness is preserved. First we show that if \widehat{P} is 4-spacious, then every polygon in $\mathcal{N}_b \cup \widehat{\mathcal{P}}_b$ is also 4-spacious. In one recursive call, only step 1 may create 4-narrow channels (Lemma 3). After step 1, every 4-narrow channel is either contained in M_b or adjacent to M_b ; and one of its longer sides is contained in a vertical edge of $M_b \setminus \widehat{P}$. Subroutine `AddPockets(G, M_b)` eliminates all 4-narrow channels adjacent to M_b . In the resulting subdivision, every 4-narrow channel lies in $\text{dom}(\widehat{M}_b)$ and at least one of its longer sides is contained in a vertical edge of $\widehat{M}_b \setminus \widehat{P}$ (both longer sides cannot be contained in edges of \widehat{P} , since \widehat{P} is 4-spacious). Every edge of $\widehat{M}_b \setminus \widehat{P}$ separates \widehat{M}_b and a face $f \in \mathcal{P}_b$; and so `AddPockets(G, f)` destroys all 4-narrow channels adjacent to f . Hence at the end of the recursive call, all faces in $\mathcal{N}_b \cup \widehat{\mathcal{P}}_b$ are 4-spacious, so the pocketed input polygons in all recursive subproblems are 4-spacious. By induction, the output subdivision G of the input polygon \widehat{P} is also 4-spacious.

Pocketed mountains are produced. Next we show that every polygon $\widehat{N} \in \mathcal{N}_b$ is a pocketed vertical mountain. Polygon M_b is a pocketed mountain, where all pockets are adjacent to the base b (they are the pockets of \widehat{P} along b). Subroutine `AddPockets(G, M_b)` may attach pockets to some vertical edges in $M_b \setminus \widehat{P}$, and \widehat{M}_b is again a pocketed vertical mountain. Then `AddPockets(G, f)` can attach some pockets r' to some faces $f \in \mathcal{P}_b$, where $r' \subset r$ for a 4-narrow channel $r \subset \text{dom}(\widehat{M}_b)$. Refer to Fig. 5(e-f). Assume that the pockets are added to the faces $f \in \mathcal{P}_b$ sequentially, and let us consider only one operation, where a pocket r' splits a polygon $\widehat{N} \subset \widehat{M}_b$ into two polygons, say \widehat{N}_1 below r' and \widehat{N}_2 above r' . We show that if \widehat{N} is a pocketed vertical mountain, then both \widehat{N}_1 and \widehat{N}_2 are pocketed vertical mountains, too. \widehat{N} can be represented as a vertical mountain N and some attached pockets. Rectangle r' splits N into two vertical mountains N_1 and N_2 . If a pocket of \widehat{N} is disjoint from r' , then it will be a pocket of \widehat{N}_1 or \widehat{N}_2 . If r' intersects a pocket p of \widehat{N} , then one of the longer side of the 4-narrow channel r is contained in an edge of pocket p . The rectangle r' partitions p it into two rectangles, say p_1 below r' and p_2 above r' . Clearly, p_1 is adjacent to an edge of N_1 , and p_2 is adjacent to an edge of N_2 . Since $r \setminus r'$ consists of two squares below and above r' , the side of p_1 (resp., p_2) adjacent to N_1 (resp., N_2) is longer than its orthogonal sides, which means that p_1 is a pocket of N_1 (resp., p_2 is a pocket of N_2).

Termination. Every point vertically visible from the base b is in $\text{dom}(M_b)$. Ideally, at the end of a recursive call, each point of $\text{dom}(M_b)$ lies in some polygon of \mathcal{N}_b . The polygons in \mathcal{N}_b are discarded from further consideration in recursive calls. Some portions of $\text{dom}(M_b)$, however, may be in pockets of some faces $\widehat{f} \in \widehat{\mathcal{P}}_b$. Note that each such pocket is adjacent to the base side $b(f)$ of a polygon $f \in \mathcal{P}_b$, and will be included in a polygon $\widehat{N} \in \mathcal{N}_{b(f)}$ in the next level of the recursion (i.e., level in the recursion tree). This means that all points of $\text{dom}(M_b)$ are removed from further consideration either in the call where M_b is created, or in a call at the next level of recursion (in which they lie below the base of some $M_{b(f)}$, $f \in \mathcal{P}_b$). We show that any point $x \in \text{int}(P)$ is contained in $\text{dom}(M_b)$ for some base b at depth at most n in the recursion tree. For $x \in \text{int}(P)$, let $f_i(x)$ denote the polygon at depth i of the recursion that contains x ; and let $\pi_i(x)$ denote the minimum integer such that there is an orthogonal polyline that consists of $\pi_i(x)$ segments and connects x to the base segment of $f_i(x)$ within $\text{int}(f_i(x))$. Initially, $\pi_0(x) < n$, since P has n vertices and so any two points can be connected by an orthogonal polyline of less than n segments. Since all points orthogonally visible from a base are either discarded or will be below the base in the next level of recursion, we have $\pi_{i+1}(x) < \pi_i(x)$ for every x and i , hence Algorithm 2 terminates within at most n levels of recursion.

Charging scheme for length. We show that the total length of the new edges created in algorithm `BuildMountains(\widehat{P}, b)` is at most $2|\widehat{P}|$ by charging every new edge to portions of edges of the input polygon \widehat{P} of the same length such that every portion is charged at most twice. Denote by E_0 the set of edges of \widehat{P} . Let E_1 be the set of new edges constructed in all steps 1 throughout the algorithm. By Propo-

sition 2, subroutine `AddPockets` does not increase the length of the given PSLG. Hence, it is enough to show that $|E_1| \leq 2|\widehat{P}|$.

Assume that the base b is horizontal in the recursive call we consider. The new edges constructed in step 1 of this call are the edges in $M_b \setminus \widehat{P}$ and they are all vertical. Every edge $e \subseteq M_b \setminus \widehat{P}$ is adjacent to M_b and some other face f_e of $\widehat{P} \cup M_b$ (Fig. 5(b)). The boundary of f_e is composed of e and portions of \widehat{P} . Project e horizontally on $f_e \setminus e$. The projection is composed of vertical portions of the input polygon \widehat{P} . Charge the length of e to the projection. We show that each portion of \widehat{P} is charged at most twice, at most once from each side. A segment $s \subset \widehat{P}$ in the projection of e either lies on the boundary of a polygon in \mathcal{N}_b (and is discarded from further consideration), or it is on the boundary of a polygon of $\mathcal{N}_{b(f)}$ for some $f \in \mathcal{P}_b$ and discarded in the following level of the recursion. However, at the next level of recursion, all new edges are charged to horizontal portions of \widehat{P} , so s cannot be charged. In either case, within two consecutive recursive calls, any polygon adjacent to s and containing e is discarded from any recursive subproblem, and so s cannot be charged again from this side. The output subdivision G is the union of \widehat{P} and the new edges of total length at most $2|\widehat{P}|$, hence $|G| \leq 3|\widehat{P}|$.

Charging scheme for vertices. We give an upper bound on the number of Steiner vertices created in Algorithm 2. Every edge in E_1 is incident to a reflex vertex v of \widehat{P} ; and every reflex vertex of \widehat{P} is incident to at most one edge of E_1 . The reason for this is that if v is incident to a new edge of some mountain M_b , then v becomes a convex vertex in the recursive subproblems. Note also that if a new vertex w is *reflex* in a subproblem \widehat{P}_1 , then w lies along the base of \widehat{P}_1 and so w is not incident to any new edge of E_1 constructed in that subproblem. Hence, we have $\#E_1 \leq n$. Each edge of E_1 is incident to a vertex of \widehat{P} and a potentially new vertex, so the construction of polygons M_b throughout Algorithm 2 increases the number of vertices by at most n . Each pocket added in a `AddPockets` subroutine increases the number of vertices by 4, with the four corners of a rectangle r' . Next, we deduce an upper bound on the total number of pockets created in Algorithm 2.

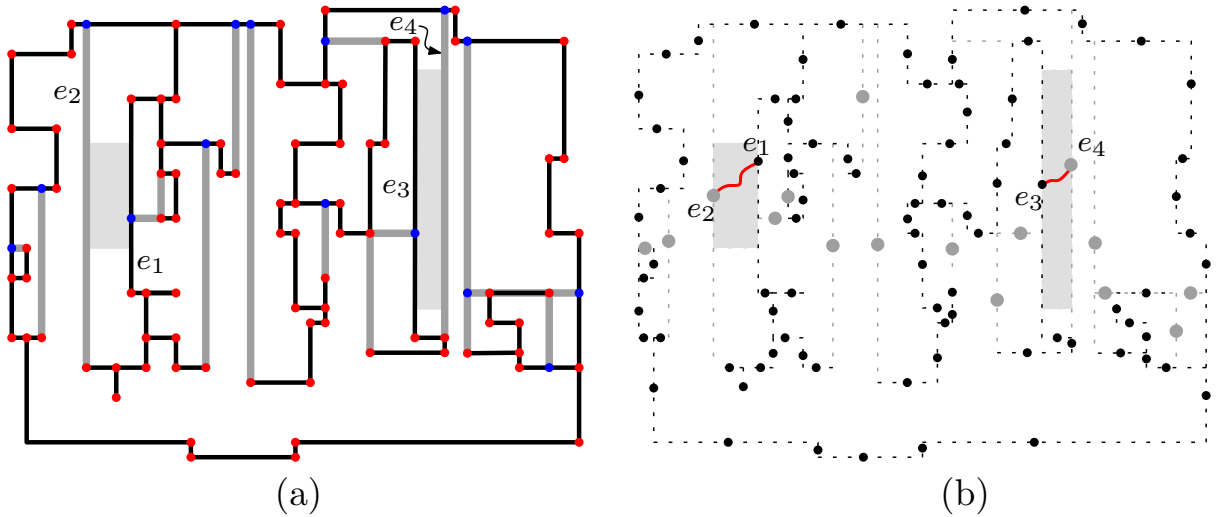


Figure 6: (a) The edges in E_0 (in black) and E_1 (in gray) from the instance in Fig. 5. One pocket was created between edges $e_1 \in E_0$ and $e_2 \in E_1$, and another one between $e_3 \in E_0$ and $e_4 \in E_1$. (b) By contracting each edge in E_0 and E_1 to a node, and drawing a curve between the edges along a pocket, we obtain a bipartite plane graph.

First consider the pockets created in subroutines `AddPockets`(G, M_b) for all bases b . Every such pocket lies between an edge of E_0 and a parallel edge of E_1 , and every pair of parallel edges in $E_0 \times E_1$

is responsible for at most one pocket. Note that the segments in $E_0 \cup E_1$ are pairwise noncrossing. By drawing a curve in each pocket that connects the two corresponding edges of E_0 and E_1 , we obtain a plane bipartite graph (see Fig. 6) where the two vertex classes correspond to E_0 and E_1 (each segment can be contracted to a single point). A plane bipartite graph on $\#(E_0 \cup E_1)$ vertices has less than $2\#(E_0 \cup E_1)$ edges by Euler’s polyhedron theorem. Since $\#E_0 + \#E_1 \leq n + n = 2n$, the number of these pockets is less than $4n$. These pockets also split some edges of E_1 into several segments. Denote by E_2 the set of all resulting subsegments of E_1 (including the edges of E_1 that are not split). Each pocket partitions an edge of E_1 into two pieces, so we have $\#E_2 \leq \#E_1 + 4n \leq 5n$. Now consider the pockets created in subroutines $\text{AddPockets}(G, f)$ for all $f \in \mathcal{P}_b$ and all b throughout the algorithm. Each such pocket lies between an edge of E_2 and a parallel edge of $E_0 \cup E_2$. Similar to the previous argument, the number of these pockets is at most $2\#(E_0 \cup E_2) \leq 2(n + 5n) = 12n$. The subdivision G of the input polygon \widehat{P} has at most $n + n + 4(4n + 12n) = 66n$ vertices.

An $O(n \log n)$ time implementation. For computing the subdivision G , we maintain the polygonal faces in every level of the recursion. The faces for which BuildMountains is applied in the same level of recursion are the *active* polygons of that level. The active polygons have pairwise disjoint interiors. We also maintain a ray shooting data structure [22] that stores all axis-parallel rays emanating from vertices in every active polygon. Note that the active polygons and all vertical (resp., horizontal) rays form a PSLG. The size of this data structure is $O(n)$, since the total number of vertices is $O(n)$.

Initially we can compute all axis-parallel rays emanating from the vertices of the input polygon \widehat{P} in four line sweeps in $O(n \log n)$ time. To compute the rays emanating from new vertices, we use a linear space dynamic data structure of Giyora and Kaplan [19] that supports vertical (resp., horizontal) ray-shooting queries, segment insertions, and deletions in $O(\log n)$ time. We can also maintain the already existing rays in $O(n \log n)$ time; we need to truncate a ray if it crosses a new edge of G . Each truncation of a ray can be done in $O(1)$ time. Each ray in our data structure is truncated in at most two consecutive recursive calls: If a ray \vec{a} emanating from a vertex v is truncated (a new edge in $E_1 \cup E_2$ crosses \vec{a}) in one recursive call, then v will be on the boundary of a polygon in \mathcal{N}_b in either this call or in a call at the next level of recursion, and \vec{a} is discarded from further consideration. Since an existing vertical (horizontal) ray can cross up to $O(n)$ new horizontal (vertical) edges, we need to make sure that no ray is truncated too many times in any invocation. In each recursive call, we sort all new vertical (horizontal) edges by their x -coordinates (y -coordinates) in a balanced binary search tree and insert them into G in the tree order. This guarantees that each ray is truncated at most $O(\log n)$ times at each level of the recursion. Altogether, we maintain a data structure of all axis-parallel rays emanating from vertices of the active polygons in $O(n \log n)$ total time.

Besides maintaining all axis-parallel rays emanating from all vertices in the active polygons, the operations of Algorithm 2 take $O(n)$ additional time. We construct the mountain polygons M_b in output-sensitive linear time (by simply walking around the weakly simple polygon M_b , which consists of edges of \widehat{P} and vertical rays emanating from certain reflex vertices), and we detect all 4-narrow channels throughout the algorithm in $O(n)$ total time (since each narrow channel is bounded by two parallel edges such that a ray emanating from an endpoint of one edge hits the other edge). Altogether, the total runtime of Algorithm 2 is $O(n \log n)$. This completes the proof of Lemma 1. \square

4 Subdividing mountains and pocketed mountains

In this section, we first subdivide a simple internally 4-spacious mountain polygon into polygons of constant internal dilation. We then extend this algorithm and subdivide arbitrary (i.e., weakly simple) 4-spacious mountains and 4-spacious *pocketed* mountains, as well, into polygons of constant internal dilation. Simple

mountain polygons are useful because their internal dilation can be bounded in terms of the detours of point pairs in horizontal or vertical position (Lemma 4).

Consider a simple vertical mountain M with a horizontal base b . For every horizontal visibility segment uv with $u, v \in M$ and $uv \subset \text{dom}(M)$, we denote by $d_M^*(u, v)$ the length of the (upper) path between u and v along M that does not contain the base b .

Lemma 4 *The internal dilation of a simple vertical mountain M is at most $\max(\delta_H(M) + 1, \delta_V(M))$, where*

- $\delta_H(M) = \max\{d_M^*(u, v)/|uv| : u, v \in M, uv \subset \text{dom}(M), \text{ and } uv \text{ is horizontal}\}$;
- $\delta_V(M) = |M|/(2|\lambda(M)|)$, where $\lambda(M)$ is the shortest vertical segment uv with $u, v \in M$ and $\text{relint}(uv) \subset \text{int}(M)$.

Proof: Clearly, $\delta_H(M)$ is an upper bound on the detour between a point pair in horizontal position along

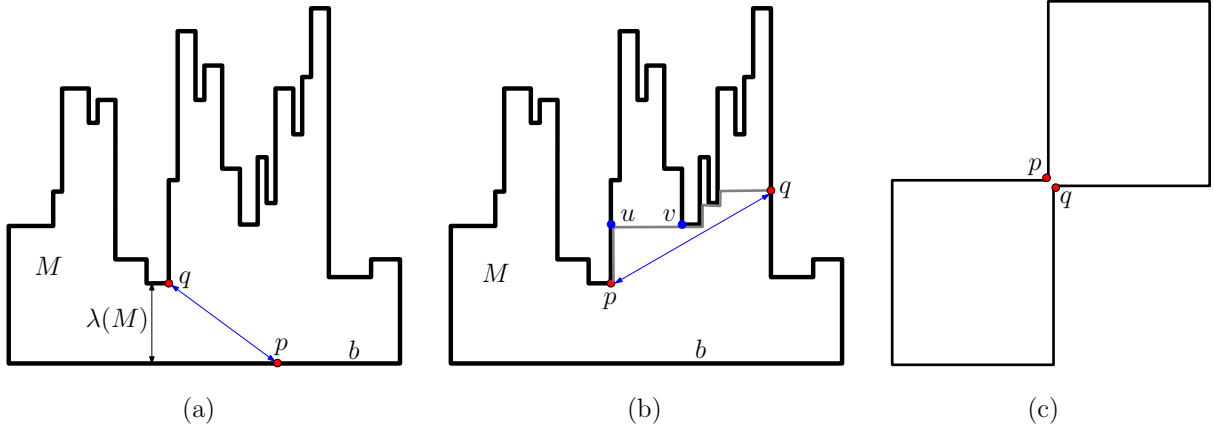


Figure 7: (a) A simple mountain polygon M , the minimum distance $\lambda(M)$ between the base and the upper path of M . (b) A point pair $p, q \in M \setminus b$ and the staircase path between them with all vertical segments lying in M . (c) For an x - and y -monotone orthogonal polygon P , the internal dilation can be arbitrarily large even though $\delta_H(P)$ and $\delta_V(P)$ are bounded by a constant.

M , and $\delta_V(M)$ is an upper bound on the maximal dilation between a point pair in vertical position along M . Refer to Fig. 7.

Consider two points $p, q \in M$ for which the internal dilation of M is attained. We may assume that pq is a visibility segment (that is, $\text{relint}(pq) \subset \text{int}(M)$). We can assume that p and q do not both lie on the base, since their detour would be 1. We distinguish two cases: (1) either p or q is contained in the base b , (2) neither p nor q is contained in the base b .

Case 1: $p \in b$. In this case, $q \notin b$ and so $|pq| \geq \lambda(M)$. Since $d_M(p, q)$ is less than $|M|/2$, we have $d_M(p, q)/|pq| \leq \delta_V(M)$.

Case 2: $p, q \notin b$. Denote by $|pq|_H$ (resp., $|pq|_V$) the length of the horizontal (resp., vertical) projection of segment pq . Let $\pi(p, q)$ be the staircase path (i.e., a monotone orthogonal path) between p and q whose vertical segments all lie in M (see Fig. 7(b)). Clearly, we have $|\pi(p, q)| = |pq|_H + |pq|_V$. The shortest path distance $d_M(p, q)$ is at most the total length of the shortest paths between consecutive vertices of $\pi(p, q)$. Every vertical segment uv along $\pi(p, q)$ is a shortest path and so $d_M(u, v) = |uv|$; and for every horizontal segment uv , the shortest path distance is at most $d_M(u, v) \leq \delta_H(M)|uv|$. Hence, we have $d_M(p, q) \leq \delta_H(M)|pq|_H + |pq|_V < (\delta_H(M) + 1)|pq|$. \square

Note that the internal dilation of an arbitrary simple orthogonal polygon cannot be bounded in terms of detours between horizontal and vertical point pairs. Fig. 7(c) shows that an x - and y -monotone polygon P can have arbitrarily large dilation even though the ratio $d_P(u, v)/|uv|$ is at most 3 for any horizontal or vertical segment $uv \subset \text{dom}(P)$, with $u, v \in P$.

4.1 Subdividing simple mountains into polygons of constant geometric dilation

We now present and analyze an algorithm for subdividing a simple internally 4-spacious mountain polygon into orthogonal polygons of constant internal dilation, namely 4-spacious pocketed mountains. Our algorithm greedily chooses polygons for which the dilation bound of Lemma 4 is above a constant threshold. We prove the following.

Lemma 5 *A simple internally 4-spacious mountain M with n vertices has a 4-spacious orthogonal subdivision G , where (i) the internal dilation of every face of G is at most 189; (ii) the length of G is at most $\frac{4}{3}|M|$; and G has at most $27n$ vertices. Such a subdivision can be computed in $O(n \log n)$ time.*

For every horizontal segment s (not necessarily an edge of M), we define a *padding*, which is a rectangle whose top side is s and whose vertical sides each have length $|s|/4$. Perturb the y -coordinates of horizontal edges of M by a tiny $\varepsilon > 0$, if necessary, so that no two horizontal edges of M are collinear. This perturbation simplifies our algorithm when we scan M with a sweep line (there are no ties). Let \mathcal{H} denote the (infinite) set of all horizontal visibility segments in M , that is, all horizontal segments uv such that $u, v \in M$ and $\text{relint}(uv) \subset \text{int}(M)$.

With a parameter α (which is later set to $\alpha = 9$), we use the following algorithm to subdivide M into 4-spacious mountains. Refer to Fig. 8.

Algorithm 3 `SubdivideMountain(M)`.

Input: a simple internally 4-spacious vertical mountain M that lies above its (horizontal) base b .

Output: a subdivision G of the input polygon M . All bounded faces of G have internal dilation at most 189.

STEP 1. Sweep a horizontal line ℓ from the top edge of M down until one of the following three events (cases) occurs for $uv \in \mathcal{H}$, $uv \subset \ell$.

1. If ℓ reaches the base b of M , then continue with STEP 2.
2. If $d_M^*(u, v) = \alpha|uv|$ (Fig. 8a,c), then do: Let G be the union of M and the lower, left, and right edges of the padding of uv . Let P be the face of G containing uv . Call `AddPockets(G, P)`, which modifies G and creates \hat{P} . Continue with STEP 2.
3. If there is a segment $uv \in \mathcal{H}$, $uv \subset \ell$, whose padding intersects b (Fig. 8d), then do: Let G be the union of M and two vertical segments connecting u and v , resp., to b . Let P be the face of G between u and v . Call `AddPockets(G, P)`, which modifies G and creates \hat{P} .

STEP 2. For every face f of the resulting subdivision G that lies entirely in the closed halfplane below ℓ , let $G = G \cup \text{SubdivideMountain}(f)$ (Fig. 8(e)). Return G (Fig. 8(f)).

Proof of Lemma 5: First we show that in Algorithm 3, every visibility segment uv with endpoints $u, v \in M$ and lying along the sweep line ℓ must be at distance at least $\frac{1}{12}|uv|$ from the base b . Initially, when the sweep line passes through the top horizontal edge of M , we have $d_M^*(u, v) = |uv|$, and the above condition holds since there is no 4-narrow channel in M . The ratio $d_M^*(u, v)/|uv|$ increases while u and v move along vertical edges of M , and it does not increase when ℓ reaches a horizontal edge of M . The padding of uv moves continuously with the sweep line ℓ while u and v move along vertical edges of M , but the padding may increase dramatically when ℓ reaches a horizontal edge of M .

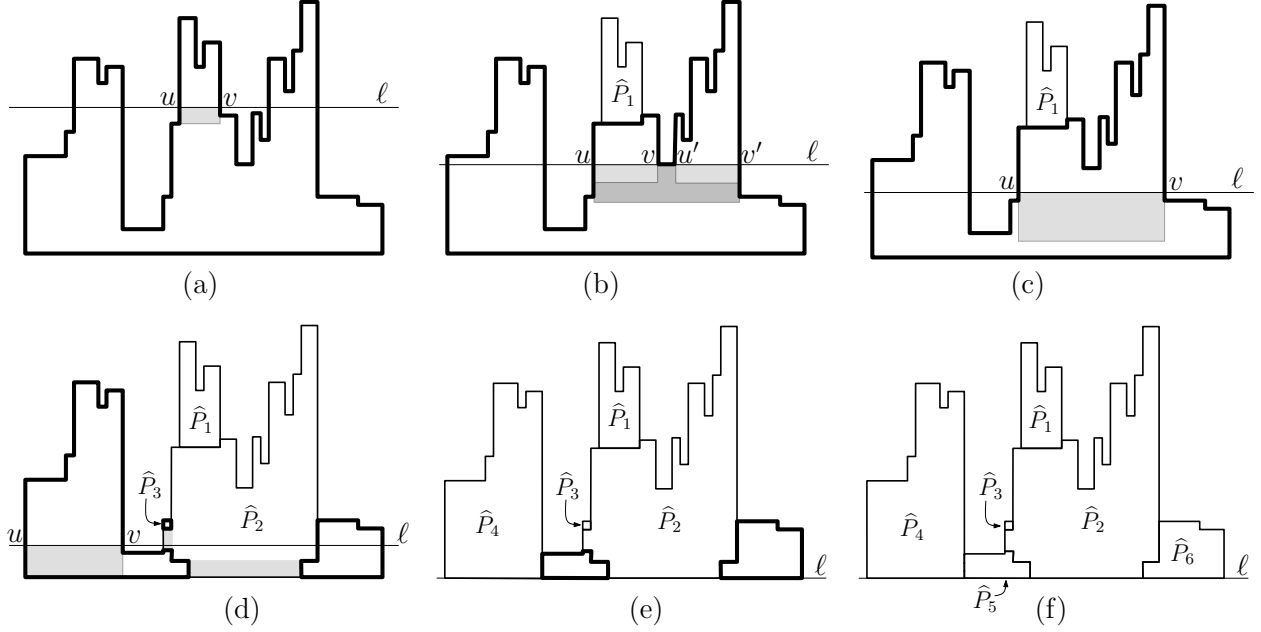


Figure 8: The progress of Algorithm 3 for subdividing a simple mountain polygon. The active polygons are marked with thick lines. (a) Case 2 applies. (b) The sweep line ℓ passes through a horizontal edge of the input mountain. (c) Case 2 applies. (d) The sweep line has reached the base of polygon \widehat{P}_3 and at its current position case 3 applies to uv . (e) The sweep line ℓ reaches the base of all active polygons. (f) The resulting subdivision.

Let us consider the case that ℓ passes through a horizontal edge $vu' \subset M$ and two visibility segments, uv and $u'v'$ (Fig. 8b). Segment vu' is a single edge of M , since we assume that no two edges of M are collinear. The paddings of uv and $u'v'$ are disjoint from b (otherwise case 3 would have occurred earlier), and the padding of the edge vu' is also disjoint from b (since M is 4-spacious). Therefore, the distance between ℓ and b is at least $\frac{1}{4} \max(|uv|, |vu'|, |u'v'|) \geq \frac{1}{12}|uv'|$. Hence the segment uv' is at distance at least $\frac{1}{12}|uv'|$ from the base.

We now show that every face in the output subdivision G has $O(1)$ dilation. First consider case 2. Note that face P (defined in the algorithm) is a mountain polygon, and after $\text{AddPockets}(G, P)$, the pocketed mountain \widehat{P} is discarded from further consideration. The length of P is at most $(\alpha + \frac{3}{2})|uv|$. By construction, we have $\delta_H(P) = \alpha + \frac{1}{2}$. The length of every maximal vertical segment w_1w_2 , with $w_1, w_2 \in P$ and $\text{relint}(w_1w_2) \subset \text{int}(P)$ is at least $\frac{1}{4}|uv|$. Hence, $\delta_V(P) \leq (\alpha + \frac{1}{4} + \frac{1}{4} + 1)|uv| / (2 \cdot \frac{1}{4}|uv|) = 2\alpha + 3$. By Lemma 4 we have $\delta_{\text{int}}(P) \leq 2\alpha + 3$ and, by Lemma 2, $\delta_{\text{int}}(\widehat{P}) \leq 6\alpha + 9$.

Next, consider case 3. Again, P (defined in the algorithm) is a mountain polygon, and after $\text{AddPockets}(G, P)$, the pocketed mountain \widehat{P} is discarded from further consideration. The total length of the two vertical segments connecting u and v to the base is at most $\frac{1}{2}|uv|$. So the length of P is at most $(\alpha + \frac{3}{2})|uv|$. By construction, we have $\delta_H(P) < \alpha + \frac{1}{2}$. We next give an upper bound on $\delta_V(P)$. If uv does not contain any horizontal edge of M , then the length of $\lambda(P)$ is at least $|uv|/4$, which is the height of the padding of uv . If uv contains a horizontal edge of M , then the length of $\lambda(P)$ is at least $|uv|/12$. Hence, $\delta_V(P) < (\alpha + \frac{3}{2})|uv| / (2 \cdot \frac{1}{12}|uv|) = 6\alpha + 9$. Lemma 4 gives $\delta_{\text{int}}(P) \leq 6\alpha + 9$ and by Lemma 2, we have $\delta_{\text{int}}(\widehat{P}) \leq 18\alpha + 27$. Thus in both cases, we have $\delta_{\text{int}}(\widehat{P}) \leq 18\alpha + 27$.

Recursive calls and active polygons. One recursive call of Algorithm 3 processes a mountain M until one of the cases 2 or 3 occurs. In both cases, it subdivides M into polygonal faces, discards a face \widehat{P} which

is a pocketed mountain polygon of internal dilation at most $18\alpha + 27$, and recurses on the remaining faces. The faces processed on level i of the recursion are the *active polygons* of level i , denoted by \mathcal{F}_i . Note that each active mountain polygon is internally 4-spacious, and the active polygons are pairwise disjoint.

Charging scheme for length. Consider one recursive call where an active mountain $M \in \mathcal{F}_i$ is processed. In case 2, at least an $\alpha|uv|$ portion of M is discarded, and new edges of total length at most $\frac{3}{2}|uv|$ are created and passed over to the subproblems. In case 3, at least a $2|uv|$ portion of M is discarded, and some new edges of length at most $\frac{1}{2}|uv|$ are created. So each unit of length is charged by $\max(\frac{1}{4}, \frac{3}{2\alpha})$ new units. It follows from a summation of the geometric series (that is, $\sum_{i=0}^{\infty} x^i = \frac{1}{1-x}$ for $0 < x < 1$) that the total length of the new edges in the entire algorithm is at most

$$\left(\frac{1}{1 - \max(\frac{1}{4}, \frac{3}{2\alpha})} - 1 \right) |M|.$$

With $\alpha \geq 6$ (note that we will use $\alpha = 9$), the total length of the new edges is at most $|M|/3$, and the output subdivision has length at most $\frac{4}{3}|M|$.

Charging scheme for vertices. We classify the set of vertices of the active polygons in \mathcal{F}_i : Let V_i denote the set that contains, for each active polygon $M \in \mathcal{F}_i$, all vertices except for the four vertices on the leftmost and rightmost vertical edges of M . Let $m_i = \#\mathcal{F}_i$ denote the number of active polygons at level i of the recursion. Initially, we have $\#V_0 = n - 4$ and $m_0 = 1$. When an active polygon $M \in \mathcal{F}_i$ is processed, some of its vertices in V_i are discarded, and some become vertices of V_{i+1} in the next level of recursion. We now show that $\#V_{i+1} - m_{i+1} < \#V_i - m_i$ at every level i .

Consider the effect of one recursive call on V_i and m_i , ignoring for a moment the effect of the calls to the subroutine `AddPockets`. Let $M \in \mathcal{F}_i$ be an active polygon. If case 2 applies, at least two vertices are discarded (the vertices directly above u and v), and two new vertices are created (below u and v). If u (resp., v) does not lie along a leftmost or rightmost vertical edge of M , then the discarded vertex above u (resp., v) is in V_i and the new vertex below is in V_{i+1} ; otherwise these vertices above and below u (resp., v) are neither in V_i nor in V_{i+1} . If $\alpha \geq 9$, then case 2 discards at least two additional vertices of V_i , because otherwise the vertical edges of M above u and v would form a 4-narrow channel in $\text{dom}(M)$. Set $\alpha = 9$. Then in case 2, the number of vertices discarded from V_i is larger by at least 2 than the number of newly created vertices in V_{i+1} , while the number of active polygons remains the same.

If case 3 applies to $M \in \mathcal{F}_i$, no new vertices of V_{i+1} are created and at least two vertices of V_i are discarded; the number of components increases by at most one. Finally, each pocket created by the subroutine `AddPockets` creates one more active polygon, but it discards the same number of vertices of V_i as it creates in V_{i+1} . Hence, $\#V_{i+1} - m_{i+1} < \#V_i - m_i$.

Initially, we have $\#V_0 - m_0 = n - 5$. It follows that cases 2 and 3 occur at most $n - 5$ times during Algorithm 3.

Let us estimate the number of pockets created by calls to the `AddPockets` subroutine in Algorithm 3. Let E_0 be the set of edges of the input mountain M . In both case 2 and 3, we define a path π consisting of three segments: In case 2, let π consist of the left, right and lower sides of the padding of uv ; in case 3 let π consist of the vertical edges connecting u and v to the projection of uv to the base. Let E_1 be the set of all paths π constructed in this way. Since cases 2 and 3 occur at most $n - 5$ times, we have $\#E_1 \leq n - 5$. Furthermore, the paths in E_1 are pairwise noncrossing. Each pocket corresponds to two parallel edges, one in E_1 and another one in $E_0 \cup E_1$; and each pair in $E_1 \times (E_0 \cup E_1)$ is responsible for at most one pocket. Drawing a curve in each pocket that connects the two corresponding edges of E_0 and E_1 , we obtain a plane graph with vertex set $E_0 \cup E_1$ (each edge and path is contractible to a point). A plane graph with

$\#(E_0 \cup E_1) \leq 2n - 5$ vertices has at most $3(2n - 5) - 6 = 6n - 21$ edges by Euler's polyhedron theorem. Hence the number of these pockets is less than $6n$.

Each pocket comes with four new vertices. Apart from the new vertices of pockets, either of case 2 and 3 creates two new vertices. Throughout Algorithm 3, a total of at most $4 \cdot 6n + 2(n - 5) < 26n$ new vertices are created. Hence, the output subdivision has at most $27n$ vertices.

An $O(n \log n)$ time implementation. The recursive subdivision of a 4-spacious mountain polygon can be implemented in a single sweep line algorithm in $O(n \log n)$ time. As the horizontal line sweeps over the input mountain M from its top horizontal edge to its base, we maintain the list of vertical edges of all active mountains sorted by x coordinates. For each horizontal segment uv with $u, v \in M$ and $\text{relint}(uv) \subset \text{int}(M)$ we also maintain $|uv|$, $d_M^*(u, v)$, and the padding of uv . We maintain an event queue of the future positions of the sweep line ℓ where cases 2 and 3 occur, and where ℓ contains a horizontal edge of M .

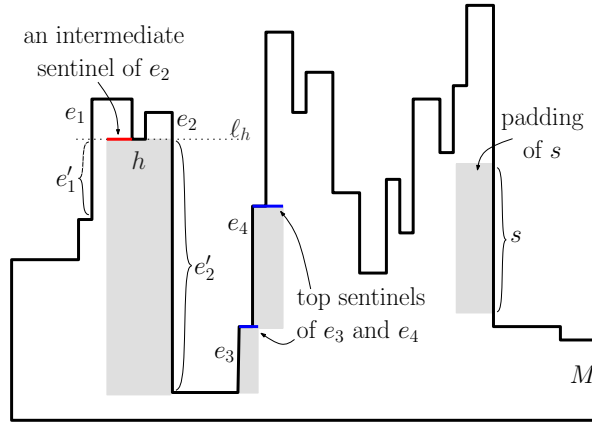


Figure 9: A padding of a vertical segment $s \subset M$ and the top and intermediate sentinels in a mountain polygon M . If a new vertical edge traverses the padding of e'_2 , e_3 , or e_4 , then it forms a 4-narrow channel with it.

It is easy to detect 4-narrow channels between the lower edge of a padding and the base. In order to detect 4-narrow channels between vertical edges efficiently, we maintain a set of horizontal line segments, which we call *sentinels*; see Fig. 9. Intuitively, the sentinels of an existing vertical edge e signal if a new vertical edge f is so close to e that the two edges form a 4-narrow channel. Recall that the new vertical edges created in Algorithm 3 are always incident to a reflex vertex along the upper chain of an active mountain.

For every vertical edge e of the active mountains, we define one *top sentinel* and possibly some *intermediate sentinels*. For every vertical segment $s \subset M$ along a simple vertical mountain M , let the *padding* of s be a rectangle whose left or right side is s , that extends into the interior of M and whose horizontal sides have length $|s|/4$. (Refer to Fig. 9.) The *top sentinel* of a vertical edge e is the top side of the padding of e . Since the visibility between the top vertices of vertical edges forming a 4-narrow channel may be blocked by the upper polygonal chain of M , we construct up to two intermediate sentinels along the line through horizontal edges whose both endpoints are incident to 270° interior angles of M . From every such horizontal edge h , shoot two horizontal rays in opposite directions along the line ℓ_h through h . The two rays hit some vertical edges e_1 and e_2 . Let e'_1 and e'_2 denote the portions of these edges below ℓ_h . The *intermediate sentinel* of e_1 along ℓ_h is a possible segment along the top side of the padding of e'_1 : it is the portion of the top side of the padding of e'_1 which is blocked by h from e_1 . Similarly, the intermediate sentinel of e_2 along ℓ_h is the possible portion of the top side of the padding of e'_2 which is blocked by h from e_2 . Each vertical edge has one top sentinel, and there are up to two intermediate sentinels along the line through each horizontal edge. Overall we maintain a total of $O(n)$ sentinels.

If a new vertical edge e_2 forms a 4-narrow channel with an existing edge e_1 , then e_2 must intersect one of the sentinels of e_1 . If e_2 intersects a sentinel of e_1 , but does not form a 4-narrow channel with it, then the bottom vertex of e_2 is above the bottom vertex of e_1 . Hence, e_2 can intersect at most two sentinel segments (of two vertical edges on opposite sides of e_2) without forming a 4-narrow channel with the corresponding edges. We can compute all intersections of e_2 with the (horizontal) sentinel segments using the dynamic ray shooting data structure of Giyora and Kaplan [19] in $O(\log n)$ time per intersection. While the collection of vertical edges of active polygons may change (edges may be removed, shortened, and new edges may be created), the ray shooting data structure can be updated in $O(\log n)$ time per change. Since a total of $O(n)$ vertices are created throughout the algorithm, there are $O(n)$ updates. Thus the total runtime of our subdivision algorithm is $O(n \log n)$. This completes the proof of Lemma 5. \square

4.2 Subdividing spacious mountains into polygons of constant geometric dilation

Next we show how to perturb a 4-spacious mountain into a *simple* internally 4-spacious mountain, for which Algorithm 3 can be applied.

Lemma 6 *A 4-spacious mountain M with n vertices has an orthogonal 4-spacious subdivision G , where (i) the internal dilation of every face of G is at most 378; (ii) the length of G is at most $\frac{5}{3}|M|$; and G has at most $54n$ vertices. Such a subdivision can be computed in $O(n \log n)$ time.*

Proof: For a 4-spacious mountain polygon M with n vertices, we construct a perturbed polygon M' , which is internally 4-spacious, simple, and has up to $2n$ vertices and $|M'| \leq 2|M| + 2n\epsilon$ length for an arbitrarily small ϵ . Let M' be the locus of all points in $\text{int}(M)$ at L_1 -distance ϵ from M for a sufficiently small $\epsilon > 0$ (Fig. 10(a)). Since M' is the boundary of a simply connected polygonal domain (the points in $\text{int}(M)$ at L_1 -distance at least ϵ from M), M' is a simple polygon. We insert a vertex at every bend point of the closed curve M' , and at every connected straight line portion of M' in an $(\sqrt{2}\epsilon)$ -neighborhood of a vertex of M . In particular, for a vertex $v = (v_1, v_2)$ of M : If M has an angle of 90° or 270° at v , then M' has a bend at a point $(v_1 \pm \epsilon, v_2 \pm \epsilon)$; if M has an angle of 180° at v , then we insert a vertex on an edge of M' at $(v_1, v_2 \pm \epsilon)$ or $(v_1 \pm \epsilon, v_2)$; finally if M has an angle of 360° at v then M' has two bends at points $(v_1 \pm \epsilon, v_2 \pm \epsilon)$. Since a mountain polygon can have at most two angles incident to any vertex, which add up to at most 360° , the simple polygon M' has at most $2n$ vertices, at most two vertices in the neighborhood of each vertex of M . If M is 4-spacious, then M' is internally 4-spacious for a sufficiently small ϵ .

By Lemma 5, $\text{SubdivideMountain}(M')$ returns a 4-spacious subdivision G' of M' where (i) the internal dilation of every face is at most 189; (ii) $|G'| \leq \frac{4}{3}|M'|$; and G' has at most $27 \cdot 2n = 54n$ vertices (Fig. 10(b)). In the computation, we use virtual coordinates, with an infinitesimally small ϵ . By making $\epsilon = 0$, each vertex of M' is snapped to a nearby vertex of M (Fig. 10(c)), and we obtain a subdivision G of M , where (i) the internal dilation of every face is at most 189; (ii) $|G| \leq \frac{8}{3}|M|$; and G has at most $54n$ vertices. For a 4-spacious mountain M , denote the resulting subdivision G by $\text{SubdivideMountain}(M)$. \square

4.3 Subdividing spacious pocketed mountains into polygons of constant geometric dilation

Next we extend Lemma 5 to *pocketed* mountains, by amending Algorithm 3, $\text{SubdivideMountain}(M)$. Our new algorithm is $\text{SubdividePMountain}(\widehat{M})$.

Lemma 7 *Every 4-spacious pocketed mountain \widehat{M} with n vertices admits an orthogonal subdivision G , where (i) the internal dilation of every face of G is at most 1134; (ii) the length of G is at most $2|\widehat{M}|$; and (iii) G has at most $324n$ vertices. Such a subdivision can be computed in $O(n \log n)$ time.*

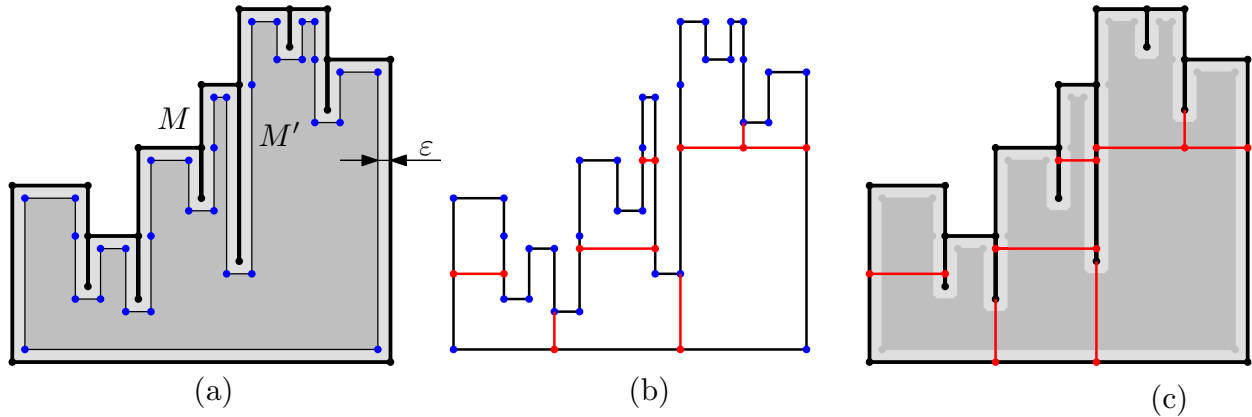


Figure 10: (a) A weakly simple mountain M with 21 vertices and the perturbed simple polygon M' with 33 vertices, (b) a subdivision of M' , (c) the induced subdivision of M .

Intuitive overview. A 4-spacious pocketed mountain \widehat{M} is represented as a 4-spacious mountain M and a finite set of pockets. Algorithm `SubdivideMountain(M)` returns a 4-spacious subdivision H of M into faces of constant geometric dilation. It remains to modify this subdivision to cover the pockets of \widehat{M} . If a pocket r of \widehat{M} has bounded aspect ratio, then we could simply add it as a new face in the subdivision of \widehat{M} . If r has large aspect ratio, however, then we would like to attach it to the adjacent faces of the subdivision H using our subroutine `AddPockets`. If a remaining portion r' of r still has large aspect ratio, then intuitively it must be adjacent to several faces of H . In this case, we simply partition r' into rectangles of aspect ratio at most 4, and charge the new Steiner vertices to the vertices of H on the boundary of r' . We continue with the details.

Algorithm 4 `SubdividePMountain(\widehat{M})`.

Input: a 4-spacious pocketed mountain \widehat{M} represented as a 4-spacious mountain M and a finite set of pockets.

Output: a subdivision G of the input \widehat{M} . All bounded faces of G have internal dilation at most 1134.

1. Let $H = \text{SubdivideMountain}(M)$ and $G = H \cup \widehat{M}$. Let \mathcal{P} denote the faces of H .
2. For every $f \in \mathcal{P}$, apply `AddPockets(G, f)`.
3. Subdivide every rectangle of $\text{dom}(\widehat{M}) \setminus \bigcup_{f \in \mathcal{P}} \text{dom}(f)$ into rectangles of aspect ratio at most 4.
4. Return G .

Proof of Lemma 7: Recall that each pocket r of \widehat{M} has a common side s_r with M , and the length of each side of r orthogonal to s_r is at most $|s_r|$. Note also that $|M| \leq |\widehat{M}|$ and that M has fewer vertices than \widehat{M} .

By Lemma 6, G is an orthogonal subdivision of M where (i) the internal dilation of every face $f \in \mathcal{P}$ is at most 378; (ii) $|G| \leq \frac{5}{3}|M|$; and G has at most $54n$ vertices. In step 2, the internal dilation of every face $f \in \mathcal{P}$ increases at most by a factor of 3 to at most $3 \cdot 378 = 1134$ by Lemma 2; and the length of G does not increase. At most one pocket is attached to each edge of G lying on the boundary of M , each of which creates four new vertices. So the number of vertices increases by a factor of at most 5 to at most $5 \cdot 54n = 270n$.

The regions $\text{dom}(f)$, $f \in \mathcal{P}$, tile the domain of the mountains polygon $\text{dom}(M)$ entirely. After adding pockets to each face in \mathcal{P} , however, the regions $\text{dom}(f)$, $f \in \mathcal{P}$, may still not fill the domain of the pocketed mountain $\text{dom}(\widehat{M})$. The difference set $\text{dom}(\widehat{M}) \setminus \bigcup_{f \in \mathcal{P}} \text{dom}(f)$ consists of maximal connected portions

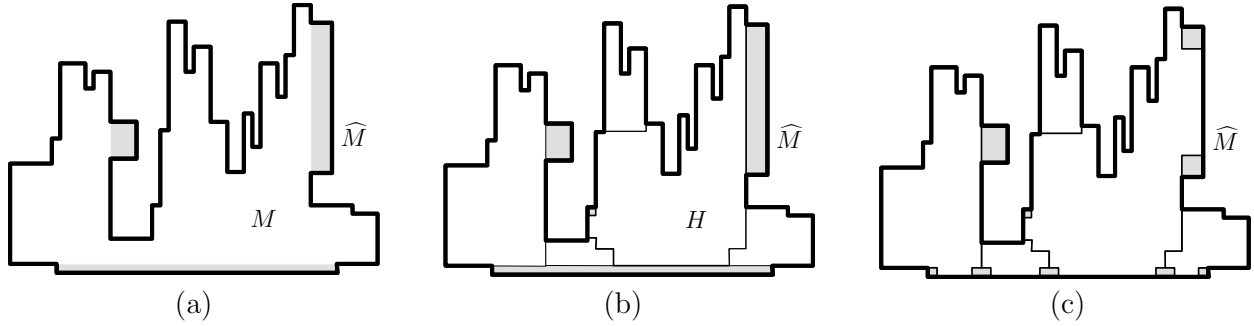


Figure 11: (a) A 4-spacious pocketed mountain \widehat{M} represented as a 4-spacious mountain M and three pockets. (b) `SubdivideMountain(M)` produces a 4-spacious subdivision H of M . (c) Subroutines `AddPockets` extend some of the faces of H into the adjacent pockets of \widehat{M} .

of pockets of \widehat{M} that have not been attached to any face of H . Each such portion is a rectangle $r' \subset r$ for a pocket r of \widehat{M} . If the aspect ratio of r' is $t \geq 4$, then G must have at least $\lfloor t/4 \rfloor$ vertices along s_r , otherwise it would form a 4-narrow channel with H , which are eliminated in step 2. Step 3 creates at most one new vertex along \widehat{M} for each vertex of H along M , and the number of vertices of H is bounded by $54n$. The length of the subdivision increases by at most $|\widehat{M}|/4$. We obtain an orthogonal subdivision \widehat{M} where (i) the internal dilation of every face is at most $3 \cdot 378 = 1134$; (ii) the total length is at most $\frac{5}{3}|M| + \frac{1}{4}|\widehat{M}| \leq \frac{23}{12}|\widehat{M}| < 2|\widehat{M}|$; and (iii) the total number of vertices is at most $(270 + 54)n = 324n$. It is straightforward to implement Algorithm 4 in $O(n \log n)$ time. \square

5 Conclusion

We have shown that any set of n points in the plane can be embedded in a planar straight line graph having $O(1)$ geometric dilation, $O(n)$ vertices, and $O(W)$ length. In addition, we also guaranteed that all edges are axis-parallel (i.e., the network is orthogonal), thus the maximum degree is 4. An obvious open problem is to improve the constants to a level close to that of low vertex dilation networks. We do not know, either, whether Steiner points are necessary (if we drop the condition that edges are axis-parallel).

One could ask whether it is possible to construct a plane network with the additional property that all bounded faces are convex (e.g., all bounded faces are rectangles in case of an orthogonal network). We have recently shown that the answer is negative: Specifically, a set S of n points uniformly distributed on two concentric circles of radii 1 and 2 has the property that every plane network with $O(n)$ vertices, $O(W)$ length, and $O(1)$ vertex dilation must have a nonconvex bounded face [14].

We now discuss a possible extension of our main result in Theorem 1. Clearly any PSLG G in which the minimum angle is α has geometric dilation at least $1/\sin(\frac{\alpha}{2})$ [15] (e.g., $\alpha = \pi/2$ for orthogonal networks). Moreover, this value cannot be reduced by *augmenting* G with new edges, since the minimum angle can only decrease. This leads us to the following natural question. Is it always possible to augment G (with new edges and Steiner vertices) so that the geometric dilation of the resulting graph G' is at most $O(1/\sin(\frac{\alpha}{2})) = O(1/\alpha)$? In addition, can the total length of the edges be small compared to $|G|$? We believe the answers to both questions are positive:

Conjecture 1 *Let G be a connected planar straight line graph with minimum angle α . Then one can augment G (if necessary) with edges of total length $O(|G|)$ into a planar straight line graph G' of geometric dilation $O(1/\alpha)$.*

This would mean that any network without sharp angles admits a cheap “fix” so that the geometric dilation of the resulting network is close to optimal. Observe, however, that there is no possible upper bound guarantee on the number of edges used: See for instance, again, the case of a thin rectangle.

Acknowledgments. We thank Ansgar Grüne and Minghui Jiang for interesting discussions on the topic. We are also grateful to the anonymous referees for many useful comments and suggestions.

References

- [1] P. K. Agarwal, R. Klein, C. Knauer, S. Langerman, P. Morin, M. Sharir, and M. Soss, Computing the detour and spanning ratio of paths, trees, and cycles in 2D and 3D, *Discrete Comput. Geom.* **39** (1-3) (2008), 17–37.
- [2] I. Althöfer, G. Das, D. P. Dobkin, D. Joseph, and J. Soares, On sparse spanners of weighted graphs, *Discrete Comput. Geom.* **9** (1993), 81–100.
- [3] B. Aronov, M. de Berg, O. Cheong, J. Gudmundsson, H. J. Haverkort, and A. Vigneron, Sparse geometric graphs with small dilation, *Comput. Geom. Theory Appl.* **40** (3) (2008), 207–219.
- [4] S. Arora, Polynomial-time approximation schemes for Euclidean TSP and other geometric problems, *J. of ACM* **45** (1998), 753–782.
- [5] M. Benkert, A. Wolff, F. Widmann, and T. Shirabe, The minimum Manhattan network problem: approximations and exact solutions, *Comput. Geom. Theory Appl.* **35** (2006), 188–208.
- [6] J. L. Bentley, Multidimensional divide and conquer, *Comm. ACM* **23** (1980), 214–229.
- [7] M. W. Bern, Two probabilistic results on rectilinear Steiner trees, *Algorithmica* **3** (1988), 191–204.
- [8] P. Bose, J. Gudmundsson, and M. Smid, Constructing plane spanners of bounded degree and low weight, *Algorithmica* **42** (2005), 249–264.
- [9] B. Chandra, G. Das, G. Narasimhan, J. Soares, New sparseness results on graph spanners, *Int. J. Comput. Geometry Appl.* **5** (1995), 125–144.
- [10] L. P. Chew, There are planar graphs almost as good as the complete graph, *J. Computer Sys. Sci.* **39** (1989), 205–219.
- [11] G. Das and D. Joseph, Which triangulations approximate the complete graph? in *Proc. Int. Sympos. on Optimal Algorithms*, vol 401 of LNCS, Springer, 1989, pp. 168–192.
- [12] D. P. Dobkin, S. J. Friedman, and K. J. Supowit, Delaunay graphs are almost as good as complete graphs, *Discrete Comput. Geom.* **5** (1990), 399–407.
- [13] A. Dumitrescu, A. Ebbels-Baumann, A. Grüne, R. Klein, and G. Rote, On the geometric dilation of closed curves, graphs, and point sets, *Comput. Geom. Theory Appl.* **36** (2006), 16–38.
- [14] A. Dumitrescu and Cs. D. Tóth, Minimum weight convex Steiner partitions, *Proc. 19th ACM-SIAM Symposium on Discrete Algorithms*, ACM Press, 2008, pp. 581–590.
- [15] A. Ebbels-Baumann, A. Grüne, and R. Klein, On the geometric dilation of finite point sets, *Algorithmica* **44** (2006), 137–149.

- [16] A. Ebberts-Baumann, R. Klein, E. Langetepe, A. Lingas, A fast algorithm for approximating the detour of a polygonal chain, *Comput. Geom. Theory Appl.* **27** (2004), 123–134.
- [17] D. Eppstein, Spanning trees and spanners, in *Handbook of Computational Geometry* (J. R. Sack and J. Urrutia, eds), North-Holland, Amsterdam, 2000, pp. 425–461.
- [18] M. Garey and D. Johnson, The rectilinear Steiner tree problem is NP-complete, *SIAM J. on Appl. Math.* **32** (4) (1977), 826–834.
- [19] Y. Gijora and H. Kaplan, Optimal dynamic vertical ray shooting in rectilinear planar subdivisions, in *Proc. 18th ACM-SIAM Sympos. on Discrete Algorithms*, ACM Press, 2007, pp. 19–28.
- [20] J. Gudmundsson and C. Knauer, Dilation and detour in geometric networks, chap. 52 in *Handbook on Approximation Algorithms and Metaheuristics* (T. Gonzalez, editor), CRC, 2007.
- [21] J. Gudmundsson, C. Levcopoulos, and G. Narasimhan, Approximating a minimum Manhattan network, *Nordic J. of Computing* **8** (2001), 216–229.
- [22] J. Hershberger and S. Suri, A pedestrian approach to ray shooting: Shoot a ray, take a walk, *J. Algorithms* **18** (1995), 403–431.
- [23] H. Imai and T. Asano, Dynamic orthogonal segment intersection search, *J. Algorithms* **8** (1) (1987), 1–18.
- [24] M. Keil and C. A. Gutwin, Classes of graphs which approximate the complete Euclidean graph, *Discrete Comput. Geom.* **7** (1992), 13–28.
- [25] J. B. Kruskal, On the shortest spanning subtree and the traveling salesman problem, *Proc. American Math. Soc.* **7** (1956), 48–50.
- [26] C. Levcopoulos and A. Lingas, There are planar graphs almost as good as the complete graphs and almost as cheap as minimum spanning trees, *Algorithmica* **8** (1992), 251–256.
- [27] J. MacGregor Smith and P. Winter, Computing in Euclidean geometry, in *Computational Geometry and Topological Network Design*, World Scientific, 1992, pp. 287–385.
- [28] G. Narasimhan and M. Smid, *Geometric Spanner Networks*, Cambridge University Press, 2007.
- [29] Y. Nekrich, Orthogonal range searching in linear and almost-linear space, in *Proc. 10th Workshop on Algorithms and Data Structures*, vol. 4619 of LNCS, Springer, 2007, pp. 15–26.
- [30] Y. C. Wee, S. Chaiken, and S. S. Ravi, Rectilinear Steiner tree heuristics and minimum spanning tree algorithms using geographic nearest neighbors, *Algorithmica* **12** (1994), 421–435.
- [31] D. E. Willard, New data structures for orthogonal range queries, *SIAM J. Comput.* **14** (1985), 233–253.