

Pointed Binary Encompassing Trees

Michael Hoffmann¹, Bettina Speckmann², and Csaba D. Tóth³

¹ Institute of Theoretical Computer Science, ETH Zürich
hoffmann@inf.ethz.ch

² Department of Mathematics and Computer Science, TU Eindhoven
speckman@win.tue.nl

³ Department of Computer Science, University of California at Santa Barbara
toth@cs.ucsb.edu

Abstract. We show that for any set of disjoint line segments in the plane there exists a *pointed binary encompassing tree* T , that is, a spanning tree on the segment endpoints that contains all input segments, has maximum degree three, and every vertex $v \in T$ is pointed, that is, v has an incident angle greater than π . Such a tree can be completed to a minimum pseudo-triangulation. In particular, it follows that every set of disjoint line segments has a minimum pseudo-triangulation of bounded vertex degree.

1 Introduction

Disjoint line segments in the plane are the fundamentals of computational geometry. They form the atomic structure of most planar geometric data structures and geographic information systems. Planar objects are typically represented by a polygonal approximation which, in turn, is composed of (interior) disjoint line segments. Not surprisingly, researchers studied many of their combinatorial properties, such as visibility, compact representation, and ray shooting.

Geometric graphs. We follow one particularly well-studied trail: that of constrained geometric graphs. A *geometric graph* is a graph together with a planar embedding such that the edges are straight line segments. We consider *crossing-free* geometric graphs, that is, we do not allow two edges to cross. Given a set of disjoint segments in the plane (that is, a crossing-free geometric matching), we say that a graph is *encompassing* if it is a connected crossing-free geometric graph that contains all input segments as edges (without Steiner points).

It is known that there does not always exist a Hamiltonian encompassing circuit (nor path) [21]. In fact, it is NP-complete to decide if a Hamiltonian encompassing circuit exists for a given set of segments, if the segments are allowed to intersect at their endpoints [17]. Among n disjoint segments in the plane there are always $\Theta(\log n)$ for which an encompassing path exists [11], this number amounts to $\Theta(\sqrt{n})$ if all segments are axis-parallel [20].

The maximum degree of an encompassing tree on the segment endpoints that is *constrained* to contain all input segments is, therefore, at least three.

After a preliminary upper bound of seven by Bose and Toussaint [6], Bose et al. [5] proved that an encompassing tree with maximum degree three always exists. Later Hoffmann and Tóth [12] showed that there is also a *Hamiltonian* encompassing graph with maximum degree three.

Pseudo-triangulations. Recently a relaxation of triangulations, called *pseudo-triangulations*, has received considerable attention. Here, faces are bounded by three concave chains, rather than by three line segments. More formally, a pseudo-triangle is a planar polygon that has exactly three convex vertices with internal angles less than π . Pseudo-triangulations were originally studied for convex sets and for simple polygons because of their applications to visibility [15,16] and ray shooting [7,10]. But in the last few years they also found application in robot motion planning [19], kinetic collision detection [1,14], and guarding [18].

Of particular interest are the so-called *minimum pseudo-triangulations*, which have the minimum number of pseudo-triangular faces among all possible pseudo-triangulations of a given domain. They were introduced by Streinu [19], who proved that every minimum pseudo-triangulation of a set S of n points consists of exactly $n - 2$ pseudo-triangles. Minimum pseudo-triangulations are also referred to as *pointed pseudo-triangulations* since every vertex v of a minimum pseudo-triangulation has an incident region whose angle at v is greater than π .

Pseudo-triangulations, just like triangulations, are also crossing-free geometric graphs. But while triangulations of a planar point set can have arbitrarily high vertex degree, there is always a pseudo-triangulation of vertex degree at most five [13]. Bounded vertex degree is a useful property for many applications, since it enables local operations or updates in constant time.

Streinu [19] showed that every pointed geometric graph can be completed to a pointed pseudo-triangulation by greedily adding edges while maintaining pointedness. But this approach does not provide any guarantee regarding the vertex degree. On the other hand, pointed spanning trees are not omnipresent in planar structures: Aichholzer et al. [3] just established that there are triangulations which do not contain any pointed spanning tree. Furthermore, both the algorithm of Bose et al. [5] and that of Hoffmann and Tóth [12] violate pointedness due to their proof techniques.

Results. Here, we show how to construct an encompassing tree that respects pointedness *and* has maximum vertex degree at most three:

Theorem 1. *For any finite set of disjoint line segments in the plane there exists a pointed binary encompassing tree.*

Our proof is constructive: we describe a recursive algorithm that builds a binary pointed encompassing tree for n disjoint segments in $O(n^{4/3}\text{polylog } n)$ time.

Aichholzer et al. [2] showed that a bounded degree pseudo-triangulation constrained to contain a Hamiltonian circuit (a simple polygon) always exists, with a degree bound of seven. With the help of a pointed binary encompassing tree we can extend these results to pseudo-triangulations constrained to contain disjoint line segments:

Theorem 2. *For any finite set of disjoint line segments there is a pointed encompassing pseudo-triangulation with maximum vertex degree at most ten.*

Organization. The next section presents the definition of a special class of polygons that we call *necklaces*. Section 3 provides an algorithmic overview and states the extensive set of invariants which we maintain during our construction. Section 4 gives the actual algorithm that constructs a necklace from the set of input segments. In Section 5, we prove Theorem 1 and sketch the runtime analysis of our algorithm. Finally, Section 6 shows how to combine the encompassing tree with the algorithm described in [2] to construct a pointed encompassing pseudo-triangulation with a maximum vertex degree of ten.

2 Definitions and Basic Operations

A *polygon* P is a sequence (p_1, p_2, \dots, p_k) of points in the plane. Denote the set of vertices of P by $V(P) = \{p_1, p_2, \dots, p_k\}$, and the set of edges by $E(P) = \{p_1p_2, p_2p_3, \dots, p_{k-1}p_k, p_kp_1\}$. Let ∂P denote the closed path $p_1p_2 \cup p_2p_3 \cup \dots \cup p_kp_1$, and let $\deg_P(p)$ be the number of edges incident to a point $p \in V(P)$. A polygon is *weakly simple* if (i) any two edges are either disjoint or intersect in one endpoint, (ii) $\sum_{i=1}^k \angle p_{i+1}p_i p_{i-1} = (k - 2)\pi$, and (iii) all edges incident to p_i are in the closed angular domain $\angle p_{i+1}p_i p_{i-1}$, for $i = 1, 2, \dots, k$. For a weakly simple polygon P , we define \bar{P} as the closed polygonal domain enclosed by ∂P . We denote the interior of the polygonal domain \bar{P} by $\text{int}(P)$. A polygon P is *simple* if ∂P is a simple closed curve. A vertex p_i of a polygon is *convex (reflex)* if $\angle p_{i+1}p_i p_{i-1}$ is convex (reflex). A polygon is *convex* if all of its vertices are convex. Finally, an orientation $u(P)$ of the vertices of a polygon P is a function $u : P \rightarrow \{-1, +1\}$. In analogy to the notation for polygons, for a line segment s we use \bar{s} to refer to it as a set of points in the plane. Similarly, for a set S of line segments let $\bar{S} := \{\bar{s} \mid s \in S\}$.

Definition 1. *For a set S of segments in the plane, a necklace P is a weakly simple polygon such that*

- every vertex is an endpoint of a segment in S ;
- at every vertex, the incident edges and input segments are pointed;
- the degree (w.r.t. P) of every vertex is two or four;
- all segments of S are contained in \bar{P} .

An edge $e \in E(P)$ is called *segment edge*, if $e \in S$, and *visibility edge*, otherwise. A segment $pq \in S$ is *saturated* with respect to a necklace P , iff $\{p, q\} \subset V(P)$. A vertex $p \in V(P)$ is saturated, iff the incident segment from S is saturated.

The graph of a weakly simple polygon is a *tree of rings*, which is a union of rings such that any two rings have at most one common vertex, and every cycle in the graph is one of the rings. If we represent every ring by a node and connect two nodes when the corresponding rings have a common vertex, then we obtain

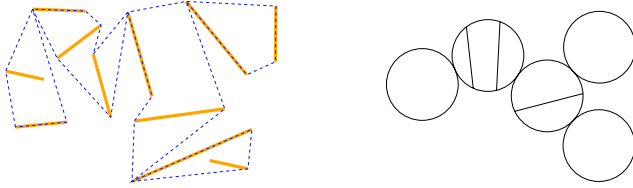


Fig. 1. A necklace and its structure: a tree of rings with diagonals.

a tree. Assuming that a necklace polygon has ℓ vertices of degree four, its graph is composed of $\ell + 1$ rings.

The segments from S which form internal diagonals of P , also referred to as *segment diagonals*, partition the rings into interior-disjoint sub-rings, any two of which are either disjoint or share a common segment edge. The tree structure of the rings (and sub-rings) allows to delete one edge from every sub-ring while maintaining connectivity. For this we color the edges of the necklace red or black in such a way that every sub-ring contains exactly one red visibility edge and every degree four vertex is incident to a red edge. Deleting all red edges from the necklace then yields a binary encompassing tree.

2.1 Basic Operations

The two basic operations used in our algorithms are based on geodesic curves. Roughly speaking, a geodesic curve between two points is a shortest curve from a specific class of curves that connect the points.

Definition 2. Consider two distinct points p and q , and a finite set S of line segments in the plane. Let $\Gamma(p, q)$ be the set of all simple polygonal paths between p and q that do not cross any segment from S . For any $\varrho \in \Gamma(p, q)$ denote by $\text{geo}(\varrho)$ the shortest curve from p to q that is homotopic to ϱ within $\Gamma(p, q)$. We say $\text{geo}(\varrho)$ is a geodesic curve between p and q with respect to S .

Operation 1: Build_cap (P, u, i) . See Fig. 2 for an example.

Input: A necklace $P = (p_1, p_2, \dots, p_k)$, an orientation $u(P)$, and an unsaturated convex vertex $p_i \in V(P)$. **Operation:** Let qp_i be the input segment incident to p_i . Replace the edge $p_i p_{i+u(p_i)}$ by the path $p_i q \cup \text{geo}(q, p_i, p_{i+u(p_i)})$. Set $u(p) := u(p_i)$ for every interior vertex of $\text{geo}(q, p_i, p_{i+u(p_i)})$ including q .

Operation 2: Extend_reflex (P, u, i, \vec{r}_i) . See Fig. 3 for an example.

Input: A necklace $P = (p_1, p_2, \dots, p_k)$, an orientation $u(P)$, a reflex vertex p_i of P , and a ray \vec{r}_i emanating from p_i such that \vec{r}_i cuts $\angle p_{i+1} p_i p_{i-1}$ into two convex angles and hits a segment $ef \in S$ with $\overline{ef} \subset \text{int}(P)$ at $g \in \overline{ef}$.

Operation: Without loss of generality, suppose that $p_{i+u(p_i)}$ and f are on the same side of the supporting line of \vec{r}_i . Replace the edge $p_i p_{i+u(p_i)}$ of P by the path $\text{geo}(p_i, g, e) \cup (e, f) \cup \text{geo}(f, g, p_i, p_{i+u(p_i)})$. Set $u(\cdot) := -u(p_i)$ for every

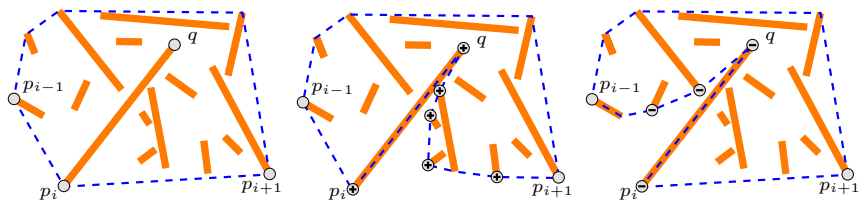


Fig. 2. The result of $\text{Build_cap}(P, u, i)$ for $u(p_i) = +1$ and $u(p_i) = -1$.

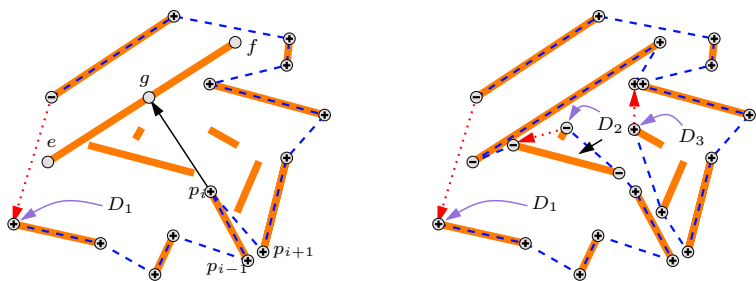


Fig. 3. The result of $\text{Extend_reflex}(P, u, i, \vec{r}_i)$.

interior vertex of $\text{geo}(p_i, g, e)$ including e , and $u(\cdot) := u(p_i)$ for every interior vertex of $\text{geo}(f, g, p_i, p_{i+u(p_i)})$ including f .

Note that whenever an operation produces a new reflex vertex p_a , the orientation is chosen so that the edge $p_a p_{a+u(p_a)}$ is a visibility edge of P . Then Extend_reflex replaces the visibility edge by an open polygonal chain. This guarantees that all segments remain in \bar{P} (as opposed to the operations in [12] where segments of S could become outer diagonals). It is not difficult to see that the two basic operations are necklace preserving, we omit the formal proof here. By an iterative application of Operation 1, we can make sure that every segment is either saturated or lies in the interior of the necklace.

Both_endpoints (P, u)

Input: A necklace $P = (p_1, p_2, \dots, p_k)$ and an orientation $u(P)$. **Operation:** Apply $\text{Build_cap}(P, u, i)$, as long as there is an unsaturated vertex $p_i \in V(P)$.

3 Algorithmic Overview

Starting from the convex hull of the segments, our algorithm greedily constructs a necklace that incorporates input segments as either edges or internal diagonals. When the greedy algorithm terminates it might happen that not all segments are included in the necklace. Therefore we maintain an extensive set of invariants—which are collected in Lemma 1—that allow us to proceed by induction. By

maintaining a necklace P , we ensure that all remaining segments lie in the interior of P . We color the edges of P red or black, so that we can delete red edges later to reduce the maximum degree to three. As a next step we generate a convex partition \mathcal{D} of \bar{P} . We can then apply induction in the interior of every convex piece. Finally we assign a vertex of P to every face $D \in \mathcal{D}$. At this vertex we connect the inductively computed encompassing tree for D to the necklace, while maintaining pointedness and low degree.

The proof of the following key lemma, Lemma 1, based on an algorithm constructing a necklace polygon, is the subject of the next section.

Lemma 1. *For a set S of disjoint line segments, not all in a line, and a vertex x of $\text{conv}(\bar{S})$, there is a necklace P , a partition \mathcal{D} of \bar{P} , an assignment $t : \mathcal{D} \rightarrow V(P) \setminus \{x\}$, and an edge coloring $\gamma : E(P) \setminus S \rightarrow \{\text{red}, \text{black}\}$ satisfying the following properties.*

- (L1) x is vertex of P such that $\deg_P(x) = 2$ and x is incident to a red edge;
- (L2) every minimal cycle in $E(P) \cup S$ contains exactly one red edge;
- (L3) for every $D \in \mathcal{D}$, the point $t(D)$ is a vertex of D ;
- (L4) for every $p \in V(P)$, the number of edges in $E(P) \cup S$ incident to p plus the number of regions of \mathcal{D} assigned to p is less than or equal to 3 plus the number of red edges incident to p ;
- (L5) every $s \in S$ is either saturated, or there is a $D \in \mathcal{D}$ such that $\bar{s} \subset \text{int}(\bar{D})$;
- (L6) every polygon $D \in \mathcal{D}$ is convex;
- (L7) for every $D \in \mathcal{D}$ the edges and input segments incident to $t(D)$ and $\bar{D} \cap \bar{S}$ are on one side of a line through $t(D)$;
- (L8) at most two regions, D_1 and D_2 , are assigned to every point $p \in V \setminus \{x\}$; and in this case $\bar{D}_1 \cup \bar{D}_2$ is a simple polygonal domain with exactly one reflex vertex, that is at p .

4 Constructing a Necklace

We describe our algorithm that constructs a necklace P along with an edge coloring γ , a partition \mathcal{D} , and a vertex assignment $t(\cdot)$ for every region of \mathcal{D} . Properties (L1)–(L4) are maintained all through the algorithm. We apply `Build_cap` and `Extend_reflex` repeatedly to ensure (L5). Finally, further partitioning of the non-convex regions in \mathcal{D} establishes (L6)–(L8).

Both basic operations replace an edge $pq \in E(P)$ by a polygonal chain χ . If pq is red, we need to color the edges of χ carefully in order to maintain (L4) at both p and q . Therefore, for every red edge, we label one endpoint as its *anchor*. Whenever a red edge pq with anchor p is replaced by a polygonal chain χ , we color edges of χ red such that one is anchored at p . For this, we have to make sure, though, that this edge of χ is a visibility edge: In case of `Build_cap`(P, u, i), if $p_i p_{i+u(p_i)}$ is red then its anchor has to be at $p_{i+u(p_i)}$; while for `Extend_cap`(P, u, i, \vec{r}_i), the edges of χ incident to p_i and $p_{i+u(p+i)}$ are both visibility edges. Notice that neither operation replaces a red edge pq if both p and q are saturated and are convex vertices of all their incident regions from \mathcal{D} .

Initialization. Let $P := \text{conv}(\overline{S})$. Label the vertices of P by $x = p_1, p_2, \dots, p_m$ such that $p_1 p_2 \notin S$, without loss of generality in anti-clockwise order. Let $u(x) = -1$ and $u(p) = +1$ for every $p \in V(P) \setminus \{x\}$. The segment diagonals of $\text{conv}(\overline{S})$ partition P into convex polygons. Let these polygons form the initial set \mathcal{D} . For every $D \in \mathcal{D}$ let $t(D)$ be the second vertex in the sequence p_1, p_2, \dots, p_m that is incident to D . Furthermore, color red the visibility edge from $E(D)$ incident to $t(D)$ that leads to the vertex with minimal index, and sets its anchor to $t(D)$. All other edges of $\text{conv}(\overline{S})$ are colored black. See Fig. 4 for an example.

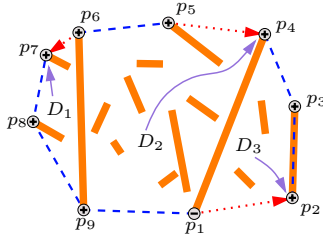


Fig. 4. An initial polygon. Red edges are dotted and point towards their anchor.

The algorithm runs in two phases: First, we apply `Both_endpoints` and `Extend_reflex` alternately, until $(L5)$ is satisfied. The phase is guaranteed to terminate, since both operations increase $|V(P)|$. The second phase keeps the polygon P intact, but subdivides the non-convex regions of \mathcal{D} to ensure $(L6)$.

First phase. Apply `Both_endpoints`(P, u). Then, as long as there is a reflex vertex p_i of P and a ray \vec{r}_i emanating from p_i such that \vec{r}_i partitions the angle $\angle p_{i+1} p_i p_{i-1}$ into two convex angles and \vec{r}_i hits a segment $s \in S, s \subset \text{int}(P)$: apply `Extend_reflex`(P, u, i, \vec{r}_i) followed by `Both_endpoints`(P, u). For a basic operation, modify \mathcal{D} as follows: replace every $D \in \mathcal{D}$ by the polygons D_1, D_2, \dots, D_ℓ that enclose the connected components of $(\overline{D} \cap \text{int}(P)) \setminus \overline{S}$.

Suppose that a basic operation replaces an edge yz by an open polygonal chain $\chi = (y_1 = y, y_2, y_3 \dots, y_k = z)$. Recall that χ is a convex chain for `Build_cap`, and it consists of two convex chains connected by a segment edge in case of `Extend_reflex` χ . Notice that each $D_i, i = 1, 2, \dots, \ell$, has a common edge with χ . Suppose w.l.o.g. that $t(D)$ is a vertex of D_1 , and let y_h be the last vertex of χ that is incident to D_1 . For every polygon $D_i, 1 < i \leq \ell$, let a_i be the closest vertex and let b_i be a second closest vertex of D_i to y_h along χ . Set $t(D_i) := b_i$, color edge $a_i b_i$ red, and set its anchor to a_i . All other edges of χ are colored black. See Fig. 3 for illustration.

Second phase. For every reflex vertex p_i of every non-convex polygon $D \in \mathcal{D}$, let \vec{r}_i be a ray emanating from p_i that partitions the angle $\angle p_{i+1} p_i p_{i-1}$ into two convex angles but does not hit any other vertex of D . By the end condition of the first phase, we know that \vec{r}_i does not hit any segment $s \in S$ before reaching

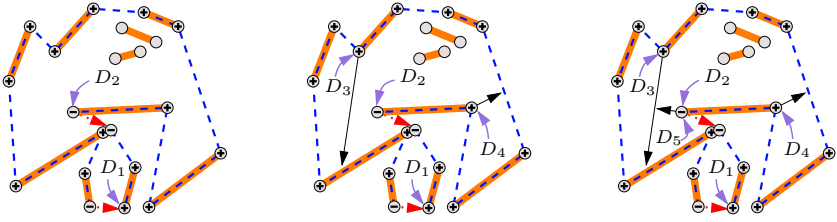


Fig. 5. Partitioning a region D by rays emanating from reflex vertices.

the boundary of D again. Consider first all such rays for which the source vertex is not assigned to any region of \mathcal{D} . After all these rays have been processed, consider the remaining rays sequentially, and split D along \vec{r}_i into two new regions D_1 and D_2 .

When splitting a region $D \in \mathcal{D}$ into two regions D_1 and D_2 along a ray \vec{r}_i , update the assignment t as follows. Assuming, without loss of generality, that $t(D)$ is a vertex of D_1 , let $t(D_1) := t(D)$ and $t(D_2) := p_i$. See Fig. 5.

4.1 Proof of Lemma 1

We have to show that the output of the algorithm described above satisfies all the properties of Lemma 1. Initially, $P = \text{conv}(\bar{S})$ is clearly a necklace polygon. Since we modify the polygon by the basic operations only, P remains a necklace throughout the algorithm.

It is easy to see that $(L1)$ – $(L4)$ hold after initialization. (Coincidentally, $(L6)$ – $(L8)$ also hold.) We need to argue that these four properties are maintained in the algorithm, and that the second phase additionally maintains $(L5)$. In fact, it is enough to check that the updates associated with the basic operations maintain these properties. We point out two benefits of our basic operations.

- (i) At every reflex vertex p of P , the segment $qp \in S$ is an edge of P .
- (ii) No operation replaces the red edge incident to x in the initial polygon.

The first Claim implies that $(L1)$ is maintained: x is always incident to a red edge. $\text{deg}_P(x) = 2$ initially, and since x is a vertex of the convex hull, no operation appends a path that would revisit x . Thus, its degree remains 2.

For $(L2)$, notice that all through the first phase, every region $D \in \mathcal{D}$ is bounded by edges of $E(P) \cup S$, so the minimal cycles corresponds to the polygons of \mathcal{D} . Whenever a polygon D is replaced by the connected components of the region $(\bar{D} \cap \text{int}(P)) \setminus \bar{S}$, one new edge is colored red in each newly created cycle of $E(P) \cup S$. The second phase does not change the structure of cycles.

$(L3)$ is satisfied at the initial polygon. The property is clearly maintained in both phases.

$(L4)$ holds initially: Every vertex of $P = \text{conv}(\bar{S})$ has degree 2, and if a vertex is assigned to a region $D \in \mathcal{D}$, it is assigned to one region only, and it is anchor of a red edge.

During the first phase, every minimal cycle of $E(P) \cup S$ is a polygon of \mathcal{D} . Consider the moment where a vertex p_i is assigned to a new region $D \in \mathcal{D}$: the red edge $p_i q$ of D is placed incident to $t(D) = p_i$, but its anchor is set to q . Vertex q is either endpoint of a segment diagonal or a degree four vertex of P . In any case, q is a convex vertex of both incident regions from \mathcal{D} . Hence, there are only two ways to replace edge $p_i q$: by `Build_cap`(P, u, i), if p_i is convex in D and $p_{i+u(p_i)} = q$, or `Extend_reflex`(P, u, i, \vec{r}_i), if p_i is reflex in D and there is a ray \vec{r}_i from p_i that hits a segment interior to D . In both cases, after such an operation p_i is a convex vertex of P with $\deg_P(p_i) = 2$. Otherwise, the red edge $p_i q$ stays incident to p_i and thus compensates for the assignment $t(D) = p_i$.

Apart from the region assignment, there are vertices of degree two and four in P . If $\deg_P(p) = 2$, then $(L4)$ holds, since p can be incident to at most three edges of $E(P) \cup S$. If $\deg_P(p) = 4$, then p appears as a reflex vertex in P . By Claim (i), the input segment incident to p is in $E(P)$. That is, p is incident to 4 edges in $E(P) \cup S$. In our operations, the degree of p can go up to 4 only if a geodesic curve $\text{geo}(\varrho)$ passes through a reflex vertex at p . In this case, $(\overline{D} \cap \text{int}(P)) \setminus \overline{S}$ has two disjoint components incident to p . The vertex $p \in \text{geo}(\varrho)$ is the closest vertex from one of the regions, say D_i , to the region D_1 containing $t(D)$. Therefore, a visibility edge $p q$ incident to p in D_i is colored red and anchored at p . Moreover, $t(D_i)$ is set to q , that is, our argument here conforms with our reasoning above regarding the region assignments. In particular, if there are two adjacent degree four vertices along a geodesic curve, one of them is incident to two red edges. Altogether we have shown that $(L4)$ holds during the first phase.

In the second phase, $E(P) \cup S$ does not change. When splitting regions of \mathcal{D} , some of the regions are assigned to degree two reflex vertices of P . Since two edges of $E(P) \cup S$ are incident to such a reflex vertex p , $(L4)$ clearly holds for p if at most one region is assigned to p . Assume that after a split $D \rightarrow D_1, D_2$, two regions are assigned to a reflex vertex p . This can only happen if we split along a ray emanating from p and $t(D) = t(D_1) = t(D_2) = p$. We have argued above that p is incident to a red edge in this case, which compensates for the additional region assigned to p . Thus, $(L4)$ holds during and after the second phase as well.

$(L5)$ is end condition of the subroutine `Both.endpoints`. Therefore it is satisfied at the end of the first phase. Obviously the second phase maintains $(L5)$ as well, since the polygon P remains unchanged.

$(L6)$ clearly holds at the end of the algorithm, since the second phase eliminates all reflex vertices of polygons from \mathcal{D} .

$(L7)$ holds trivially for every $t(D)$, $D \in \mathcal{D}$, which is a convex vertex of P . If $t(D) = p_i$ is a reflex vertex of P , consider the inverse wedge $W(p_i)$ of its exterior angle $\angle p_{i-1} p_i p_{i+1}$. By the end condition of the first phase, $W(p_i) \cap \text{int}(\overline{D}) \cap \overline{S} = \emptyset$, which proves $(L7)$.

Finally, we have to consider $(L8)$. Note that this property is void all through the first phase, since every vertex is assigned to at most one region. But during the second phase, we might split a region $D \in \mathcal{D}$ into two regions D_1 and D_2 along a ray r emanating from p , such that afterwards $t(D_1) = t(D_2) = p$. But

this can happen only if $t(D) = p$, that is, for at most one reflex vertex of every region D . Since the algorithm treats these vertices after all other reflex vertices to which no region is assigned to, (L8) holds. This completes the proof of Lemma 1.

5 Constructing a Pointed Binary Encompassing Tree

In this section we use Lemma 1 to give an inductive proof of Theorem 1. For the sake the inductive argument, we actually prove a stronger theorem:

Theorem 3. *For any finite set S of disjoint line segments in the plane and any vertex x of $\text{conv}(\bar{S})$, there exists a pointed binary encompassing tree T such that $\text{deg}_T(x) < 3$.*

Proof. We proceed by induction on $n := |S|$. If all segments are collinear, then there is a Hamiltonian encompassing path along this line. In the general case consider a necklace P as claimed in Lemma 1.

If every input segment is either an edge or a diagonal of P , then the black edges of P together with all segment diagonals form an encompassing tree T : The graph is connected because of (L2), and pointed because P is a necklace. Moreover, the number of black edges and segment diagonals incident to any vertex $p \in P$ in T is at most three by (L4).

Otherwise, there is a segment $s \in S$ that is neither an edge nor a diagonal of P . By (L5), s lies in the interior of a convex polygon $D \in \mathcal{D}$. For every $\mathcal{F} \subseteq \mathcal{D}$, let $S \cap \mathcal{F}$ denote the set of all segments from S which lie in the interior of $\bigcup_{D \in \mathcal{F}} \bar{D}$. By induction, there is a binary encompassing tree $T_{\mathcal{F}}(x)$ for $S \cap \mathcal{F}$ and any vertex x of $\text{conv}(S \cap \mathcal{F})$ such that $\text{deg}_{T_{\mathcal{F}}(x)}(x) < 3$. Consider a region $D \in \mathcal{D}$ with $S \cap D := S \cap \{D\} \neq \emptyset$.

If vertex $p = t(D)$ is not assigned to any region other than D , add a tangent px_D from p to $\text{conv}(S \cap D)$, and add $T_{\{D\}}(x_D)$ to the tree T . Vertex x_D is pointed in the resulting tree, since px_D is tangent to $\text{conv}(S \cap D)$, and $\text{deg}_T(x_D) \leq 3$ because $\text{deg}_{T_{\{D\}}(x_D)}(x_D) < 3$. Vertex p remains pointed by (L7), and its degree is at most three by (L4).

It remains to consider the case that $p = t(D_1) = t(D_2)$ is assigned to two different regions D_1 and D_2 of \mathcal{D} . If one of $S \cap D_1$ or $S \cap D_2$ is empty, we can proceed as above. Hence, assume that both $S \cap D_1$ and $S \cap D_2$ are non-empty. We distinguish two cases.

Case 1: There is a line ℓ through p such that all edges incident to p in T and part of both $S \cap D_1$ and $S \cap D_2$ are on one side of ℓ (Fig. 6(a)). According to (L7) there are two tangents, $t_1 = px_1$ from p to $\text{conv}(S \cap D_1)$, and $t_2 = px_2$ from p to $\text{conv}(S \cap D_2)$, such that p remains pointed if both t_1 and t_2 are added to T . As above, we also add $T_{\{D_1\}}(x_1)$ and $T_{\{D_2\}}(x_2)$ to T and observe that all vertices involved satisfy the required degree and pointedness conditions.

Case 2: There is a line ℓ through p such that both $S \cap D_1$ and $S \cap D_2$ are on one side of ℓ and all edges incident to p in T are on the opposite side of ℓ (Fig. 6(b)). In this case $\text{conv}(S \cap \{D_1, D_2\}) \subset \text{int}(\bar{D}_1 \cup \bar{D}_2)$ follows from (L8).

Add a tangent px from p to $\text{conv}(S \cap \{D_1, D_2\})$ and $T_{\{D_1, D_2\}}(x)$ to T . The degree of p in the resulting tree is at most two by $(L4)$, and therefore p is also pointed.

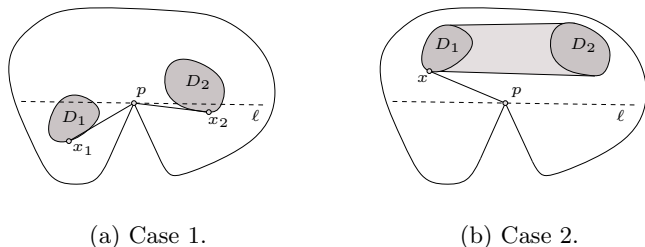


Fig. 6. Attaching encompassing trees to $p = t(D_1) = t(D_2)$.

After considering every region $D \in \mathcal{D}$, the black edges of the necklace P , the input segments, the tangents, and the encompassing trees for the subproblems, together form a pointed encompassing tree of maximum degree three. Since vertex x is not assigned to any region from \mathcal{D} , it has maximum degree two according to $(L1)$. \square

Analysis. The time complexity of our algorithm depends on the best available algorithms for the operations we use. We assume that our polygons are represented in a *doubly connected edge list*, and so we can easily replace an edge by a polygonal path or retract simple subpolygons of a necklace. The maintenance of the partition \mathcal{D} , and the labels t and γ require linear time and space. We also maintain a flag for every segment and vertex about their saturation status.

We use ray shooting queries in operations in both phases. Every ray is an extension of an input line segment. In the first phase we apply *Extend_reflex* if an extension \vec{r}_i of a line segment incident to a reflex vertex p_i hits another line segment lying in the interior of the face $D \in \mathcal{D}$ incident to p_i . An $O(n \log n)$ time sweep-line algorithm can precompute all pairs of segments such that the extension of one hits the other. We also maintain a dynamic point location data structure [8] for the polygons in \mathcal{D} with $O(\log n)$ update time and $O(\log^2 n)$ query time. For every ray \vec{r}_i , we look up the first segment s_i hit from the precomputed list, and if s_i is unsaturated, then a point location query for $\vec{r}_i \cap s_i$ tells whether s_i lies in the face incident to p_i . The total time spent on ray-shooting related operations in all first phases throughout the algorithm amounts to $O(n \log^2 n)$.

In the second phase, we partition every non-convex face $D \in \mathcal{D}$ along rays emanating from reflex vertices. Two line-sweep algorithms (left-to-right and right-to-left) complete this partition in $O(n_D \log n_D)$ time where n_D denotes the size of D . Thus the total runtime of all second phases throughout the algorithm is $O(n \log n)$.

Computation of geodesic paths, known as *shortest path homotopic to an input path*, was in the focus of recent studies [9,4]. The convex hull of segments lying in the interior of a polygon $D \in \mathcal{D}$ can also be computed by such a query. The total size of all geodesic paths is $O(n)$ in our algorithm, since every vertex on a geodesic path becomes saturated once only. Moreover, all our query paths are simple and pairwise non-crossing, and their total size is $O(n)$. Bespamyatnikh [4] can compute shortest homotopic paths in the presence of n point obstacles in $O(n^{4/3} \text{polylog } n)$ time if the total size of both the input and the output is linear. (Note that if the query paths were given *in advance*, then the geodesic paths could be computed in $O(n \text{polylog } n)$ time [4].)

Theorem 4. *A pointed binary encompassing tree for n disjoint segments in the plane can be computed in $O(n^{4/3} \text{polylog } n)$ time.*

6 Bounded Degree Pseudo-Triangulations for Segments

A careful analysis reveals that Theorem 1 holds in the following form.

Theorem 5. *For any finite set of disjoint line segments in the plane there exists a pointed binary encompassing tree such that the maximum degree is at most three and if a convex hull vertex has degree three then at least one of its incident edges is part of the convex hull.*

We combine this theorem with the algorithm of Aichholzer et al. [2], according to which a simple polygon can be pseudo-triangulated such that the degree of every convex vertex is at most four and the degree of every reflex vertex is at most five, that is, every convex (reflex) vertex has at most two (three) new incident edges in addition to the two incident polygon edges.

We apply this result to each polygon into which the tree of Thm. 5 dissects P and obtain an upper bound of 10. On the other hand, there is a lower bound construction that consists of ten disjoint segments and forces a vertex of degree at least six in every encompassing pseudo-triangulation (cf. the full version of [2]).

References

1. P. K. Agarwal, J. Basch, L. J. Guibas, J. Hershberger, and L. Zhang, Deformable free space tilings for kinetic collision detection, in *Proc. 4th WAFR*, 2001, 83–96.
2. O. Aichholzer, M. Hoffmann, B. Speckmann, and Cs. D. Tóth, Degree bounds for constrained pseudo-triangulations, *Proc. 15th CCCG*, 2003, pp. 155–158.
3. O. Aichholzer, C. Huemer, and H. Krasser. Triangulations without pointed spanning trees, in *Abstracts of 20th European Workshop Comput. Geom.*, 2004.
4. S. Bespamyatnikh, Computing homotopic shortest paths in the plane, *J. Algorithms* **49** (2003), 284–303.
5. P. Bose, M. E. Houle, and G.T. Toussaint, Every set of disjoint line segments admits a binary tree, *Discrete Comput Geom.* **26** (2001), 387–410.

6. P. Bose and G. T. Toussaint, Growing a tree from its branches, *J. Algorithms* **19** (1995), 86–103.
7. B. Chazelle, H. Edelsbrunner, M. Grigni, L. J. Guibas, J. Hershberger, M. Sharir, and J. Snoeyink, Ray shooting in polygons using geodesic triangulations, *Algorithmica* **12** (1994), 54–68.
8. S-W. Cheng and R. Janardan, New results on dynamic planar point location, *SIAM J. Comput.* **21** (1992), 972–999.
9. A. Efrat, S. Kobourov, and A. Lubiw, Computing homotopic shortest paths efficiently, in *Proc. 10th ESA*, vol. 2461 of LNCS, Springer-Verlag, 2002, pp. 411–423.
10. M. Goodrich and R. Tamassia. Dynamic ray shooting and shortest paths in planar subdivision via balanced geodesic triangulations. *J. Algorithms* **23** (1997), 51–73.
11. M. Hoffmann and Cs. D. Tóth, Alternating paths through disjoint line segments, *Inform. Proc. Letts.* **87** (2003), 287–294.
12. M. Hoffmann and Cs. D. Tóth, Segment endpoint visibility graphs are Hamiltonian, *Comput. Geom. Theory Appl.* **26** (2003), 47–68.
13. L. Kettner, D. Kirkpatrick, A. Mantler, J. Snoeyink, B. Speckmann, and F. Takeuchi, Tight degree bounds for pseudo-triangulations of points, *Comput. Geom. Theory Appl.* **25** (2003), 1–12.
14. D. Kirkpatrick and B. Speckmann, Kinetic maintenance of context-sensitive hierarchical representations for disjoint simple polygons, in *Proc. 18th Sympos. Comput. Geom.*, ACM Press, 2002, pp. 179–188.
15. M. Pocchiola and G. Vegter, Minimal tangent visibility graphs, *Comput. Geom. Theory Appl.* **6** (1996), 303–314.
16. M. Pocchiola and G. Vegter, Topologically sweeping visibility complexes via pseudo-triangulations, *Discrete Comput. Geom.* **16** (1996), 419–453.
17. D. Rappaport, Computing simple circuits from a set of line segments is NP-complete, *SIAM J. Comput.* **18** (1989), 1128–1139.
18. B. Speckmann and Cs. D. Tóth, Allocating vertex π -guards in simple polygons via pseudo-triangulations, in *Proc. 14th SODA*, ACM Press, 2003, pp. 109–118.
19. I. Streinu, A combinatorial approach to planar non-colliding robot arm motion planning, in *Proc. 41st FOCS*, IEEE Press, 2000, pp. 443–453.
20. Cs. D. Tóth, Alternating paths along orthogonal segments, in *Proc. 8th WADS*, vol. 2748 of LNCS, Springer-Verlag, Berlin, 2003, pp. 389–400.
21. M. Urabe and M. Watanabe, On a counterexample to a conjecture of Mirzaian, *Comput. Geom. Theory Appl.* **2** (1992), 51–53.